Regular Paper

A Secure Cloud-centric IoT Framework for Smart Device Registration

Songpon Teerakanok[1,2,3,a)] Tetsutaro Uehara[1,4,b)] Atsuo Inomata[1,4,5,c)]

Received: August 3, 2020, Accepted: February 2, 2021

Abstract: In this paper, a generic framework for IoT device registration is proposed. Unlike existing approaches, the proposed method is designed to provide a high level of compatibility, allowing it to work well with devices from different manufacturers. Furthermore, this framework requires only some commonly available technologies (like Bluetooth or BLE) to perform, making it highly applicable to most of the current generation of smart devices available in the market. With security and user-friendliness in mind, the developed registration protocol requires less user interaction while maintaining a considerable high level of security against various types of attacks, i.e., eavesdropping, replay attacks, modification, and man-in-the-middle attack.

Keywords: IoT security, internet of things, framework, network security, and cloud services

1. Introduction

The Internet of Things (IoT) has recently become part of our daily life. It is ingrained in our routines, introducing a new and smarter way to engage or interact with a machine [1]. Blending itself into our surroundings, IoT offers a higher quality of services to its users. There are a large number of IoT-related devices available in the market, ranging from personal gadgets (e.g., smartphone, Fitbit [*1]) and home appliances (e.g., smart thermostat, doorbell) to the industrial level automation.

Developing an IoT solution that is capable of communicating with a smartphone via Bluetooth or other wireless technology is relatively simple and more comfortable than before, thanks to today's publicly available libraries and APIs. However, developing an integrated IoT environment unifying all device controls into a single point (e.g., one software/application) or allowing devices from different vendors to communicate with each other is considered a challenging task.

1.1 Need for Standardization

Let us assume a user bought a Google Nest thermostat [*2]. He/she wants to register this newly brought thermostat and control it through the Amazon Alexa [2]. To make it possible, Amazon provides developers with a "skills" system [*3], which allows them to create applications using the Amazon-provided libraries and install these new skills into the user's Amazon account. For a big technology company like Google, building one or two additional applications for each of their products may not be a big problem. However, it is very different for some startups and small companies.

What if the same user wants to change his/her IoT control system from Alexa to HomeKit [*4], which is a smart home integration system provided by Apple? Can the same user command the Nest thermostat using his/her voice through Apple's Siri [*5]? Unfortunately, the answer is no, due to the incompatibility between Google Nest products and Apple's HomeKit application.

Currently, small companies and startups are usually questioned whether their products support Alexa, Siri, or Google Home. To successfully release a new product, the company needs to develop and maintain applications on various platforms to make sure that they can deliver high-quality services to all groups of customers. This is a very challenging problem regarding productivity and resource management. Each company needs to invest more time, human resources, and also money in developing their products, leading to the problem of limited productivity in the IoT industry.

Instead of following rules and proprietary protocols set by these big technology companies, what if developers, vendors and these large companies are required to implement their system to support only one generic/non-proprietary framework or a standard protocol? By following the framework or standard, it should allow makers or developers to creatively design and develop their IoT products to their full extent with much higher productivity.

1.2 Challenges in IoT Registration

The first step in providing compatibility between IoT devices

1 Cyber Security Laboratory, Kusatsu, Shiga 525–8577, Japan
2 Research Organization of Science and Technology, Ritsumeikan University, Kusatsu, Shiga 525–8577, Japan
3 Faculty of Information and Communication Technology, Mahidol University, Nakhonpathom, Thailand
4 College of Information Science and Engineering, Ritsumeikan University, Kusatsu, Shiga 525–8577, Japan
5 Osaka University, Cybermedia Center, Ibaraki, Osaka 567–0047, Japan
a) songpon.te@cysec.cs.ritsumei.ac.jp
b) t-uehara@fc.ritsumei.ac.jp
c) inomata@mail.osaka-u.ac.jp

*1 https://www.fitbit.com
*2 https://store.google.com/us/product/nest_thermostat_e
*3 https://developer.amazon.com/en-US/alexa/alexa-skills-kit
*4 https://developer.apple.com/homekit
*5 https://www.apple.com/siri

is to develop a generic registration process of IoT devices. Ideally, this IoT registration process should allow all IoT devices from different vendors to seamlessly register with the same user's account on an IoT management application of their choice (e.g., Google Home, Alexa, HomeKit).

Today's registration process is usually done by utilizing a predefined secret (QR [3], or secret SSID code, for example) that comes together with the product (often written in the manual). This approach offers an excellent way to register a newly bought IoT device into the user's smart home account (e.g., Amazon or Google account). These secret codes, however, can be lost. Also, users have to keep all these secret information safe all the time in case they want to sell their IoT devices to someone, or change/modify their account, which might require them to reinstall and register some devices all over again.

Aside from scanning the QR code, some companies (Apple, for example) provided an ultra short-range wireless technology-based (e.g., NFC [4]) registration process that requires no predefined secret information. Although the technique may require additional hardware support (NFC reader, for example), this registration technique is very efficient in terms of both security and user-friendliness. From a user perspective, the user can register a new IoT device easily by just opening a smart home management application on his/her phone and physically touch the target IoT device with his/her mobile phone to register the device. However, one drawback of this approach is that some IoT devices are located in the hard to access location (e.g., ceiling) rendering ultra short-range communication-based method impractical. Also, not all mobile devices come with such technology. Therefore, this is also another limitation of the mentioned registration process.

1.3 Contributions

In this paper, a generic framework for IoT device registration is presented. The framework is designed based on a typical cloud-centric IoT infrastructure that aims to provide a high level of compatibility between IoT devices, making it applicable to most IoT products, regardless of their brands or manufacturers. With the goal of high user-friendliness in mind, the main contribution of this work is to develop a secure and easy method for users to register their IoT products to their accounts. The developed framework is intentionally designed to create separation between IoT devices, the controller (i.e., software/application), and the cloud service. This approach makes each component replaceable and easier to manage. Lastly, the proposed communication protocol utilizes various cryptographic algorithms (i.e., hashing, encryption, and digital signature) offering a high level of security during data exchange between IoT devices, controller application, and cloud servers.

1.4 Organization

The rest of this paper is organized as follows. Section 2 introduces the current trend in IoT registration schemes. Next, Section 3 explains the concept of cloud-centric IoT architecture adopted in this work. In Section 4, the details of our proposed IoT framework are presented and briefly discussed. Next, results from our developed proof of concept system and some discussion are presented in Section 5. Sections 6 and 7 provide discussions regarding limitations and security aspects of the proposed framework respectively. Section 8 provides a brief discussion regarding user-friendliness aspects of our work. Finally, this paper is briefly summarized in the last Section 9.

2. Current trends in IoT Registration

In this section, trends and current approaches in IoT devices registration are discussed. At present, there are two main trends in IoT registration. Section 2.1 introduces the use of predefined shared secret information in the registration process. The following Section 2.2 presents a user-friendly alternative utilizing close-range communication technology in registering IoT devices to the user's account.

2.1 Shared Secret-based Approach

As the name suggested, a shared secret-based approach uses the predefined piece of information as a critical factor in securing an IoT registration process. This secret information usually comes in the form of codes, numbers, or digit sequences. QR code is an excellent example of techniques that belonged to this group.

This approach relies on the fact that each IoT device is hardcoded with some secret information. When the user issues a registration process by submitting the same or another piece of secret information that is consistent with the hardcoded information to the IoT device, the target device will trust and treat that user as its owner. After exchanging some further information with the device and the service provider (e.g., cloud server), the registration process is completed.

The method of registering IoT devices to the user's account using a shared secret is considerably straightforward and can be used with devices located in difficult to access location (e.g., high wall, ceiling). Depending on the product and design, it does not mean that this approach is considered user-friendly. Using shared secrets with IoT products having a screen or monitor is quite simple; for some devices, it becomes more complex and less user-friendly when a user is trying to register a product without a screen.

Let us give an example of a registration process of the smart air conditioner, Panasonic CS-X228C [*6], recently released in 2018. The user begins the registration process by turning on the wireless adapter on the device by pressing a small button located inside the hole (same as a reset button on a network router) on the remote control using a small pin. Second, the user then disconnects his/her mobile phone from the current wifi network. Next, using the Eolia application from Panasonic, the user connects his/her phone to Panasonic CS-X228C's wifi network using the SSID and password written in the manual. Lastly, the user finishes the registration process by selecting and providing a password to his/her home wifi network for the smart air conditioner to connect. As we will see, even though the predefined secret information is used, it does not guarantee the user-friendliness of the registration process.

[*6] https://panasonic.jp/aircon/p-db/CS-X228CS_spec.html

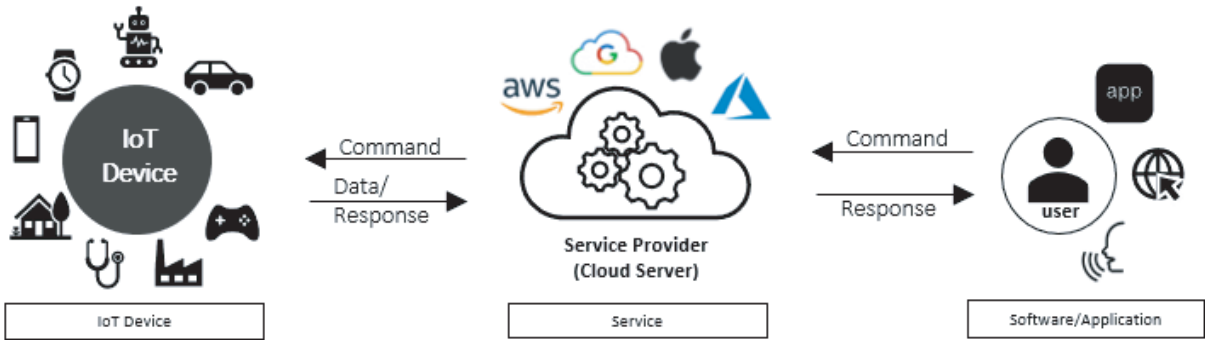

Fig. 1 A simplified overview of cloud-centric IoT architecture.

2.2 Close-range Communication Approach

To provide a better level of user-friendliness, close-range communication technologies, such as near field communication (NFC), or RFID, are utilized. The registration process begins with a user opening the mobile application on his/her smartphone. Starting the registration process from the smartphone, the user then physically touches the IoT device with his/her phone to complete the process.

This approach offers a high level of user-friendliness and security. Unlike the shared secret-based approach, this technique requires much less user interaction leading to a secure and smooth registration process. This method, however, is still considered less practical when using on devices located in hard to access locations (e.g., wall, ceiling).

Not only must the controlling device (e.g., smartphone) support these close-range communication technologies, such as NFC, IoT devices also need to be equipped with such technologies making this approach becomes impractical with some group of IoT devices.

3. Cloud-centric IoT Architecture

This section discusses the concept and details of the cloud-centric architecture of IoT technology, which is the basis of our research [5], [6]. **Figure 1** shows a simplified concept of a cloud-centric architecture of today's IoT technology. In this model, the entire system is separated into three primary components: devices, service, and application [7].

The device component represents IoT devices, including a wide range of devices such as sensors, home appliances, and industrial automation. Generally, each device may be equipped with different hardware making it supports diverse technologies and protocols. Some devices may support only wifi communication, while the other offers both wifi and Bluetooth or NFC.

The service component refers to the cloud services from any service provider (AWS from Amazon, Google Cloud, Microsoft Azure, and Apple iCloud, for example), including any proprietary cloud server. All major companies have their strategies, technologies, and communication protocols, which most commercial devices in the IoT business are required to support.

To control IoT devices, users issue a command to the devices through the control or management application/software via a service. There is a large number of IoT-related software in the market. Amazon Alexa, Google Assistant, and Apple's Siri are perfect examples of controlling applications that are currently available in the market. Many devices come with a particular control application of its own; Philips Hue light bulb, for example.

The communication procedure of the cloud-centric architecture is generally centered on the cloud server. On one side, IoT devices, such as sensors and home appliances, collect data and useful information from their surroundings and feed them to the cloud server. The server then uses this information to decide what to do with the devices automatically, e.g., locking the door when there is no one at home. On the other side, users monitor the working state (e.g., door locking status) and additional information about their IoT devices by pulling data from the cloud server. Additionally, users can issue commands over the internet through the cloud server to remotely manage or control their IoT devices. The command can be made through a mobile application, web service, or even voice control.

Regarding the development of IoT technology, the current state of IoT development is under intense competition. Big companies like Amazon and Google are trying to develop their system (i.e., devices, cloud service, and control application) and attempting to invite more developers to join them by providing API and libraries for developments. Since users will most likely choose to buy integrated IoT solutions from one of these companies, therefore, small developers have no choice but to create a system of their own while trying to support most of, if not all, technologies of these big companies. Otherwise, their products will be left unsold.

At this point, standardization appears to be a desirable solution to this problem of limited productivity in IoT development [8]. However, making a standard for the entire IoT environment is still an extremely challenging task. There are some available standards in the network-layer; MQTT v5.0 [9] for machine-to-machine (M2M) communication, for example. Also, in December 2019, Apple, Google, Amazon, Zigbee Alliance, and some board members formed a working group to develop an internet protocol (IP)-based open standard for smart home device communication [10]. This standard is expected to solve the incompatibility problem between devices at the network level; as a result, this IP-based standard may become one of the most crucial infrastructures for IoT development in the near future.

On the other hand, the problem of incompatibility in the upper level (i.e., services and application) remains unsolved, since there is currently no widely accepted standard for it. The problem in

service and application levels can be divided into two main parts: device registration (first contact between a user and his/her IoT device), and services handling (i.e., delivering services to a user after completion of device registration). In this paper, we focus only on the first part, the device registration process.

In this research, a generic framework and a secure communication protocol for IoT device registration are proposed. The proposed framework was designed to work with widely supported communication technologies like Bluetooth, BLE [11], and wifi, making it applicable to many currently available IoT technologies. The following Section 4 presents our proposed framework and communication protocol in detail.

4. Proposed Framework

This section presents and discusses our proposed IoT registration framework and its underlying communication protocol in detail. First, the overview of our proposed framework is introduced in Section 4.1, following with our developed registration protocol in Section 4.2.

4.1 Overview

Although this work is based on cloud-centric architecture, our proposed IoT registration process revolves around the user (user's control device such as a smartphone or smart home speaker like Amazon's Alexa, to be precise). In this section, the term user and user's control device will be used interchangeably.

The registration process begins by a user first discovering and verifying his/her ownership to the target IoT device via an insecure channel, by using some commonly available methods, such as Bluetooth, Bluetooth low energy (BLE) [12], or wifi. The user then asks the IoT device to generate a one-time secret key for secure communication, which will be used later in the last phase of the registration. Next, the IoT device sends the newly generated key to the cloud server via the user's control device. At this stage, the key is encrypted; the user is unable to read its content.

After checking the integrity of this one time key, the server then replies to the user with the verified secret key. Subsequently, a secure data exchange between the user and the IoT device is made using the previously confirmed key. This marks the completion of a registration process of an IoT device.

The registered IoT device can now communicate with the server directly by creating a new secret key from pieces of data obtained during the registration process. **Figure 2** shows an overview of an entire registration process.

4.2 IoT Registration Protocol

This section gives details of our proposed IoT device registration protocol. The protocol is divided into six different phases.

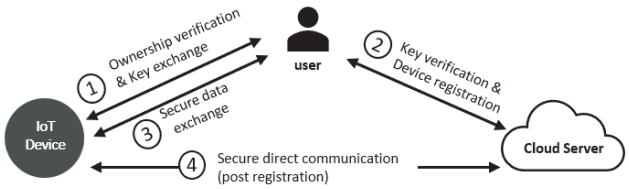

Fig. 2 An overview of our proposed IoT registration process.

The following Sections 4.2.1 to 4.2.6 explain each phase of our developed protocol in detail.

4.2.1 Phase 1: Discovery

The IoT registration begins with a user attempts to discover an IoT device located nearby. First, let us assume that all components (i.e., user, IoT and cloud server) have a pair of asymmetric cryptographic keys (public and private keys) of their own. Also, as an example, we will use Bluetooth as a means for a user to communicate with an IoT device. The method of communication can be changed and substituted with other techniques such as wifi, or Bluetooth low energy (BLE), for example. **Figure 3** shows the details of the proposed protocol in phase 1.

The protocol starts as soon as the IoT is powered, or the pairing/advertising button is pressed. The IoT device makes itself discoverable to the nearby control devices by broadcasting a predefined registration service id (RSID) information. In this paper, we define RSID as a specific UUID for IoT device registration service that is publicly known by everyone in the system (including attackers).

Subsequently, the user looks for available IoT devices by filtering broadcast messages with RSID. When the user found an IoT device, an insecure connection between the user and the IoT is made. Using the generated one-time random number (Ω), the user hashes and signs Ω and sends it to an IoT together with the user's public key (P_{b_U}) and an *INIT* request indicating the start of an IoT-side registration process. Finally, the IoT device replies to the user with a message, $E_{P_{b_S}}(S_k, H(\Omega))$, containing a secret key (S_k) and $H(\Omega)$ which are encrypted using the server's public key (P_{b_S}).

4.2.2 Phase 2: Server-side Initialization

As shown in **Fig. 4**, the second phase of our proposed protocol involves a data exchange between the user and a cloud server. It is assumed that the communication between the user and cloud server is made through a secure channel.

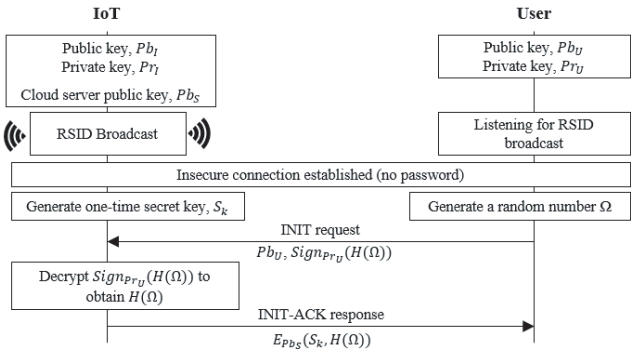

Fig. 3 Phase 1: IoT device discovery.

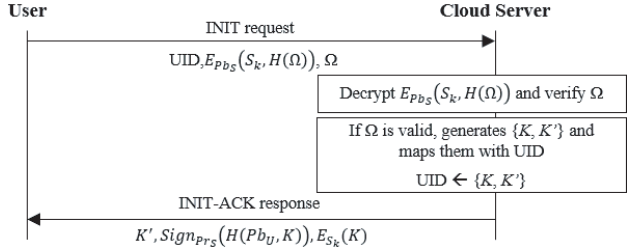

Fig. 4 Phase 2: server-side initialization.

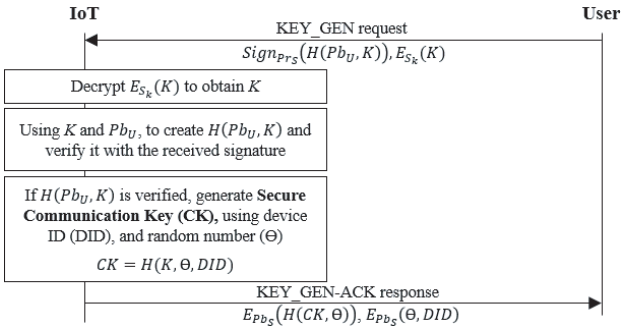

Fig. 5 Phase 3: key generation.

This phase starts by the user sending an *INIT* request together with the encrypted message, $E_{P_{b_S}}(S_k, H(\Omega))$, obtained from phase 1, Ω, and the user's id (*UID*) to the server. After decrypting the message and verifying the integrity of Ω with $H(\Omega)$ in the encrypted message, the server subsequently generates a pair of random numbers (K and K') and replies to the user with $Sign_{P_{rs}}(H(P_{b_U}, K))$, $E_{S_k}(K)$, and K'. Note that the server maps K and K' with *UID* and remembers them throughout the rest of the registration process.

4.2.3 Phase 3: Key Generation

In this phase, the user forwards $Sign_{P_{rs}}(H(P_{b_U}, K))$ and $E_{S_k}(K)$ to the IoT device. The IoT retrieves K by decrypting $E_{S_k}(K)$. Next, the IoT device verifies both the user's public key and K by computing $H(P_{b_U}, K)$, and then confirms it with a digital signature sent by the user.

Subsequently, if the verification succeeds, a secure communication key (CK) is generated using the IoT device's ID (*DID*), K, and a newly created random number θ. Lastly, the IoT replies to the user with two messages encrypted with the cloud server's public key, $E_{P_{b_S}}(H(CK, \theta))$, and $E_{P_{b_S}}(\theta, DID)$. **Figure 5** shows details of the key generation process.

4.2.4 Phase 4: Key Verification

This phase of the proposed protocol starts with the user forwarding two encrypted messages, $E_{P_{b_S}}(H(CK, \theta))$, and $E_{P_{b_S}}(\theta, DID)$, to the server together with *UID*, K', and a newly generated random number α.

Subsequently, the cloud server figures out K using the given *UID* and K'. Next, the server obtains θ and *DID* by decrypting $E_{P_{b_S}}(\theta, DID)$ with its private key. The server then generates the same secure communication key (CK').

Ideally, CK' should be identical to CK created in phase 3. After the key generation, the cloud server verifies the correctness of this key by computing $H(CK, \theta)$ and compares it with $H(CK, \theta)$ obtained from the encrypted message $E_{P_{b_S}}(H(CK, \theta))$. In case the confirmation is completed without any error, the communication key (CK') is sent back to the user together with some service-related information, such as details of the server's API for post-registration communication. This step marks the end of the server's side registration process. **Figure 6** shows the details of the key verification process.

4.2.5 Phase 5: Secure Data Exchange

At this stage, both the user and IoT have the same communication key (CK) and are ready for a secure data communication. Using CK as an encryption key or a password, the secure connection between the user and an IoT device is finally established.

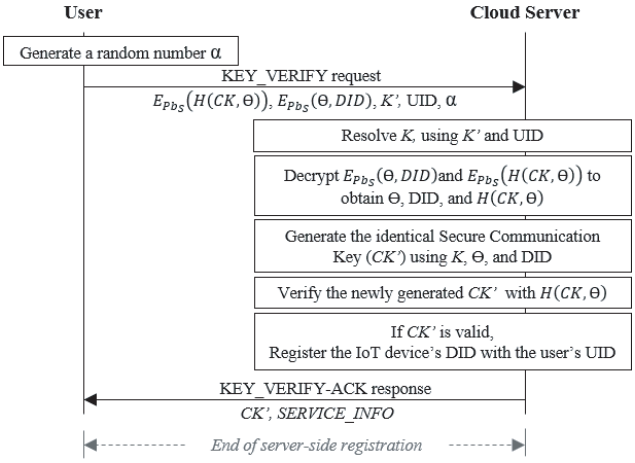

Fig. 6 Phase 4: key verification.

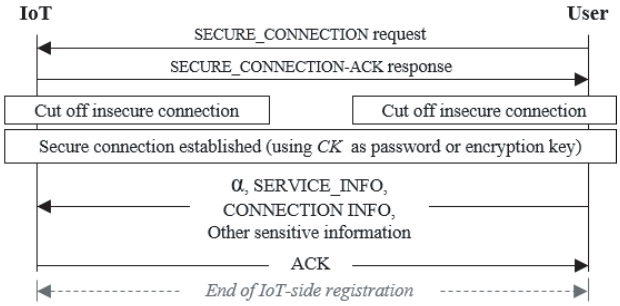

Fig. 7 Phase 5: secure data exchange between an IoT device and the cloud server.

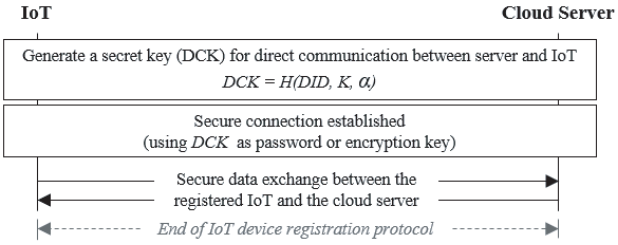

Fig. 8 Phase 6: post-registration.

The user then sends a random number α generated in phase 4, network-related information (e.g., wifi's SSID and its password) and other required sensitive information to an IoT device. The exchange of all confidential information between the user and IoT will be done here through the secure channel. Finally, This marks the end of the IoT device registration. The overall details of this phase are shown in **Fig. 7**.

4.2.6 Phase 6: Post Registration

This phase is, in fact, beyond the scope of our proposed registration protocol. However, this section will discuss and demonstrate how the IoT device can communicate with the cloud server directly through the internet.

As we knew, both the cloud server and the IoT received the random number α during phases 4 and 5, respectively. Therefore, we can utilize this information combining with some information known by both sides, such as *DID*, and K, to create a new secret communication key (DCK) known only by the device and the server. Subsequently, the IoT device can establish a secure connection and can now securely communicate with the cloud

server directly through this channel. **Figure 8** shows an example of processes during the post-registration phase.

5. Prototype & Results

We built a proof of concept system to demonstrate the practicability of the proposed framework and the developed registration protocol. The test system consisted of three main components, i.e., IoT device, a user's control device, and the server.

5.1 Testing Environments & Configurations

The prototype of an IoT device was achieved using a Raspberry Pi 3 (Model B) running Android Things OS (v. 1.0.15). We implemented a user's control device using Kotlin on a HUAWEI JDN2-W09 tablet, running Android 9.0 (API level 28). Lastly, a HTTP server is created using NodeJs (v. 12.13.1).

At the beginning of the registration process, the IoT device communicates with the user's control device using close range communication (e.g., Bluetooth, BLE) via an insecure channel. On the other hand, the experiment is tested with an assumption that the communication between user's control device and the cloud server is secured. Finally, after the registration, the IoT device can now establish a secure communication directly to the cloud server over the internet (see Fig. 2).

```
Server listening on port 3315 ...
add new user:
{
        uid: 7738,
        secretkey: "iotkey_secretlol",
        K: 3711,
        K_:5300
}
IoT Device Registration Completed!
==>> device: AX77 with user id: 7738
{ iot_heartrate: '53' }
{ iot_heartrate: '19' }
{ iot_heartrate: '52' }
{ iot_heartrate: '10' }
```

Fig. 9 The log results on a developed HTTP server.

5.2 Results

The following **Figs. 9** and **10** show the log results of our proof of concept system. According to Fig. 9, on the server-side, we started the test by running the HTTP server on port 3315. The server keeps listening to any incoming request. When the registration request arrives, the server records the user's information (e.g., *UID*). Once the registration is completed, the server uses a different API to send/receive data from registered IoT devices.

Figure 10 shows a snapshot of the process and the flow of data between an IoT device and the user's control device perspective. Initial data are shown in Fig. 10 and are retrieved from real devices during an experiment. However, we edited and formatted most of the data contents to provide better readability (i.e., most of the JSON and all encrypted data have been removed, and some sensitive information like wifi's SSID and password have been omitted).

We first start the test on the IoT-side by running a developed Android application on the device. As soon as the application is invoked, the IoT device begins to advertise itself on the Bluetooth network. At the same time, the user's control device also starts searching for a Bluetooth device broadcasting RSID information.

The registration process is completed once the IoT device ob-

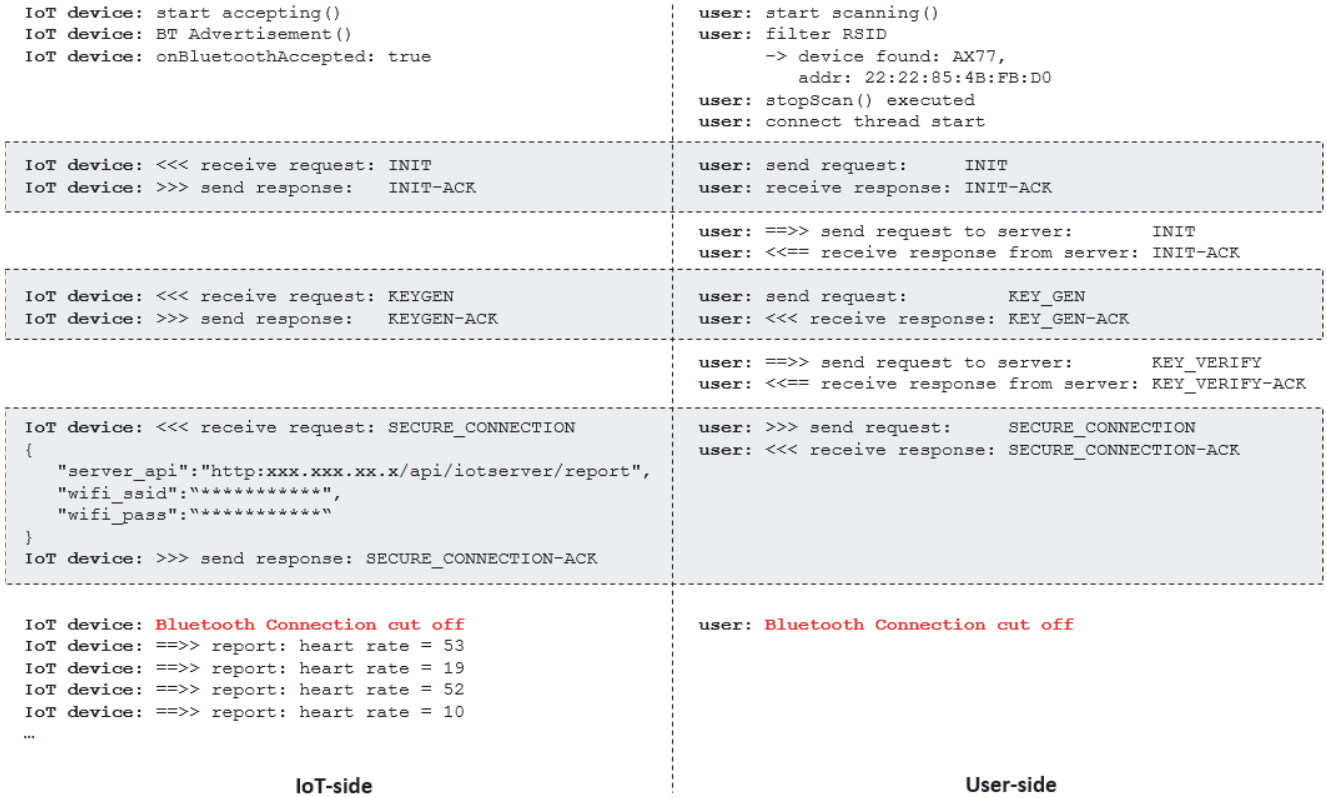

Fig. 10 The communication log between an IoT device (Raspberry Pi 3 running Android Things 1.0.15) and the user's control device (HUAWEI JDN2-W09 running Android 9.0 API 28).

Table 1 Security aspects of the proposed framework (compared with traditional approaches).

Methods	Security Aspects				Additional Resources
	Eavesdroping	Modification	Replay Attack	MITM	
proposed method	✓	✓	✓	✗	not required
shared secret	✓	✓	✓	✓	required (secrets)
close-range (e.g., NFC)	✓	✓	✓	n/a	required (hardwares)

Table 2 A breakdown of security aspects in our framework (see Fig. 12).

Attack Surfaces	Security Aspects			
	Eavesdroping	Modification	Replay Attack	MITM
C1	✓	✓	✓	✗
C2	✓	✓	✓	✓
C3	✓	✓	✓	✓

tains both the network-related information (i.e., wifi's SSID and password) and the service-related information (i.e., server's API). With SSID and password to the nearby wifi router, the IoT device can now connect itself to the internet. Furthermore, using the given server's API, the IoT will automatically know which server and API to send its data. As a result, after the Bluetooth connection between the user's control device and an IoT was cut off, the IoT device can automatically send its information to the server without involving the user's control device.

5.3 Discussions

In the previous Section 5.2, we demonstrate the real implementation of the proposed framework using a simple prototype. By utilizing the developed framework, we successfully divide all components into three main tiers; device tier (i.e., IoT devices), control tier (i.e., user's control devices and applications), service tier (e.g., cloud service). This concept of multi-tier separation provides benefits to the development of an IoT system in term of flexibility by getting rid of the tight coupling problem where devices from different vendors and platforms are not made compatible with each other. Using our proposed framework, developers can now focus on creating their products without worrying too much about the compatibility problems as long as they follow the proposed protocol.

In the following Section 6, we discuss drawbacks and current limitations of our framework. Furthermore, Sections 7 and 8 discuss the security and user-friendliness aspects of the proposed framework in details.

6. Design Constraints & Limitation

In this section, we discuss about the constraints and limitations of our framework. Currently, there are three major limitations in our design at the moment.

First, the developed framework is currently developed for the smart devices and gadgets (e.g., home/office appliances or personal gadgets) having a relatively high computational power. Due to the use of numerous cryptographic operations, this makes our framework not an appropriate solution for some industrial systems relying on ultra-low power devices, like sensors.

Second, the proposed method can only mitigate the man-in-the-middle (MITM) attack, but cannot completely solve the problem. In some situations where an attacker is already positioned nearby the victim (i.e., user) and timing the attack correctly,

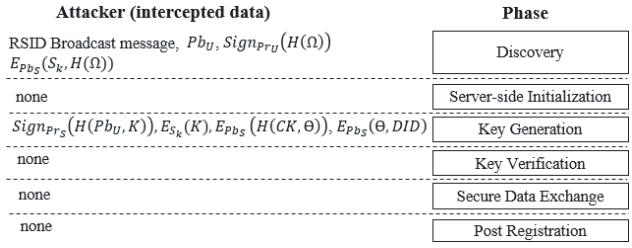

Fig. 11 Data intercepted by the attacker during each phase.

MITM attack can be successfully executed (see Section 7 for further details and explanations).

Finally, in term of user-friendliness, the proposed method is considered as a semi-automatic approach which still requires human interactions to complete the registration process; e.g., asking a user to type the network ssid and password. Therefore, the method is also prone to human errors. Section 8 explains this problem in more details.

7. Security Analysis

Previously, details and implementation of the proposed framework and its IoT registration protocol are introduced in Sections 4 and 5. In this section, we now discuss the security aspect of our developed protocol against various attacks. Some remaining risks and security challenges are also discussed. **Tables 1** and **2** provides a quick summary of the security aspects of our framework.

Before we begin, first, let us assume that the communication between the user and the cloud server is secured. Also, we assume that attackers can intercept all data exchange between the user and the IoT device. **Figure 11** shows data intercepted by attackers during each phase.

7.1 Attack Surface

As mentioned in Section 4.1, the proposed IoT registration framework consists of three components: the IoT device, the user's control device (also called "IoT Controller"), and the cloud server. **Figure 12** shows the attack surfaces associated with each component.

According to Fig. 12, there are many attack surfaces associated with each component. We divided these surfaces into three main groups: software, memory & storage, and communication. Each category also has several potential attacks of its own. For example, attackers may launch a supply chain attacks on the IoT device's firmware to allow them to gain root privilege or to allow

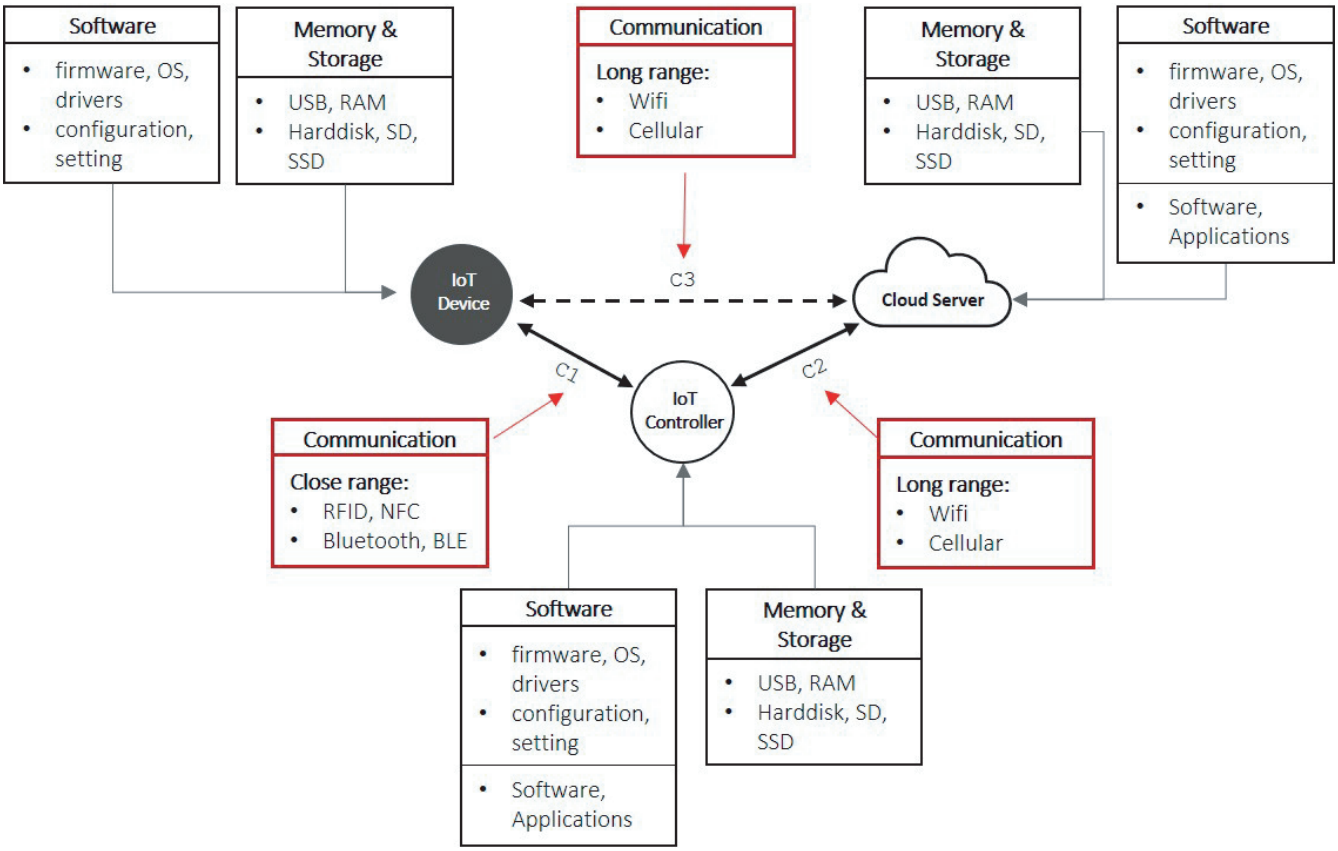

Fig. 12 Attack surfaces associated with each component.

them to exploit the device and system at will. There are several studies, prevention, and countermeasure techniques proposed in the literature regarding attack surfaces on software and memory & storage, which are not included in the scope of this paper.

In this work, we introduce a generic framework for IoT registration. Inside the framework, an underlying IoT registration protocol was presented. The goal of this protocol, according to Fig. 10, is to utilize the communication channel $C1$ and $C2$ to exchange information among all components (i.e., the target IoT device, a user (IoT controller), and the cloud server) in order to safely establish a new communication channel $C3$. Since the protocol relies heavily on the security of the communication channels $C1$, $C2$, and $C3$, in this section, we will focus our security analysis on potential attacks on theses channels.

7.2 Potential Attacks on Communication Channels

We categorized communication-related attack surfaces into two subcategories: long-range and close-range communication. As the name suggested, long-range communication-related attack surfaces allow attackers to launch their attack from far away (any distance). On the other hand, attacking using close-range communication (e.g., Bluetooth, RFID) requires the attackers to stay near the victims or the target machine to execute their attacks.

In this section, we will discuss security and potential attacks on the communication channel, $C1$, $C2$, and $C3$. Note that some and most common attacks (eavesdropping, for example) can be launched at both close and long-range communication. On the other hand, some attacks may be limited to a close-range communication channel only.

Eavesdropping (close & long-range). According to Fig. 11, among all information that an attacker can intercept, most of them are encrypted or signed, making it impossible or very difficult for an attacker to read or modify their contents.

Modification (close & long-range). The proposed protocol was designed with the concept of sharing incomplete information in which each party (i.e., IoT, user, and server) tries to give the other two parties digested (irreversible) information or data fragments. At the same time, that particular party is the only one that holds the original or complete piece of information. Therefore, if the attacker tries to modify or substitute any digested information, the other party that is holding the original data will notice automatically.

For example, there are two pieces of crucial information sent through an insecure channel during the discovery phase, i.e., the user's public key P_{b_U} and $H(\Omega)$. Regarding the discovery phase, the main objective of this stage is to send an IoT device a user's public key and one-time token $H(\Omega)$. $H(\Omega)$ is initially signed with the user's private key. Therefore, only the public key sent together with this token can be used to decrypt this message. This allows the IoT device to believe that the given token is associated with the user's public key.

At the end of the discovery phase, $H(\Omega)$ is sent once again, but in the encrypted form, $E_{P_{b_S}}(S_k, H(\Omega))$. Since the user knows the actual value of Ω while the adversary does not, he/she cannot secretly modify or substitute $H(\Omega)$ with other information.

Replay attack (Re-execution) (close & long-range). To pre-

Table 3 A comparison between the proposed framework and traidiontal approaches in term of user-friendliness and practicability.

Methods	User-friendliness Aspects		
	User inputs	Operational Distance	Additional Resources
proposed method	less	any	not required
shared secret	many	any	required (secrets)
close-range (e.g., NFC)	few	short	required (hardwares)

vent against replay attack, all messages (except the user's public key, P_{b_U}) transmitted through an insecure channel (i.e., during phases 1 and 3) contains at least one of the one-time random numbers: Ω, Θ, α, or K. Since these numbers are randomly generated and will be used only once, it is considered meaningless to replay any hashed or encrypted messages in our protocol.

Man-in-the-middle attack (MITM) (close range). In this work, commonly accessible wireless technologies like Bluetooth or BLE are adopted during the registration process. Although these wireless technologies can provide users with a smooth and user-friendly registration process; however, they also make the proposed method slightly prone to attack, such as MITM.

Let us assume that a user A has an unregistered (newly bought) IoT device D. The user A then turns on the device and tries to initiate the IoT registration process. To perform MITM, an attacker first waits for the IoT device to broadcast its RSID message to the network (in this case, the Bluetooth local network) meaning it is ready to be registered. The attacker then quickly responds to the IoT device and tries to perform a fake registration tricking the IoT device to think that it is being registered by its legitimate user. This fake registration will make the IoT device stop broadcasting its RSID message.

Next, the attacker will try to masquerade him/herself as the IoT device by broadcasting an RSID message to the network. The user seeing the RSID broadcast message will now think that the device broadcasting RSID message right now is the IoT device he/she wants to register in the first place. As a result, the attacker successfully gains control over the IoT device without the user noticing.

To summarize, MITM can be successfully executed when there is an attacker nearby listening to the Bluetooth RSID broadcast message and then launching an attack before the user initiates the IoT registration process. It can be said that MITM can be executed if the mentioned conditions are fullfilled and the timing is right. Unlike a secret-based approach or ultra-close range communication-based method (like NFC), our proposed method is considered prone to this type of attack. Limiting the advertising/discoverable duration of the IoT device, or reducing the signal strength to shorten the advertising range can help mitigate this problem by reducing the risk of being attacked. Furthermore, the IoT developers/manufacturers can also add a pairing button or a tiny LED signal light to their devices to assist users during the registration.

8. User Friendliness

In this section, we discuss the user-friendliness aspects of the proposed framework. To define the term user-friendliness, in this research, the IoT registration method is considered user-friendly if it satisfies the following three main requirements. First, the registration process should involve only little to no human interaction in order to reduce the chance of human error. Second, the procedure during an IoT registration should be simple and straightforward without any misleading steps. Lastly, the IoT registration process should be generic, meaning users should be able to apply the very same process with any IoT devices. Hence, there is no need for them to repeatedly learn the entire process everytime they try to register a new product. Additionally, the registration should be non-proprietary (or vendor neutral). Therefore, users are not required to have many different applications downloaded and installed for each individual IoT device; this will make managing and controlling IoT devices less complicated.

To evaluate the user-friendliness of the proposed framework, three major factors are put into consideration: 1) number of user inputs, 2) operational distance, and 3) additional resources requirement. The number of user inputs indicates how many times a user needs to give the input to the system. A higher quantity leads to a higher chance of human errors which can cause a failure during the registration. Next, the operational distance refers to how far (from the target IoT device) a user can be during the registration. This factor reflects the flexibility of the registration method especially with some IoT devices that are usually installed in the hard to access location like an air conditioner. Finally, additional resources requirement refers to whether the method requires additional resources in order to perform the registration or not. These additional resources may come in the form of information (e.g., secret code, PIN) or even additional pieces of hardware. **Table 3** provides a quick overview of our approach regarding the three mentioned factors.

In this work, the goal of our work is to minimize the amount of human interaction while still maintaining a high security. To make the IoT registration system user-friendly, the proposed method consists of only few steps and requires only a few user's inputs during the entire registration process. Here, we provide an example of how user-friendliness can be achieved in practice. We compare our approach with the shared secret-based approach (i.e., PANASONIC CS-X228C air conditioner) as a reference. **Table 4** shows a brief comparison between these two approaches.

In Table 4, we use the following four criteria to evaluate and compare user-friendliness aspects between the two approaches: 1) number of steps, 2) number of inputs, 3) loss of connectivity, and 4) vendor neutrality. The number of steps shows how many steps a user has to take in order to complete a registration for an IoT device. Similar to Table 3, the number of user inputs refers to how many times a user has to give inputs to the system (e.g., typing a password) to complete the registration. Third, in some case, an IoT device registration requires the user to disconnect his/her smartphone (or any control device) from the current wireless network in order to join a temporary network created by the

Table 4 User experiences (a case study of PANASONIC CS-X228C).

Methods	User-friendliness Aspects			
	Steps	User inputs	Loss of connection	Vendor neutral
proposed method	5	1 time	no	yes
shared secret (PANASONIC CS-X228C)	6	3 times	yes	no

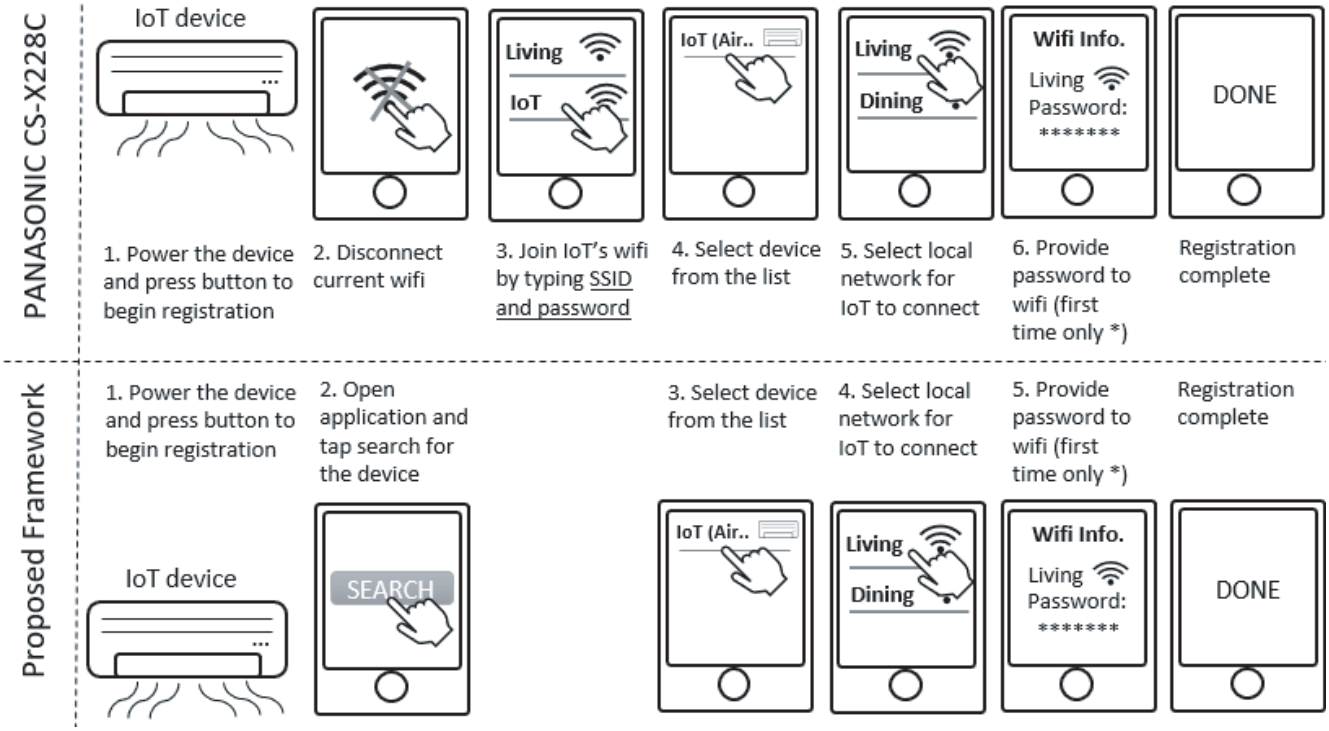

Fig. 13 A case study of PANASONIC CS-X228C compared with the proposed framework.

IoT device. This momentary loss of connectivity to the network or the internet can cause disruptions in the other services running on the user's control device. Therefore, we include this factor as one of the main aspect for evaluation. Lastly, as the name implied, vendor neutrality represents the whether the method is considered vendor specific (proprietary) or vendor neutral (non-proprietary). **Figure 13** demonstrates the registration process of our framework compared with the secret-based method.

According to Fig. 13 and Table 4, the shared secret approach (i.e., PANASONIC CS-X228C) requires a total of 6 steps with 3 user's inputs to complete the registration process. During the registration, the user needs to disconnect from his/her current wifi network in order to initiate the registration process. Furthermore, the user has to download and install the "Eolia" application from the vendor prior to the registration.

On the other hand, using the proposed framework, the user requires 5 steps to complete the registration (with only 1 time input). In addition, our approach does not require the user to join the temporary wifi network; it does not cause a disconnection or any disruption to other running services. The proposed method requires neither shared secret information or the proprietary application. The framework and its registration protocol can be adopted and implemented by any vendor/developer. Therefore, if application developers implement the same registration protocol in their IoT control applications (e.g., Alexa, HomeKit), this will allow users to choose the control application of their choice while still enjoying the benefit of cross-vendor compatibility.

9. Conclusion

In this work, a framework for IoT device registration is proposed. Unlike existing methods, our framework is designed as a generic framework to overcome the problem of incompatibility between devices, making it applicable to most of today's IoT devices and smartphones.

Utilizing existing technologies, the proposed method is designed to use commonly available communication technologies like Bluetooth while also requiring no predetermined secret information (such as codes or passwords) to perform. With the fact that our proposed method requires no special hardware supports (e.g., NFC, or RFID), this makes the proposed method more applicable to most devices in the market.

Regarding the developed registration protocol, the framework offers an IoT registration service with high user-friendliness while maintaining a high level of security against eavesdropping, replay attack, and modification. Also, even though our approach currently cannot provide full protection against the man-in-the-middle attack, we can reduce the risk and mitigate this problem efficiently by setting a discoverable duration and reduce the signal strength of the IoT device during the discovery stage.

References

[1] Mattern, F. and Floerkemeier, C.: From the Internet of Computers to the Internet of Things, *Lecture Notes in Computer Science*, Vol.6462, pp.242–259 (online), DOI: 10.1007/978-3-642-17226-7_15 (2010).

[2] Green, P.: Alexa, Where Have You Been All My Life? (2017).

[3] Tiwari, S.: An Introduction to QR Code Technology, *International Conference on Information Technology* (*ICIT*), pp.39–44, IEEE (online), DOI: 10.1109/icit.2016.021 (2016).

[4] Benyo, B., Vilmos, A., Kovacs, K. and Kutor, L.: NFC Applications and Business Model of the Ecosystem, *IST Mobile and Wireless Communications Summit*, pp.1–5, IEEE (online), DOI: 10.1109/ISTMWC.2007.4299324 (2007).

[5] Gupta, R. and Garg, R.: Mobile Applications Modelling and Security Handling in Cloud-Centric Internet of Things, *International Conference on Advances in Computing and Communication Engineering*, pp.285–290, IEEE (online), DOI: 10.1109/ICACCE.2015.119 (2015).

[6] Kantarci, B. and Mouftah, H.T.: Sensing services in cloud-centric Internet of Things: A survey, taxonomy and challenges, *IEEE International Conference on Communication Workshop* (*ICCW*), pp.1865–1870, IEEE (online), DOI: 10.1109/ICCW.2015.7247452 (2015).

[7] Happ, D., Karowski, N., Menzel, T., Handziski, V. and Wolisz, A.: Meeting IoT platform requirements with open pub/sub solutions, *Annals of Telecommunications*, Vol.72, No.1-2, pp.41–52 (online), DOI: 10.1007/s12243-016-0537-4 (2017).

[8] Wood, A.: The internet of things is revolutionising our lives, but standards are a must (2015).

[9] MQTT: MQ Telemetry Transport (2020).

[10] Apple: Amazon, Apple, Google, Zigbee Alliance and board members form working group to develop open standard for smart home devices (2019).

[11] Chandan, A.R. and Khairnar, V.D.: Bluetooth Low Energy (BLE) Crackdown Using IoT, *International Conference on Inventive Research in Computing Applications* (*ICIRCA*), pp.1436–1441, IEEE (online), DOI: 10.1109/ICIRCA.2018.8597189 (2018).

[12] Lee, J.-S., Dong, M.-F. and Sun, Y.-H.: A preliminary study of low power wireless technologies: ZigBee and Bluetooth Low Energy, *IEEE Conference on Industrial Electronics and Applications* (*ICIEA*), pp.135–139, IEEE (online), DOI: 10.1109/ICIEA.2015.7334098 (2015).

Songpon Teerakanok received his M.E. and D.Eng. degrees in Information Science and Engineering from Ritsumeikan University in 2016 and 2019, respectively. Currently, he is serving as an assitant professor at Research Organization of Science and Technology, Ritsumeikan University. Regarding his past research, he was also a former research student at the Centre for Network Research (CNR) (Prince of Songkla University, Thailand) from 2009 to 2013. He is now also a current member of the Cyber Security Laboratory at Ritsumeikan University. His research interest covers Cryptography, Privacy, Location-based Service (LBS), and Digital Forensics.

Tetsutaro Uehara received his B.E., M.E., and D.Eng. degrees from Kyoto University in 1990, 1992, and 1996, respectively. He was an assistant professor on the Faculty of Systems Engineering, Wakayama University from 1996 to 2003. From 2003 to 2005, he was an associate professor at the Center for Information Technology of the Graduate School of Engineering, Kyoto University. From 2006 to 2011, he was an associate professor at the Academic Center for Computing and Media Studies, Kyoto University. From 2011 to 2013, he was a Deputy Director of Standardization Division in the Ministry of Internal Affairs and Communication, Japan. He has been a professor at College of Information Science and Engineering, Ritsumeikan University from 2013. He has also been the vice president of the Institute of Digital Forensics from 2017. His research interest covers Systems Security, Digital Forensics, Privacy, Education in Information Ethics, and Information System Management in Local Government.

Atsuo Inomata received his Ph.D. in information science from Japan Advanced Institute of Science and Technology, Japan, in 2008. He was an associate professor of Nara Institute of Science and Technology until 2016 and then became a professor of Tokyo Denki University until 2019. Currently, he is serving as a professor at the cybermedia center and headquarters of information security, Osaka University and a visiting professor at Ritsumeikan University. Furthermore, he is a director of JPCERT coordination center (JPCERT/CC) and a representative director of Wireless LAN Certification Organization (WiCert). His research interest includes elliptic cryptography and IoT security. He is a holder of CISSP and RISS.