# A USER-FRIENDLY PRIVACY FRAMEWORK FOR USERS TO ACHIEVE CONSENTS WITH NEARBY BLE DEVICES

Seminar Report

*Submitted in partial fulfillment of the requirements for the award of degree of*

**BACHELOR OF TECHNOLOGY**

In

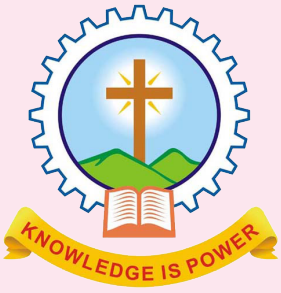
**COMPUTER SCIENCE AND ENGINEERING**

*of*

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**

Submitted By

**BINIL BIJU**




Department of Computer Science & Engineering
**Mar Athanasius College Of Engineering Kothamangalam**

# A USER-FRIENDLY PRIVACY FRAMEWORK FOR USERS TO ACHIEVE CONSENTS WITH NEARBY BLE DEVICES

Seminar Report

*Submitted in partial fulfillment of the requirements for the award of degree of*

**BACHELOR OF TECHNOLOGY**

In

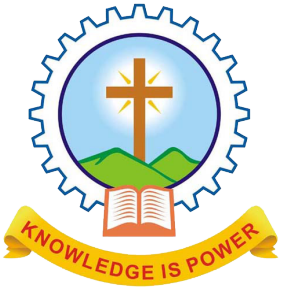
**COMPUTER SCIENCE AND ENGINEERING**

*of*

**APJ ABDUL KALAM TECHNOLOGICAL UNIVERSITY**

Submitted By

**BINIL BIJU**




Department of Computer Science & Engineering
**Mar Athanasius College Of Engineering Kothamangalam**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**
**MAR ATHANASIUS COLLEGE OF ENGINEERING**
**KOTHAMANGALAM**


# CERTIFICATE

*This is to certify that the report entitled* **A User-Friendly Privacy Framework For Users To Achieve Consents With Nearby BLE Devices** *submitted by* **Mr. BINIL BIJU, Reg. No.MAC15CS021** *towards partial fulfillment of the requirement for the award of Degree of Bachelor of Technology in Computer science and Engineering from APJ Abdul Kalam Technological University for December 2018 is a bonafide record of the seminar carried out by him under our supervision and guidance.*

| | | |
|---|---|---|
| ..... | ..... | ..... |
| **Prof. Joby George** | **Prof. Neethu Subash** | **Dr. Surekha Mariam Varghese** |
| ***Faculty Guide*** | ***Faculty Guide*** | ***Head Of Department*** |

Date:

Dept. Seal

# ACKNOWLEDGEMENT

*First and foremost, I sincerely thank the 'God Almighty' for his grace for the successful and timely completion of the seminar.*

*I express my sincere gratitude and thanks to Dr. Solly George, Principal and Dr. Surekha Mariam Varghese, Head Of the Department for providing the necessary facilities and their encouragement and support.*

*I owe special thanks to the staff-in-charge Prof. Joby george, Prof. Neethu Subash and Prof. Joby Anu Mathew for their corrections, suggestions and sincere efforts to co-ordinate the seminar under a tight schedule.*

*I express my sincere thanks to staff members in the Department of Computer Science and Engineering who have taken sincere efforts in helping me to conduct this seminar.*

*Finally, I would like to acknowledge the heartfelt efforts, comments, criticisms, co-operation and tremendous support given to me by my dear friends during the preparation of the seminar and also during the presentation without whose support this work would have been all the more difficult to accomplish.*

# ABSTRACT

The deployment of IoT devices with significant data collection capabilities around the world raises concerns about user privacy. People are worried about ubiquitous IoT devices collecting and sharing their data with unknown parties without their awareness or consent. There are several governmental agencies that have stated that IoT service providers should obtain user consent before collecting and using their personal data. But there is no standard means for users to reach agreements on privacy practices for IoT applications. The scenario considered is when people use their personal smartphones to access nearby IoT devices via Bluetooth Low Energy. To address the privacy issue in the scenario, this paper proposes a privacy preferences expression framework for BLE-based applications. The framework defines specifications for users to achieve agreements on privacy practices with nearby BLE devices. In addition, this framework provides guidelines for a device to process user requests according to the agreement.

# Contents

# List of figures

# List of abbreviations

| | |
|---|---|
| BLE | Bluetooth Low Energy |
| MAC | Media Access Control |
| GATT | Generic Attribute |
| UUID | Universally Unique Identifier |
| UPnP | Universal Plug and Play |
| P3P | Platform for Privacy Preferences Project |
| IoT | Internet Of Things |
| NFC | Near-field Communication |
| UDN | Unique Device Name |
| FTC | Federal Trade Commission |
| EU | European Union |

# Introduction

With the advances of IoT technologies, an increasing number of organizations can now deploy sensors, actuators and other IoT devices around the world to provide IoT application services. The Internet of Things refers to the ability of everyday objects to connect to the Internet and to send and receive data. It includes, for example, Internet-connected cameras that allow you to post pictures online with a single click; home automation systems that turn on your front porch light when you leave work; and bracelets that share with your friends how far you have biked or run during the day. For the first time, the number of "things" connected to the Internet surpassed the number of people. However, when people interact with the IoT devices, the devices may collect personal data. For example, operators of auto lighting control systems, such as Philips Hue, can collect lightbulb control events and use the information to derive the behavior patterns of users.

According to recent surveys [1], [2], there is a considerable proportion of people worried about the privacy risks of ubiquitous IoT devices with significant data collection capabilities. Such privacy concerns may become a critical obstacle to the growth and adoption of IoT applications. There appeared to be widespread agreement that companies developing IoT products should implement reasonable security. Of course, what constitutes reasonable security for a given device will depend on a number of factors, including the amount and sensitivity of data collected and the costs of remedying the security vulnerabilities. Commission staff encourages companies to consider adopting the best practices highlighted by workshop participants, including those described below. First, companies should build security into their devices at the outset, rather than as an afterthought. As part of the security by design process, companies should consider: (1) conducting a privacy or security risk assessment; (2) minimizing the data they collect and retain; and (3) testing their security measures before launching their products. Second, with respect to personnel practices, companies should train all employees about good security, and ensure that security issues are addressed at the appropriate level of responsibility within the organization. Third, companies should retain service providers that are capable of maintaining reasonable security and provide reasonable oversight for these service providers. Fourth, when companies identify significant risks within their systems, they should implement a defense-indepth approach, in which they consider implementing security measures at several levels. Fifth, companies should consider implementing reasonable access control measures to limit the ability of an unauthorized person to access a consumer's device, data, or even the consumer's network. Finally, companies should continue to monitor products throughout the life cycle and, to the extent feasible, patch known vulnerabilities.

In light of this, several governmental agencies, such as the US FTC and EU Article 29 Working Party, have stated that IoT service providers should obtain user consent before collecting and using personal data [3], [4]. However, most IoT applications may not properly inform and acquire user consent. For example, IoT application providers may just post notices on walls to notify users of the existence of IoT devices. This results in "low-quality" consents described by the report of the EU Article 29 Working Party and leads to users lacking trust in IoT devices.The Article 29 Working Party was an advisory body made up of a

representative from the data protection authority of each EU Member State, the European Data Protection Supervisor and the European Commission.

As major smartphone platforms, such as iOS and Android, support the BLE (Bluetooth Low Energy) specification, BLE has become a de facto standard in scenarios where a user employs his/her smartphone to access nearby IoT or wearable devices. This study focuses on this scenario and proposes a Privacy Preferences Expression Framework for BLE-based applications named PrivacyBat. The framework provides a standard format and method for administrators of IoT devices to present the privacy policies of their devices to the user. The framework enables device administrators to register their devices and privacy polices. Therefore, when users discover nearby BLE devices, they can find associated privacy policies via the interface defined in the proposed framework. The proposed framework also defines a standard means for users to notify BLE devices of their privacy preferences. The devices can then follow user preferences to process personal data. To the best of our knowledge, there is no standard or research on how users can reach an agreement with BLE-based IoT application providers on privacy practices. This study can hopefully contribute to the improvement of the quality of consent for BLE-based IoT applications.

# Background knowledge and related works

The preliminary knowledge related to BLE security and privacy, potential threats from privacy invasions and possible countermeasures are mentioned below.

## 2.1 Bluetooth Low Energy and its security and privacy mechanisms

BLE was merged into the Bluetooth standard with Bluetooth 4.0 Core Specification. Compared to the traditional Bluetooth specification, BLE provides a means for devices to communicate with one another with lower power consumption. While Bluetooth Low Energy is a good technology on its own merit, what makes BLE genuinely exciting and what has pushed its phenomenal adoption rate so far so quickly is that it's the right technology, with the right compromises, at the right time. For a relatively young standard (it was introduced in 2010), BLE has seen an uncommonly rapid adoption rate, and the number of product designs that already include BLE puts it well ahead of other wireless technologies at the same point of time in their release cycles. Compared to other wireless standards, the rapid growth of BLE is relatively easy to explain: BLE has gone further faster because its fate is so intimately tied to the phenomenal growth in smartphones, tablets, and mobile computing. Early and active adoption of BLE by mobile industry heavyweights like Apple and Samsung broke open the doors for wider implementation of BLE.

Perhaps one of the less visible key factors contributing to the success of BLE is that it was designed to serve as an extensible framework to exchange data. This is a fundamental difference with classic Bluetooth, which focused on a strict set of use cases. BLE, on the other hand, was conceived to allow anyone with an idea and a bunch of data points coming from an accessory to realize it without having to know a huge amount about the underlying technology. The smartphone vendors understood the value of this proposition early on, and they provided flexible and relatively low-level APIs to give mobile application developers the freedom to use the BLE framework in any way they see fit.

With a relatively easy-to-understand data model, no intrusive licensing costs, no fees to access the core specs, and a lean overall protocol stack, it should be clear why platform designers and mobile vendors see a winner in BLE. Like all things in engineering, good design is all about making the right tradeoffs, and Bluetooth Low Energy is no different. BLE doesn't attempt to be a solution to every wireless data transfer need, and classic Bluetooth, WiFi, NFC, and other wireless technologies clearly still have their place, with their own unique set of design tradeoffs and decisions.

Generally, a BLE-enabled device (or simply a device) can use the following means to communicate with other devices:

- A device can broadcast (or advertise) messages containing information to nearby devices. Therefore, a nearby device can receive the advertised messages.

- A device (or a central device) can connect to another device (or a peripheral device) to use services provided by the peripheral device

A major limitation of broadcasting, when compared to a regular connection, is that there are no security or privacy provisions at all with it (any observer device is able to receive the data being broadcasted), so it might not be suited for sensitive data. If you need to transmit data in both directions, or if you have more data than the two advertising payloads can accomodate, you will need to use a connection. A connection is a permanent, periodical data exchange of packets between two devices. It is therefore inherently private (the data is sent to and received by only the two peers involved in a connection, and no other device unless it's indiscriminately sniffing). Connections provides more information about connections at the lower level, and Roles discusses the corresponding GAP roles.

Connections allow for a much richer, layered data model. They also have the potential to use much less power than broadcast mode because they can extend the delay between connection events further out, or push large chunks of data out only when new values are available, rather than having to continually advertise the full payload at a specific rate without knowing who is listening or how often. Connections involve two separate roles:

- Central(master) : Repeatedly scans the preset frequencies for connectable advertising packets and, when suitable, initates a connection. Once the connection is established, the central manages the timing and initiates the periodical data exchanges.
- Peripheral(slave): A device that sends connectable advertising packets periodically and accepts incoming connections. Once in an active connection, the peripheral follows the central's timing and exchanges data regularly with it.

The bonding process is critical to BLE security and privacy mechanisms. After a central device connects to a peripheral device, either one of the devices can request to initiate the bonding process [3] and [4]. Before the two devices establish a bonding relationship, the two devices need to pair with each other. In the pairing process, the two devices will exchange security features, such as I/O capability, to decide a pairing scheme. There are four different pairing schemes:

- Numeric comparison. The two devices generate a six digit number mutually and display the number on their screens. Therefore, the owner of a device can authenticate that his/her device is pairing with the right device by checking whether the two devices display the same number.
- Passkey entry. One device displays a randomly generated six digit number and requests the owner of the other device to input the displayed number. The other device will then transfer the inputed number back to the first device for checking. Therefore, the former can ensure the owner of the latter device has seen the displayed number and input the number for authentication.
- Out of band. A device can generate a key and transfer the key to the other device using channels other than BLE, such as NFC and USB sticks. Then, the device can use the key to verify that it is pairing with the device that the key is delivered to.

- Just work.Once the devices exchange their public keys, the non-initiating device will generate a nonce, which is essentially a random seed value, and then use it to generate a confirmation value Cb. It then sends the Cb along with the nonce to the initiating device. At the same time, the initiating device generates its own nonce and sends it to the non-initiating device. The initiating device then uses the non-initiating device's nonce to generate its own confirmation value Ca which should match Cb. If the confirmation values match, then the connection proceeds.

A major hurdle to using BLE in secure applications has been that the most secure pairing methods have significant disadvantages in other areas. OOB pairing requires the device to have additional circuitry, raising the cost of the device and the designer must also guarantee that the OOB channel is secure, which can be a significant design challenge in and of itself. Numeric comparison requires each device to have a display, raising device cost, as well as have the user manually verify the codes match, which is detrimental to the user experience. Therefore it is reasonable to assume that most devices will use the passkey method or Just Works, which means that most devices will have some degree of vulnerability. Designers working on products with high security requirements, such as medical devices, should consider other wireless protocols if OOB pairing or Numberic Comparison cannot be implemented in their designs.

Each BLE device is identified using a device address. These addresses are similar to the MAC addresses used in other communications protocols however, it is usually possible to change BLE device addresses at will. Due to this similarity, it is common to see BLE device addresses referred to as BLE MAC addresses. BLE currently supports four different types of addresses all of which are 48 bits in length.

- Public IEEE Format- Purchased through the IEEE Registration Authority, these addresses are manufacturer specific. The 24 most significant bits of this address are the Organization Unique Identifier (OUI), aka company ID, and are assigned by the IEEE. The 24 least significant bits are free for the company to modify. Since these addresses do not change, they offer no protection from identity tracking.
- Random Static- These addresses are either burned into the device's silicon during manufacture or generated when the device power cycles. If the device generates a new address every power cycle and the user power cycles the device on a regular basis, then this address type offers some protection from identity tracking. Otherwise, the protection this address type offers is limited.
- Random Private Resolvable- This addressing method can only be used if the Identity Resolving Key (IRK) is exchanged between the two devices during the bonding process. With this method, the device will use the IRK to translate its device address in to a random address that appears in the advertisement packet. A second device that also possesses the IRK is then able to convert the random address back into the real address and identify the first device. In this method, the device will periodically generate a new random address based off the IRK, providing significant protection against identity tracking.
- Random Private Non-Resolvable- In this addressing method, the device address is simply a random number and a new device address can be generated at any time. If

new addresses are generated fairly often, then this method offers significant protection against identity tracking.

To sum up, BLE allows a device to verify the device it is pairing with if a numeric comparison, passkey entry, or out of band pairing scheme is used. No matter which scheme is adopted, the two devices will exchange information to generate a temporary key for further use. Then, a device can use the temporary key to exchange the following keys with the other device in the bonding process:

- A Long Term Key (LTK) for the device to encrypt data transferred to the other device. It is a persistent key that is stored in both devices and used to derive a fresh encryption key each time the devices go encrypted.
- A Connection Signature Resolving Key (CSRK) for the device to generate signatures of transferred data to enable the other device to verify the integrity of the transferred data
- An Identity Resolving Key (IRK) used by the device to generate random addresses to prevent others from tracking its address. To protect user privacy, the Bluetooth specification defines the random device address feature. Simply speaking, the random address feature enables BLE devices to change their Bluetooth MAC addresses so that others cannot track the devices based on their addresses. The random addresses can further be classified into non-resolvable addresses and resolvable addresses. A non-resolvable address is generated randomly and there is no way to identify the device using the address. If the device uses a resolvable address, then the other devices will be able to identify the device if they have the IRK used by the device to generate the resolvable address

Note that the original BLE pairing protocol is vulnerable to brute force attacks [8]. Malicious individuals may eavesdrop on the messages exchanged in the pairing process and derive the temporary keys and ultimately be able to eavesdrop on all future communication. To address this vulnerability, the Bluetooth specification has enabled two devices to use elliptic curve cryptography to establish secure connections for key exchanging from version 4.2 onward. Elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-EC cryptography (based on plain Galois fields) to provide equivalent security. Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other tasks. Indirectly, they can be used for encryption by combining the key agreement with a symmetric encryption scheme. They are also used in several integer factorization algorithms based on elliptic curves that have applications in cryptography, such as Lenstra elliptic-curve factorization.

The session encryption provided by BTLE is known to be relatively secure. BTLE uses AES-CCM, against which there are no known practical attacks. CCM mode (Counter with CBC-MAC) is a mode of operation for cryptographic block ciphers. It is an authenticated encryption algorithm designed to provide both authentication and confidentiality. CCM mode is only defined for block ciphers with a block length of 128 bits.

## 2.2 Threats of privacy invasion and countermeasures

This subsection discusses the threats of privacy invasion when people use their smartphones to access nearby BLE devices. As depicted in Figure 2.1, when a user communicates with a BLE-based sensor using his/her smartphone, a malicious BLE device may try to invade user privacy passively or actively.

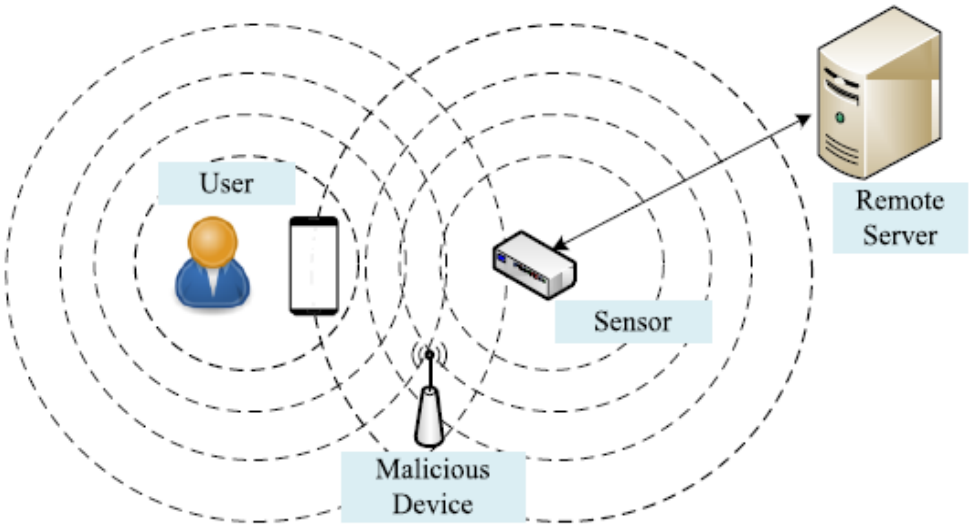



Figure 2.1: The scenario for threat modeling.

First, a malicious device may eavesdrop on messages transferred between the user's smartphone and the sensor passively. Then, the malicious device may obtain the smartphone's BLE MAC address or other advertised identifiers to track the user. A media access control address (MAC address) of a device is a unique identifier assigned to a network interface controller (NIC) for communications at the data link layer of a network segment. Bluetooth Address is usually displayed as 6 bytes written in hexadecimal and separated by colons. The upper half of a Bluetooth Address (most-significant 24 bits) is so called Organizationally Unique Identifier (OUI). It can be used to determine the manufacturer of a device (Bluetooth MAC Address Lookup form). OUI prefixes are assigned by the Institute of Electrical and Electronics Engineers (IEEE). If the sensor is a wearable device, the malicious device may be able to track the user via the MAC address of the sensor. To solve this issue, the Bluetooth specification has defined the random address scheme to prevent a device from being identified by unauthorized parties, as described in Chapter 2.1. Moreover, to address the problem that legacy BLE devices could not support the random address scheme, proposed the BLE-Guardian to "hide" a BLE device by invoking jamming to prevent adversaries from obtaining advertised messages of the device.

The malicious device may also collect personal data from messages transferred between the user's smartphone and the sensor. To mitigate the threat, the user's smartphone and the sensor can encrypt personal data before transferring the data. For example, the user's smartphone and the sensor can adopt the means defined in the Bluetooth specification to encrypt transferred messages.

A malicious device may invade user privacy actively. Even if a BLE device adopts the random address scheme, if the device responds to scanning requests issued by an another device, a malicious person may record the scanning requests and replay the requests to identify the device. Therefore, Ping Wang proposed to use counters to enhance the existing Bluetooth specification to overcome this vulnerability .It proposes a 3-way handshake protocol for nonce Rs deployment. There are two nonces Ra and Rs involved in the 3-way handshake protocol. The advertiser generates a nonce Ra as challenge sent to the scanner, which assures of freshness of the advertising session. Then the scanner generates a nonce Rs for advertising confidentiality and replay prevention. After the nonce Rs is deployed successfully from the scanner to the advertiser, the local counters Receiving (RX) and Transmitting (TX) on both sides are initialized to be Rs which protects all the following advertisement in the advertising session.To accommodate to open BLE advertising channels a handling mechanism of counter out-of-synchronization is given in system design. Moreover, to avoid unnecessary power consumption in the BLE devices then mitigation for Denial-of-service (DoS) is also proposed. In addition, advertising confidentiality, replay prevention, and antitracking of device have been simulated in Scyther and also been integrated into the code. The functional tests have been done in a realistic testing environment. The results show that the added functionalities work as designed.

GATT profiles enable extensive innovation while still maintaining full interoperability with other Bluetooth devices. The profile describes a use case, roles and general behaviors based on the GATT functionality. Services are collections of characteristics and relationships to other services that encapsulate the behavior of part of a device. This also includes hierarchy of services, characteristics and attributes used in the attribute server. Besides relying on the BLE authentication mechanism, BLE application providers may also implement their own authentication and access control mechanisms.

If a sensor provides GATT services that enable others to access stored personal data, unauthorized people may access the personal data if the sensor does not adopt an appropriate authentication and access control mechanism. GATT is built on top of the Attribute Protocol (ATT) (see Bluetooth Core System Architecture for block diagram and explanations), which uses GATT data to define the way that two Bluetooth Low Energy devices send and receive standard messages. Note that GATT is not used in Bluetooth BR/EDR implementations, which use only adopted profiles. First, after verifying identities of smartphone users, Internet services can generate credentials based on identification information of smartphones. Therefore, an IoT device can restrict that the smartphone with a specified identification information can only use associated credentials to connect to the device. Similarly, smartphone applications can use smartphone identification information to encrypt credentials for IoT devices communication. Then, smartphone applications need to use local identification information to obtain the credentials. If people copy the encrypted credentials to other smartphones, the smartphone applications cannot obtain credentials with incorrect identification information. On of the most critical deficiencies of the above schemes is smartphone applications may have trouble obtaining real identification information of the smartphones located. Furthermore, unauthorized people may steal personal data via physical attack. This study does not address the threats of physical attacks.

Even if there is no malicious device trying to collect personal data, when a person uses his/her smartphone to access a BLE device, the person may not be able to obtain the privacy

policies of the device. This may violate current personal data protection rules [5],[6]. To address this issue, [7] and [8] have proposed negotiation mechanisms for application providers to negotiate with users. Using these mechanisms, application providers can reach agreements with users on privacy practices. However, the proposed negotiation mechanisms do not apply for device to device BLE communication.

As there have been limited efforts focusing on enabling a BLE device to obtain user consent to collect and use personal data, this study provides a means for a user to obtain the privacy practices of a BLE device and send his/her privacy preferences to the device over BLE.

# Proposed framework

## 3.1 Overview of proposed framework

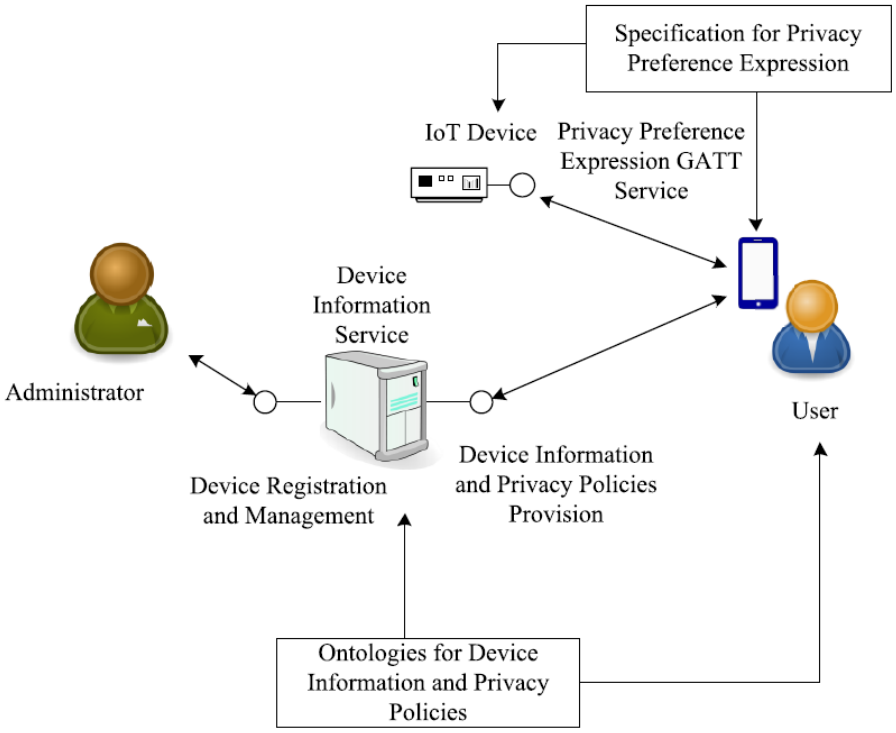



Figure 3.1: Overview Of proposed framework

Figure 3.1 gives an overview of the proposed PrivacyBat framework. The kernel of the framework is a Device Information Service. As illustrated in Figure 2, the Device Information Service provides two major interfaces:

- The Device Registration and Management interface enables an authorized device administrator to register a device using its device identity. Each device that supports the PrivacyBat framework has a unique 128-bit UUID (Universally unique identifier). The administrator can upload and manage device information and associated privacy policies by means of device identity via the interface. In this case, the framework defines Ontologies for Device Information and Privacy Policies based on UPnP and P3P, respectively.
- Each device that supports the PrivacyBat framework should advertise its UUID periodically, similar to iBeacons. Therefore, a person can collect UUIDs of nearby BLE

devices and query the Device Information Service through the Device Information and Privacy Policy Provision interface to obtain device information and associated privacy policies.

The PrivacyBat framework provides what we call the Specification for Privacy Preference Expression, which defines the Privacy Preference Expression GATT service. A PrivacyBat compatible device should implement this service. After receiving the privacy policies of a device, a user can notify the device whether or not he/she accepts the policies via its Privacy Preference Expression GATT service.

## 3.2 Ontologies

To help a person to determine whether or not to allow a device to collect and use their personal data, this study borrows from the UPnP device schema to provide device information. UPnP technology defines an architecture for pervasive peer-to-peer network connectivity of intelligent appliances, wireless devices, and PCs of all form factors. It is designed to bring easy-to-use, flexible, standards-based connectivity to ad-hoc or unmanaged networks whether in the home, in a small business, public spaces, or attached to the Internet. UPnP technology provides a distributed, open networking architecture that leverages TCP/IP and Web technologies to enable seamless proximity networking in addition to control and data transfer among networked devices. The technologies leveraged in the UPnP architecture include Internet protocols such as IP, TCP, UDP, HTTP, and XML. Like the Internet, contracts are based on wire protocols that are declarative, expressed in XML, and communicated via HTTP. Using Internet protocols is a strong choice for UDA because of its proven ability to span different physical media, to enable real world multiple-vendor interoperation, and to achieve synergy with the Internet and many home and office intranets. The UPnP architecture has been explicitly designed to accommodate these environments. Further, via bridging, UDA accommodates media running non-IP protocols when cost, technology, or legacy prevents the media or devices attached to it from running IP.

The UPnP Device Schema is written in XML and according to the conventions of XML Schema. XML Schema provides a method of describing the structure of an XML document. The XML Schema description language itself is based upon XML. The language is very robust; it specifies which elements are REQUIRED vs. OPTIONAL, element nesting, data types for values (as well as other properties not of interest here) and much more. The UPnP Device Schema uses these XML Schema constructions to define elements like ¡specVersion¿, ¡URLBase¿, ¡deviceType¿, et al listed in detail above. Because the UPnP Device Schema is constructed using a precise description language, it is unambiguous. As the UPnP Device Schema, UPnP Device Templates, and UPnP device descriptions are all machine-readable, software tools MAY be devised to validate the latter two, checking that they contain all the REQUIRED elements, are correctly nested, and have values of the correct data types. This study selects attributes in the UPnP device schema to describe a device. As shown in Figure 3.2, a device has a Unique Device Name (UDN) attribute to represent its UUID and a friendlyName attribute to provide short description for users. The manufacturer attribute of a device provides information on the device's manufacturer. Moreover, the deviceType, modelName, modelNumber, and serialNumber attributes describes the features of a device.

Users can obtain more detailed information germane to a device from the URL links in the manufacturerURL and the modelURL attributes. Users can also obtain surrounding images of a device stored in the iconList attribute. Therefore, users can locate the device based on the images and its location description.

The device information format proposed in this study extends the UPnP device schema in the following respects:

- In the proposed framework, a device may have one or more privacy policies. This study uses the policyList attribute to store identities of the privacy policies. Users can query the Device Information Service and retrieve the policies using their identities from the policyList attribute.
- Although the UPnP device schema has defined the serviceList attribute to describe the services of a UPnP device, the BLE GATT service is different from the UPnP service. Therefore, this study defines types of Services and Characteristics to describe GATT services provided by a device.

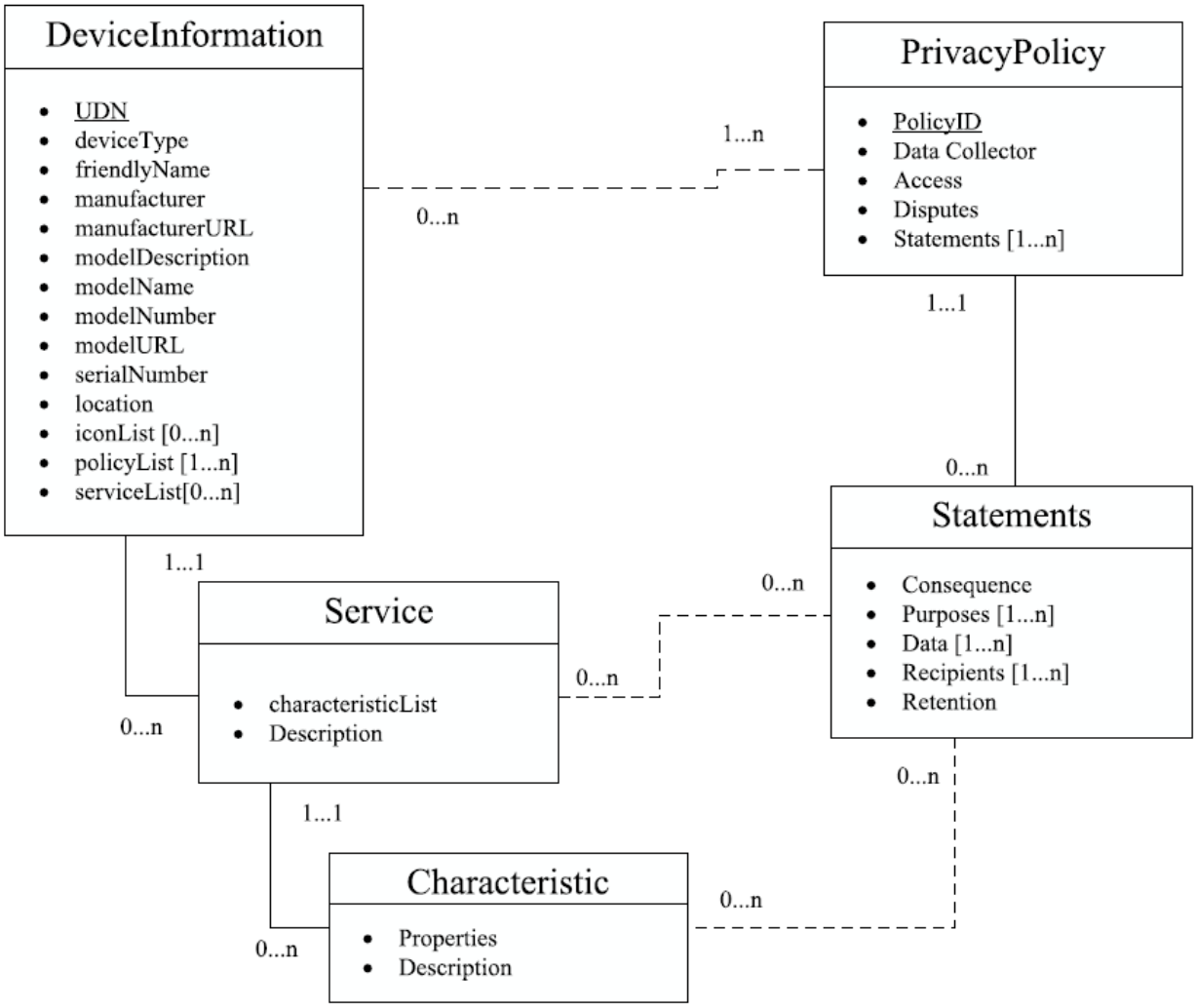



Figure 3.2: Data schema of device information and privacy policie.

This study defines the privacy policies based on P3P. The Platform for Privacy Preferences Project (P3P) enables Websites to express their privacy practices in a standard format that can be retrieved automatically and interpreted easily by user agents. P3P user agents will allow users to be informed of site practices (in both machine- and human-readable formats) and to automate decision-making based on these practices when appropriate. Thus users need not read the privacy policies at every site they visit. P3P uses machine readable descriptions to describe the collection and use of data. Sites implementing such policies make their practises explicit and thus open them to public scrutiny. Browsers can help the user to understand those privacy practises with smart interfaces. Most importantly, Browsers can this way develop a predictable behavior when blocking content like cookies thus giving a real incentive to eCommerce sites to behave in a privacy friendly way. This avoids the current scattering of cookie-blocking behaviors based on individual heuristics imagined by the implementer of the blocking tool which will make the creation of stateful services on the web a pain because the state-retrievel will be unpredictable.

Although P3P has become less popular recently, P3P is still the most well-known specification used to express privacy policies. Therefore, researchers use and adapt the specification in areas such as database accessing, e-Commerce, RFID applications, cloud computing, smartphone applications and so on.

For each privacy policy, a device administrator should specify who collects and uses personal data, the means for users to access collected personal data, and how disputes will be solved. The core of a privacy policy is a set of statements.A statement describes what personal data are collected and used for what purposes, with whom the collected data will be shared, and the retention period of the collected data. This study modifies the consequence attribute in P3P by linking the consequence attribute in a statement to associated GATT services and characteristics. Therefore, the administrator can limit user access to specified GATT services and characteristics if the user is not willing to accept related statements.

## 3.3 Privacy preference expression Generic Attribute service

In the PrivacyBat framework, every BLE device that collects and uses personal data should implement the Privacy Preference Expression Service. This study uses Figure 3.3 to illustrate the concept of the service. The Privacy Preference Expression Service contains three characteristics:

- Policy ID. The characteristic is a string and can be written by the user to specify to which policy the user wishes to express his/her preference.
- Action. An integer for a user to specify which action he/she wishes to adopt for the policy. Currently, the proposed framework provides three types of actions: First, a user can query the preference for the policy that is currently written in the Policy ID characteristic. A user can also express whether to accept or decline a privacy policy using this characteristic.
- Policy Preference. This characteristic exposes the preference (accepted or declined) of the policy specified in the Policy ID characteristic. The characteristic is readable and can send notifications in the event that changes occur during a connection.

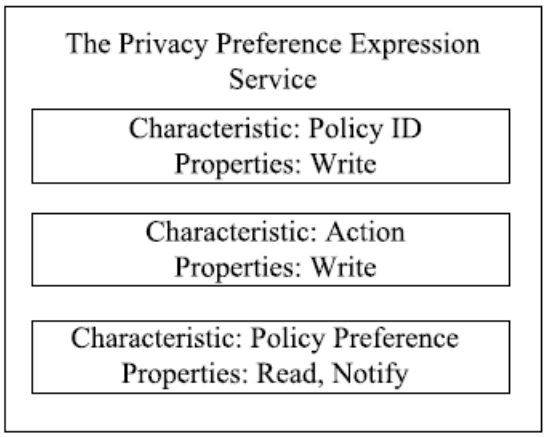



Figure 3.3: Concept of the privacy preference expression Generic Attribute service

Figure 3.4 illustrates the flowchart for a PrivacyBat compatible peripheral device to process a request from a user device to its Privacy Preference Expression Service. The peripheral device starts to handle a user request when it receives a write request to its action characteristic. The peripheral device first checks whether it can identify the user device that issued the request. The PrivacyBat framework uses a device's real bluetooth MAC address to identify a device. If the user device adopts the random address scheme and has not bonded with the peripheral device, the peripheral device will ignore the request.

This study assumes that each peripheral device has a table of tuples that contains DID( Device Idenifier ) and preferences of each policy. Also assumes that a device is related to privacy policies. For user device DID , the peripheral device will store the user preference for each policy. After obtaining the identity of the user device, the peripheral device checks whether the table contains the tuple to represent the preference settings of the user device. If the peripheral device cannot find the tuple of the user device, the peripheral device generates a new one for the user device and stores the tuple in its storage. Note that a peripheral device can only store privacy policy preferences for a limited number of user devices because of resource constraints. If the storage is full, the peripheral device may replace the least recently used tuple with the new one.

The peripheral device will then check the policy ID characteristic to determine the targeted policy of the request. The request will be ignored by the peripheral device if the policy ID in the characteristic is not valid. Finally, the peripheral device can handle the request based on request type: If the request is to query the privacy preference for a specified policy, the peripheral device will notify the user device with the value "accepted" or "declined" to reflect whether the user device has accepted the policy before. A user device can choose to decline a privacy policy that the user accepted previously and vice versa.

## 3.4 Request processing

This section describes how a PrivacyBat compatible peripheral device processes requests to normal characteristics ( or characteristics that do not belong to the Privacy Preference Expression GATT Service). As described in Section 3.2, the PrivacyBat framework allows

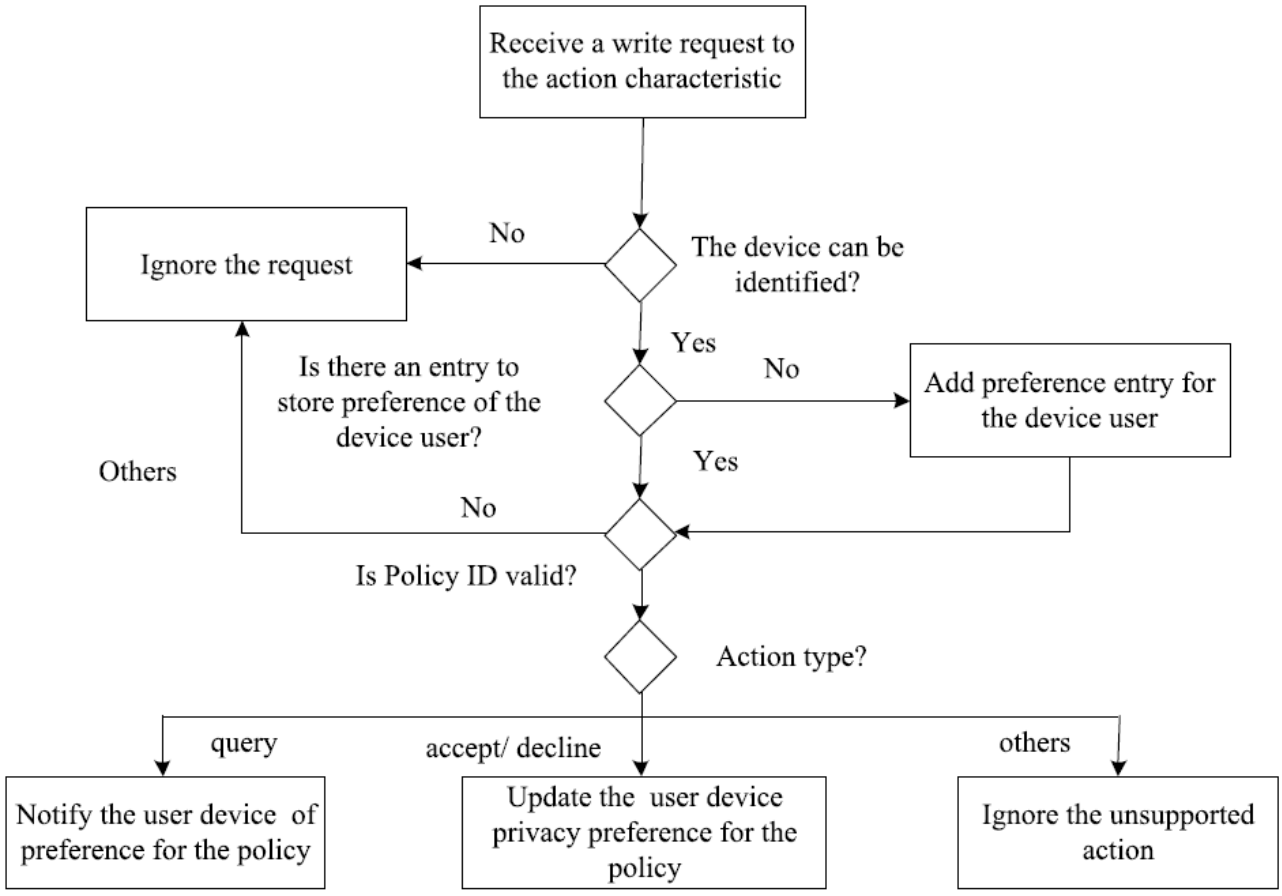



Figure 3.4: The flowchart for dealing with requests to the privacy preference expression service

device administrators to prohibit a user from accessing specified services or characteristics of a device if the user does not accept the associated privacy policies. Figure 3.5 illustrates the process for a PrivacyBat compatible peripheral device to handle requests to access normal characteristics.

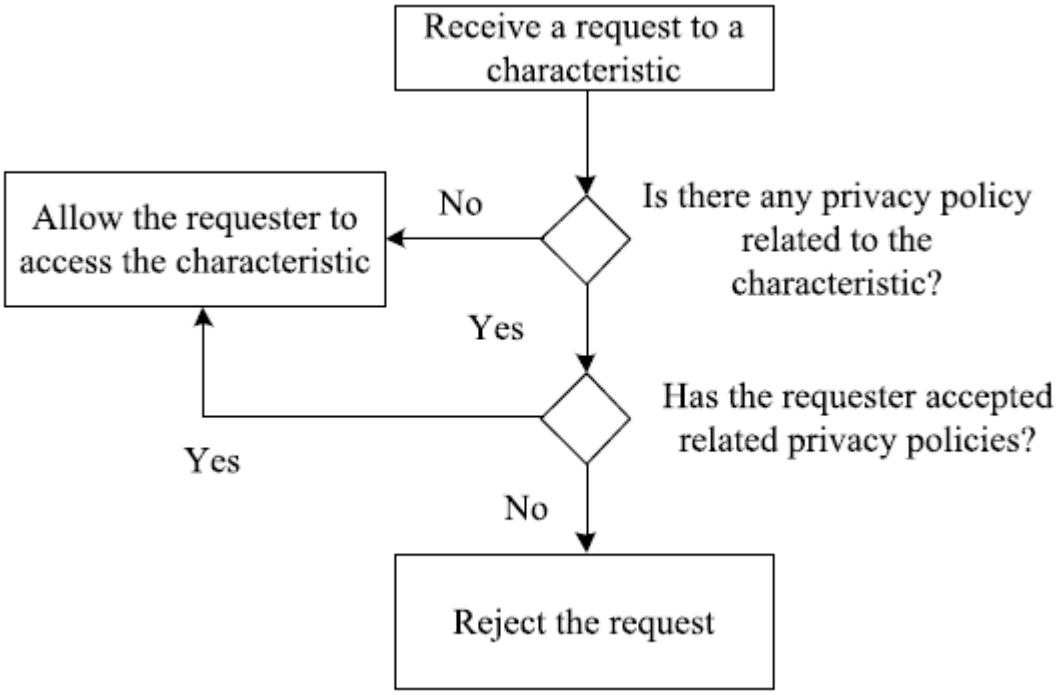



Figure 3.5: The flowchart for handling requests to normal characteristics.

After receiving a request from a user device to access a normal characteristic, a PrivacyBat compatible peripheral device will first look up which service the characteristic belongs to. Then, the peripheral device checks if there are privacy policies that have statements linked to the characteristic or service in the consequence field. As described in the previous section, the peripheral device records which policies the user device has accepted. Therefore, the peripheral device can determine whether or not to allow the request.

## 3.5 Proof of concept implementation

This study has implemented a prototype system to verify that the proposed framework has practical potential.As depicted in Figure 3.6, the Device Information Service is hosted on a desktop with Intel Core i7-4790 3.6GHz CPU and 16G RAM running Windows 10 professional. The service is implemented with Java Servlet, JDK 8(u131), Jetty application server version 9.3.6 and MySQL Community Database Server version 5.7.18. Note that although this study implements the service on a centralized server, it is possible to implement the service using a distributed architecture. The desktop is connected to a D-Link DIR 850L switch with Gigabit Ethernet. Therefore, user smartphones can connect to the switch through WiFi to communicate with the desktop. An experimental application is implemented and deployed on a Nexus 5X with Qualcomm Snapdragon 808 1.8GHz processor and 2G RAM

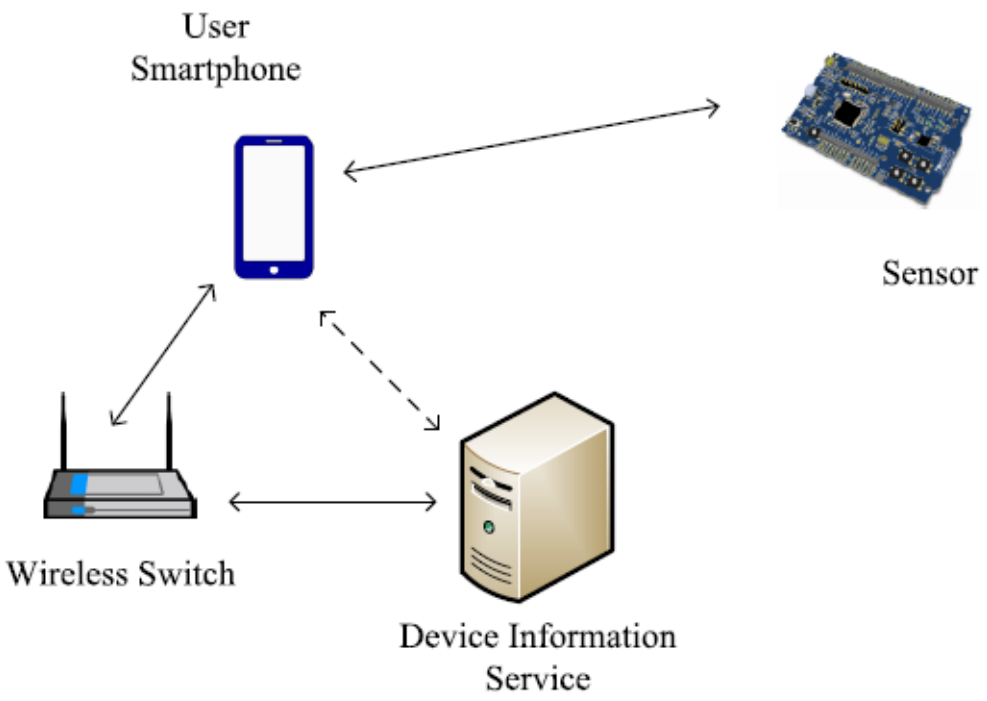



Figure 3.6: The experimental environment.

running Android 6.0.1. Finally, this study implements the Privacy Preference GATT Service on a Nordic nRF52 DK board with nRF52832 SOC to simulate a sensor or a IoT device that will be accessed by the user's smartphone.

In the experiment, the simulated IoT device provides an Air Conditioner GATT service (Figure 3.7). The Air Conditioner GATT service contains two characteristics: the Current Temperature characteristic showing the current temperature and the Power Switch characteristic enabling a user to turn on/off the air conditioner. The simulated IoT device advertises its 128-bit device identity periodically. The implemented Android application can be used to discover nearby IoT devices, obtain privacy policies of the devices if any, and express user preferences on the policies. After receiving a device identity, the application retrieves the device information using the received identity. The user can further use the smartphone to query the contents of privacy policies of interest. This study uses JSON to represent privacy policies rather than XML because JSON is more lightweight compared to XML. In our experiment, if a privacy policy has one statement, it has a size of about 2K bytes. A user smartphone can retrieve the privacy policy in less than 1 second (about 0.285 sec). Upon obtaining a privacy policy, the experimental application displays the privacy policy and lets the user choose between accepting or declining the policy by checking or unchecking the associated check boxes (Figure 3.8). Hereafter, the application can connect to the simulated IoT device to express the user's privacy preference.

As illustrated in Figure 3.7, the simulated IoT device also provides the Privacy Preference Expression service. This study measures the time taken for the experimental application to query the privacy preference of a privacy policy from the IoT device by averaging the result from 100 experiments. The average time of such experiments is 0.316 seconds.

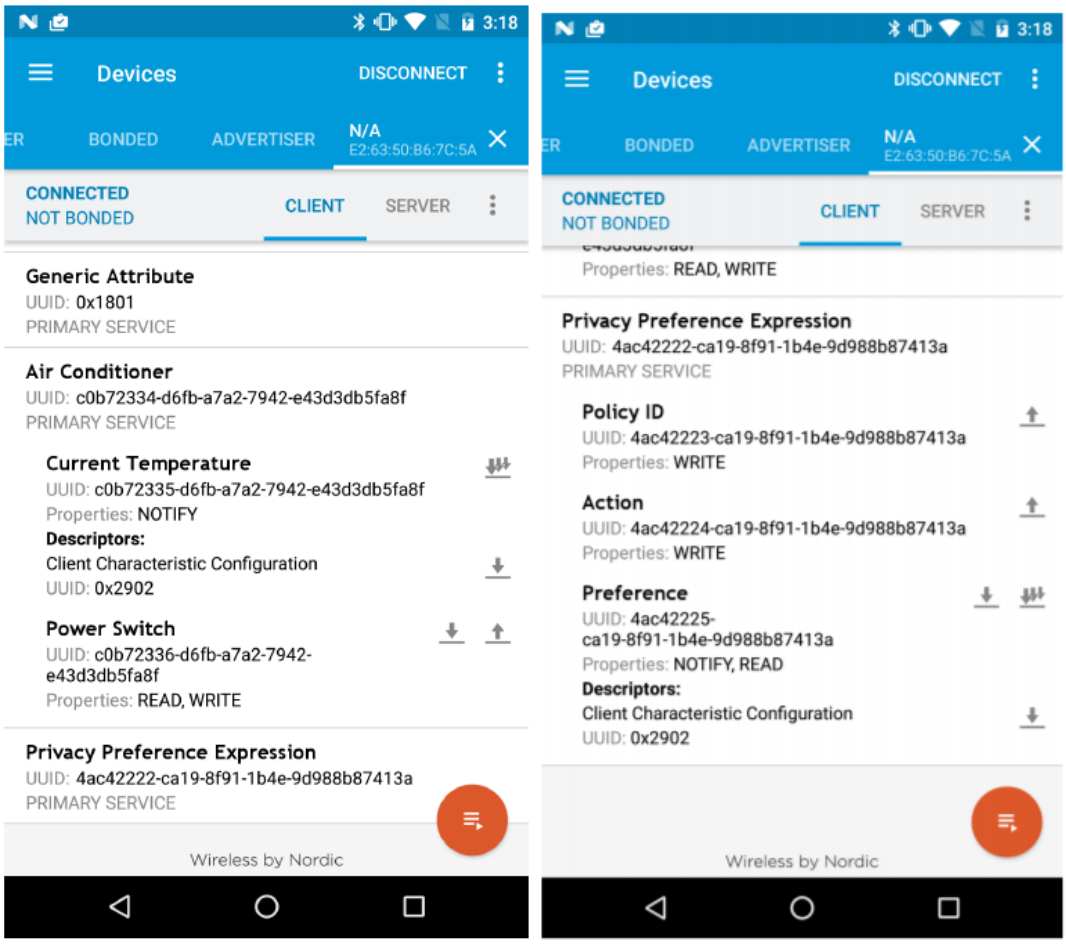



Figure 3.7: The services provided by the sensor device.

This study has certain limitations that point the way toward future research. First, legacy BLE devices may not be able to support the proposed framework. To address the issue, a gateway can be developed to connect to the legacy devices and provide the privacy preference expression services their behalf. Users can store their privacy preferences in the gateway. Therefore, when users wish to access the devices through the gateway, the gateway can play the role of a gatekeeper to control whether users can access the devices based on their preferences. In this case, designing and implementing such a gateway would be a challenging task. Second, this study only developed an experimental smartphone application to validate the proposed framework. However, this study does not consider the user's attitude toward the application. Usability tests need to be perfomed on the application to help improve the user experience. Last but not least, people may be curious about whether a device follows the accepted privacy policies. To address the issue, we can develop several kinds of tools to detect suspicious devices. For example, we can develop tools for a user smartphone to detect whether or not it transfers personal data to an IoT device. In addition, we can use tools like IoTScanner to monitor the outgoing traffic of an IoT device to detect whether it transfers personal data to remote hosts.the problem stated above, we now propose the IoTScanner. The aim of the IoTScanner is to provide realtime, passive monitoring of an existing wireless infrastructure that potentially constitute an IoT environment. The scanner will identify active devices that are communicating using that infrastructure, and attempt to categorize

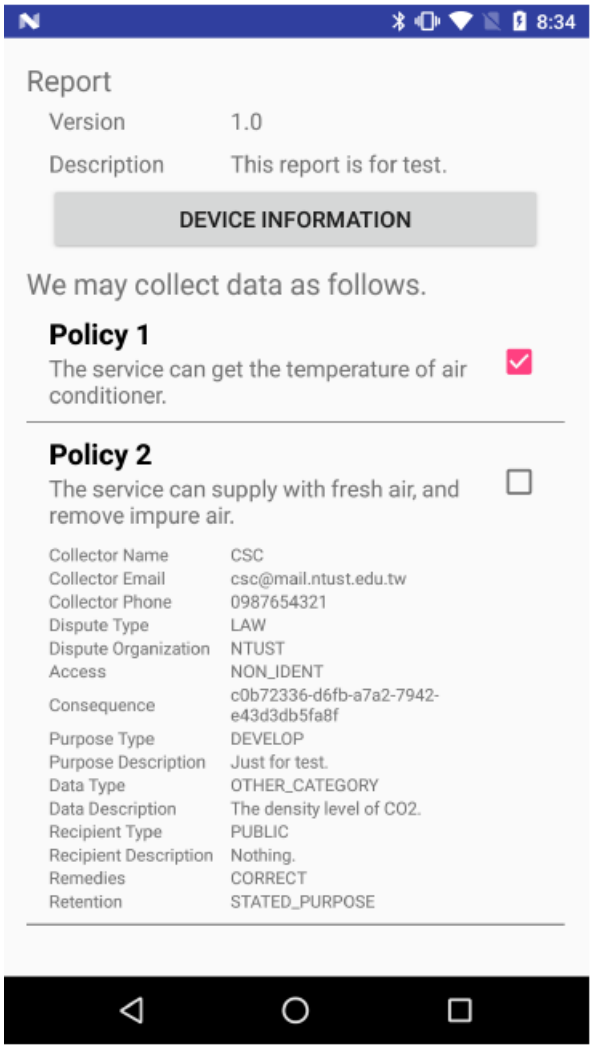



Figure 3.8: The screenshot of the experimental application.

IoT traffic depending on features such as the volume of traffic observed. The IoTScanner only observes the network traffic at the Link layer, and then analyzes this traffic using frame header information. A more offensive active scanner would not need to follow those constraints. However, users still cannot know how a device will deal with their personal data. In this case, it would be interesting future work to develop vetting systems on IoT devices.

# Conclusion

To provide a user-friendly means for users to achieve agreement on privacy practices with nearby BLE-based IoT devices, this study has proposed a Framework of Privacy Preferences Expression for BLE-based applications called PrivacyBat. The PrivacyBat framework provides a standard means for users to discover nearby devices and obtain device information along with associated privacy policies. In addition, the framework defines machine-processable ontologies of device information and privacy policies. Therefore, application developers can develop applications to display device information and privacy policies of nearby devices as user notifications. Moreover, the proposed framework offers the Privacy Preference Expression GATT service. A PrivacyBat compatible device should implement this service. Consequently, a user can connect to a PrivacyBat compatible device and express their preferences for received privacy policies through such a service. To demonstrate how the framework works, this study offers a proof of concept implementation and performs experiments to evaluate the performance of major operations. Experimental results indicate that the proposed framework can be implemented with commercially available products. As the proposed framework improves the process for IoT application providers to obtain user consent, this study can hopefully contribute to increasing user trust in IoT applications.

# References


[1] J. Groopman and S. Etlinger. (Jun. 2015). Consumer perceptions of privacy in the Internet of Things: What brands can learn from a concerned citizenry. Altimeter Group. Accessed: Feb. 9, 2018.

[2] V. Pureswaran and P. Brody, "Device democracy: Saving the future of the Internet of Things," IBM Inst. Bus. Value, Tech. Rep. GBE03620USEN, 2015, accessed: Feb. 9, 2018.

[3] EU Article 29 Data Protection Working Party, Opinion 8/2014 on the on Recent Developments on the Internet of Things, European Commission, Brussels, Belgium, 2014

[4] US Federal Trade Commission, The Internet of Things: Privacy and Security in a Connected World, Federal Trade Commission Staff Reports, DIANE Publishing Company, Collingdale, PA, USA, 2015.

[5] Bluetooth Core Specification, Bluetooth SIG, Inc., Kirkland, WA, USA, 2016

[6] R. Davidson, K. Townsend, C. Wang, and C. Cufí, Getting Started With Bluetooth Low Energy: Tools and Techniques for Low-Power Networking. Sebastopol, CA, USA: O'Reilly, 2014

[7] N. Gupta, Inside Bluetooth Low Energy. Norwood, MA, USA: Artech House, 2013, accessed: Feb. 9, 2018.

[8] M. Ryan, "Bluetooth: With low energy comes low security," in Proc. 7th USENIX Workshop Offensive Technol. (WOOT), Washington, DC, USA, Aug. 2013, accessed: Feb. 9, 2018.

[9] K. Fawaz, K.-H. Kim, and K. G. Shin, "Protecting privacy of BLE device users," in Proc. 25th USENIX Secur. Symp. (USENIX Security), Austin, TX, USA, Aug. 2016, pp. 1205–1221

[10] P. Wang, "Bluetooth low energy—Privacy enhancement for advertisement," M.S. thesis, Dept. Telematics, Norwegian Univ. Sci. Technol., Trondheim, Norway, Jun. 2014, accessed: Feb. 9, 2018.