 CMC, vol.59, no.2, pp.457-471, 2019

Developing a New Security Framework for Bluetooth Low Energy Devices

Qiaoyang Zhang[1] , Zhiyao Liang[1, *] and Zhiping Cai[2]

Abstract: Wearable devices are becoming more popular in our daily life. They are usually used to monitor health status, track fitness data, or even do medical tests, etc. Since the wearable devices can obtain a lot of personal data, their security issues are very important. Motivated by the consideration that the current pairing mechanisms of Bluetooth Low Energy (BLE) are commonly impractical or insecure for many BLE based wearable devices nowadays, we design and implement a security framework in order to protect the communication between these devices. The security framework is a supplement to the Bluetooth pairing mechanisms and is compatible with all BLE based wearable devices. The framework is a module between the application layer and the GATT (Generic Attribute Profile) layer in the BLE architecture stack. When the framework starts, a client and a server can automatically and securely establish shared fresh keys following a designed protocol; the services of encrypting and decrypting messages are provided to the applications conveniently by two functions; application data are securely transmitted following another protocol using the generated keys. Prudential principles are followed by the design of the framework for security purposes. It can protect BLE based wearable devices from replay attacks, Man-in-The-Middle attacks, data tampering, and passive eavesdropping. We conduct experiments to show that the framework can be conveniently deployed with practical operational cost of power consumption. The protocols in this framework have been formally verified that the designed security goals are satisfied.

Keywords: Bluetooth Low Energy, security, privacy, protocol, wearable devices, Internet of Things.

1 Introduction of the proposed security framework of BLE devices

1.1 Motivation

Bluetooth Low Energy (BLE) is first introduced in 2010 as part of the Bluetooth specification 4, which has a remarkable capability to keep devices working for a long time (months to years [Nick (2015)]). Comparing to Classic Bluetooth, which is used usually to establish short-range and continuous wireless connection, BLE is usually used for short bursts of some long-range radio connection [Bluetooth Special Interest Group

[1] Macau University of Science and Technology, Macau.

[2] National University of Defense Technology, Changsha, 410073, China.

[*] Corresponding Author: Zhiyao Liang. Email: zyliang@must.edu.mo.

(2017)], that is a reason why BLE is energy efficient. Therefore, most wearable devices use BLE in communication nowadays because of the size limit of batteries in these devices. In recent years, with the fast development of the Internet of Things (IoT), wearable devices are growing rapidly. Smart watches and wristbands and similar devices are booming in the market, since they can provide convenient services to people such as monitoring their health, tracking their fitness, and conducting medical tests. Wearable devices are indispensable in the personal area network (PAN) scenario, and they form the foundation of Medical Internet of Things (MIoT). IoT-based intelligent services depend on the data of various wireless devices that perform interoperability functions through technologies including RFID (Radio Frequency Identifier), GPS (Global Positioning System), NFC (Near Field Communication), BLE et al. Among these technologies, BLE is especially helpful to build intelligent human-oriented services because it supports many wearable devices. It is evident that the security and privacy issues of wearable devices are becoming more and more important. It is evident that the security and privacy issues of wearable devices are becoming more and more important.

The BLE specification allows two devices to conduct a process called pairing to exchange keys for data encryption of further communication. At present, there are four methods used in Bluetooth pairing [Bluetooth Special Interest Group (2016)], which are "Just Works", "Passkey Entry", "Out-Of-Band", and "Numeric Comparison with ECDH". The Numeric Comparison method is the most secure one because it uses the ECDH algorithm to prevent passive eavesdropping and uses human visual channel to prevent Man-in-The-Middle (MiTM) attacks. However, the Numeric Comparison method requires that both devices are capable of performing input and output (each device has a display and two buttons at least). In practice, most BLE based wearable devices, which are usually very small and lack complex hardware support, cannot fulfill these requirements. Hence, these BLE based wearable devices have to use one of the other three methods of pairing; however, these three methods are proved insecure [Ryan (2013); Qu and Chan (2016); Padgette, Bahr, Batra et al. (2017)].

In practice, paring and data encryption are not necessary in order to let two BLE devices connect with each other; doing so can save power consumption with the sacrifice of dropping security protection. Unfortunately, this could be a common case for the BLE wearable devices nowadays. In a recent research we have found that several popular BLE based wearable devices (smart bracelets) are vulnerable [Zhang and Liang (2017)]. By sniffing unencrypted packets, an attacker can easily steal sensitive data from these devices and control their reactions using fake commands.

In order to mitigate the disadvantages of the existing inconvenient or insecure pairing mechanisms, and to fix the vulnerabilities of the BLE based wearable devices, we decide to develop a security framework to protect the data transmitted via BLE. The security framework can be seen as a supplement to Bluetooth pairing mechanisms. It can protect BLE based wearable devices that cannot fulfill the requirements of the Numeric Comparison method. We do not aware of any other work that is similar to our research. The designed protocols of the framework have been formally verified using ProVerif [Blanchet (2016)], which is an automatic protocol verification tool (details are not included in this article because of the size limit). Chang and Shmatikov [Chang and

Shmatikov (2007)] presented a formal analysis of authentication of the Simple Pairing protocol of Bluetooth, also using ProVerif.

1.2 Related research

Security issues related to wearable devices can occur in different aspects, that are challenging when considering a broad scenario of IoT devices such as a network of MIoT, as discussed in the survey paper by Sun et al. [Sun, Cai, Li et al. (2018)]. Researchers have proposed solutions with different aims and approaches. Gong et al. [Gong, Huang, Li et al. (2015)] designed a new encryption algorithm that is suitable for IoT devices and can provide improved transmission success rate. Li et al. [Li, Yu, Zheng et al. (2013)] pro- posed a scheme for enhanced access control of personal health records (PHR) based on cloud services. Unlike these works, the framework that we propose is aimed at the simple and fundamental task of how to establish the security of direct communications between two BLE devices, such as a smart bracelet and a mobile phone. The framework does not assume any change, extension, or customized function provided by the underlying hardware, operating system, cloud service, or BLE architecture.

1.3 Design of the security framework

1.3.1 Architecture

Generic Attribute Profile (GATT) is a specification defining the way that two BLE devices transfer data. Information in a GATT profile is organized by the concepts of Services and Characteristics. In the protocol stack of BLE, the application layer sits above the GATT layer, where the proposed framework is between these two layers. An application will send data to, and receive data from, the security framework. The architecture of the security framework is shown in Fig. 1.

The security framework is built on above GATT and as a module of the application. It is transparent to the application. The security framework encrypts the data to be sent and decrypts the received encrypted data.

Since the security framework is built above GATT, it does not need to process any detail of BLE transmissions or connections, such as concurrent connections, timeout retransmission, error checking, etc. The link layer and the L2CAP (Logical Link Control & Adaption Protocol) layer will handle these details.

The operations of the framework have three parts:

1. Distributing initial information. The public key of a GATT server, such as a smart bracelet, will be obtained by a GATT client, such as a mobile phone. It should be done not by message transmission using a BLE channel. We propose that the public key of a GATT server can be encoded as a QR code which can be conveniently scanned by a client.
2. Establishing a fresh session key that is shared by the client and server.
3. Securely transmitting data. The session key will be used to encrypt the data transmitted between the client and server in a session.

Details of the parts 2 and 3 are discussed in the rest of this section.

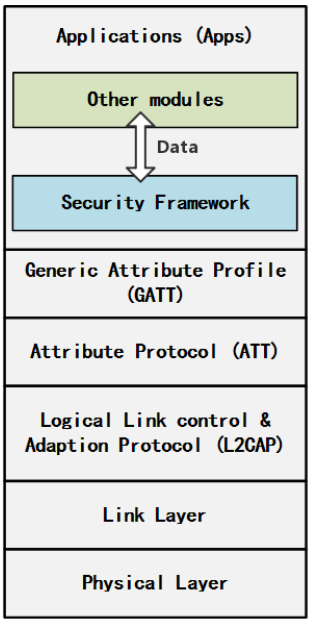

Figure 1: The architecture of the security framework

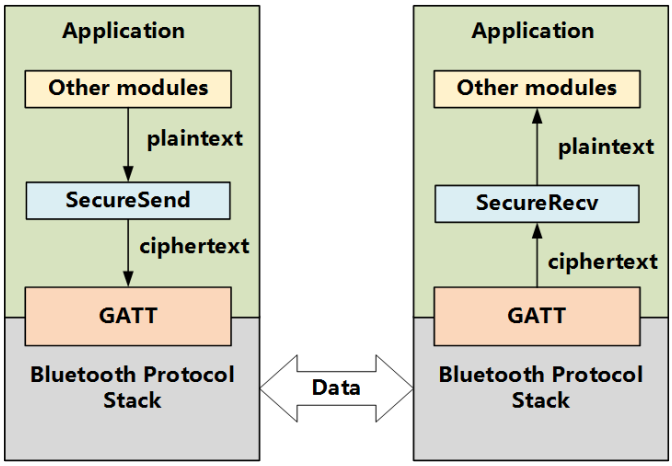

Figure 2: The architecture of securely transmitting data between two BLE devices

In our implementation, for the ease of using the framework, two functions *SecureSend* and *SecureRecv* are provided by the framework. When a session key is established, these two functions can be called by the modules of the application layer so that data are encrypted and decrypted transparently. The architecture of securely transmitting data between two BLE devices is shown in Fig. 2.

1.3.2 Cryptography

There are four cryptographic algorithms used in the security framework. The Advanced Encryption Standard 128-bit (AES-128) algorithm, the Elliptic Curve Diffie-Hellman (ECDH) algorithm, the Secure Hash Algorithm 256 (SHA-256) algorithm, and the Time-based One-Time Password (TOTP) algorithm.

AES-128 The AES-128 algorithm is used in the framework to encrypt or decrypt messages transmitted via BLE. This algorithm has a high level of security and it is also an officially supported algorithm in BLE specifications (including versions 4.0, 4.1, 4.2 and 5.0). The AES-128 algorithm is a symmetric key algorithm, which means that it uses the same cryptographic keys for both encryption of plaintext and decryption of ciphertext. Therefore, participants in BLE communication need to establish a cryptographic key before transmitting encrypted messages. Since it is not secure for transmitting a cryptographic key via BLE without encryption, another algorithm, TOTP, is used to generate a cryptographic key in the framework.

ECDH The ECDH algorithm allows two parties, each having an Elliptic Curve public-private key pair, to establish a shared secret key (SSK) over an insecure channel [Wikipedia (2017)]. Therefore, ECDH can be used for secure key exchange and anonymous key agreement. The SSK is generated from a local private key and a remote public key and is never transmitted via BLE in the operations of the framework. The public key and the shared secret key (SSK) will be used by the TOTP algorithm to generate a session key.

SHA-256 The SHA-256 algorithm is used to calculate the Hash-based Message Authentication Code (HMAC), which is used by TOTP in its computation.

TOTP The TOTP algorithm is used to generate a session key (SK), which is used by AES-128 to encrypt messages. A session key (SK) is computed using three terms: the public key of a GATT server, the SSK, and a time value. If a GATT server and a GATT client have the same three terms, they can generate the same SK. The computation of SK is described in Eq. 1. Therefore, both the GATT server and the GATT client can encrypt or decrypt messages without exchanging a SK before. Since a SK is never exchanged via BLE, it is protected from being stolen by an attacker during the execution of the framework.

$$SK = TOTP(servPubKey + SSK, time) \quad (1)$$

1.4 Security goals and principles

In this section, we describe the security goals that should be achieved by this security framework, and the principles that are followed by the design of this framework in order to achieve these security goals. Checking security goals for cryptographic protocols is generally a very difficult problem, undecidable with unbounded settings [Liang and Verma (2008)] and NP-complete with bounded settings [Liang and Verma (2009)]. On the other hand, by following prudential principles, some secure protocols can be designed.

Secrecy Valuable information (plaintext, application data) transmitted via BLE should not be revealed to attackers, even if an attacker can obtain some cyphertext (by eavesdropping). Secrecy is achieved by the following principles.

- All the ciphertexts transmitted via BLE are encrypted by the AES-128 algorithm, which is very secure. It will be impractical for an attacker to crack a ciphertext without knowing the key, since it may take about 1.02×10^{18} years [Arora (2012)].
- The keys that are used to encrypt and decrypt messages do not appear in the content of any messages sent in BLE; therefore, no way to obtain these keys by decrypting or analyzing the messages. The client obtains the public key of the server by scanning a QR code. The other two shared keys (a long-term key SSK and the session key SK) are known by the two parties by their internal computation using the ECDH and TOTP algorithms, not by reading received messages.

Authentication When a principal executes a protocol in a session, at a certain step, it will accept the received messages only if they are sent by the corresponding authorized interlocutor with the correct semantics according to the protocol; otherwise the message will be rejected. Authentication is achieved in the framework by the following considerations:

- **Fake messages**. Since the encryption keys of a regular user are not leaked to an unauthorized principal, and all the important messages are encrypted, an attacker cannot do attacks by creating some fake message that can be accepted by another principal.
- **Replay attacks**. The following mechanisms are used to prevent replay attacks.
  - When a client and a server are trying to establish a session key, each principal generates a fresh nonce, which is sent to and received from the other principal, in encrypted messages. This mechanism is called an authentication test [Guttman (2002)]. A replayed message cannot be accepted since it does not contain a fresh nonce.
  - Each session of data exchange uses a new session key (SK); therefore, messages from previous sessions cannot be replayed.
  - The messages in a session of data exchange contains a counter (an integer) which is monotonously growing; therefore, a message that is previously sent or accepted cannot be replayed in the same session.
- **Forward & backward security**. Even if a key of some round (SSK) or session (SK) has been leaked, attackers cannot crack any data transmitted before and after the round/session, because the encryption keys are freshly generated. The details of the freshly generated keys are listed as follows:
  - Each time when a client starts to connect with a server, the client generates a new pair of asymmetrical keys (*clientPubKey*, *clientPriKey*). The *clientPubKey* is sent to the server.
  - Based on the known keys and new keys, using the ECDH algorithm, both the client and the server freshly generate a new key SSK (a shared secret key), which is used to encrypt the messages that are designed to exchange a time value, and to perform authentication tests by exchanging nonces generated by the client and the server. Note that a new time value is generated and exchanged repeatedly after a certain regular time interval.
  - Based on the fresh SSK and the sequence of new time values, a sequence of short term session keys (SK) are generated by the client and the server using the TOTP algorithm. Each SK corresponds to a new time value. Each SK is used to encrypt the

exchanged (business) data between the client and the server.

Therefore, the security framework should guarantee forward and backward security.

- **Man-in-The-Middle attacks**. Since the term structure of the messages are different from each other (because of the different encryption keys and increasing counter values), an attacker cannot confuse a regular principal by forwarding messages around. Therefore, MiTM attacks such as the one to the Needham-Schroeder protocol [Lowe (1995)] are prevented.
- **Insider attacks.** Consider the case that an attacker is a legitimate client (an insider) of a device (GATT server), and the device is shared by multiple users. The attacker can obtain the public key of the GATT server (*servPubKey*) but cannot obtain the private key of another GATT client (*clientPriKey*), because each client will independently establish its keys for communication with the server device. Hence, the attacker cannot obtain and calculate the SSK and SK of any other client; therefore, the attacker cannot steal information from data transmission of regular agents or confuse them.

Availability When the security framework operates, if an agent receives some wrong message, it will drop the session and stop replying any message to the sender until some correct messages are received from the sender. This feature can prevent Denial of Service (DoS) attacks to a certain extent and maintain the availability of a BLE wearable device. On the other hand, the computation load of the framework is light, and no noticeable lag of performance is introduced. Therefore, the security framework should be easy to use and should work normally under some attacks attempts.

1.4.1 Design of the synchronizing time protocol

The business data of a client and a server will be encrypted by the AES-128 algorithm using a session key SK. Before the SK is generated by the TOTP algorithm, TOTP needs to obtain first a shared secret key SSK and a time value as the computation parameters. Therefore, the security framework needs to maintain some specific time values for the TOTP algorithm. We call such a value as a "framework time". Note that a framework time is just used in the security framework, not affecting any other time value used by the devices for functions outside the security framework that we propose.

Before a SK is generated, we need to make sure that the same SSK and the time value are shared by the client and the server; for this purpose, we design a new protocol called the "synchronizing time protocol", whose sequence diagram is in Fig. 3, which is described as follows:

1. Each GATT server holds a unique and fixed public-private key pair (*servPubKey*, *servPriKey*) of the Elliptic Curve Diffie-Hellman (ECDH) algorithm. The public key *servPubKey* must be sent to a GATT client via some non-BLE way. In the current design, the *servPubKey* is generated as a QR code. A GATT client obtains the *servPubKey* by scanning a QR code.
2. Each GATTclient will generate a fresh ECDH public-private key pair (*clientPubKey*, *clientPriKey*) every time when it starts the synchronizing time protocol. The public key *clientPubKey* will be sent to the GATT server.

3. The GATT client and the GATT server each computes a shared secret key (SSK) by the ECDH algorithm using the other's public key and its own private key. Both the GATT client and the GATT server have the same SSK. The GATT server generates a fresh nonce (*servNonce*), encrypts the *servNonce* using the AES-128 algorithm with the key SSK, and sends it to the GATT client.
4. The GATT client generates a fresh nonce (*clientNonce*), and encrypts its own time (*clientTime*), the *servNonce*, and the *clientNonce* together in a message with the SSK as the encryption key. Then, the GATT client sends the message to the GATT server.
5. The GATT server checks the received *servNonce* with its own *servNonce*. If they match, the GATT server set the received *clientTime* as its own framework time. Then, the GATT server encrypts the *clientNonce* using the SSK as the encryption key and sends it to the GATT client. Otherwise, if the nonces do not match, the protocol is aborted.
6. The GATT client checks the received *clientNonce* with its own *clientNonce*. If they match, the protocol is successfully completed. Otherwise, the protocol is aborted.

1.4.2 Design of securely transmitting data

When two devices transmit data to each other in a session, they maintain a counter. The counter will be increased by one when a device successfully sends or receives data. The procedure of securely transmitting data is shown in Fig. 4.

In order to securely transmit data via BLE, two functions are designed in the security framework: SecureSend and SecureRecv. An application forwards its plaintext to the function SecureSend, which will append the plaintext to the counter and encrypt these two terms (counter, plaintext) together in a message using a session key (SK) that is generated earlier by the TOTP algorithm as the encryption key. Then, the application sends the ciphertext message to the other device. After that, the sender increments the counter by one.

The other device receives the ciphertext and forwards it to the function SecureRecv, which decrypts the ciphertext using the key SK (generated earlier by the TOTP algorithm) and extracts the counter and the plaintext. The function SecureRecv checks if the received counter is equal to its own counter. If the two counters match, the counter of the receiver will be incremented by one, and the plaintext will be forwarded to the application. Otherwise, if the two counters do not match, the function SecureRecv will drop the received message and return error.

It is impossible to use a message from another session with the same expected counter value for some attacking purpose since the SK is fresh and unique in each session.

1.4.3 Advantage features

The security framework can provide very strong cryptography protections with low power consumption, and its code size is tiny. All data transmitted via BLE are encrypted and prevented from replay attacks, Man-in-The-Middle attacks, data tampering, etc. Convenience of using is another advantage of the security framework. A user only needs to scan a QR code (using a camera-capable device like a mobile phone) to start secure data transmission with another BLE device.

1.4.4 Compatibilities

Since the security framework is a module of the application, the deployment of the security framework does not need to make any change to the BLE protocol stack. Hence, the security framework is compatible with all versions of the BLE specification (from 4.0 to 5.0). For now, the security framework has been tested and it is completely compatible with nRF51/nRF52 series BLE chips and Android smartphones.

2. Implementation of the security framework

The implementation of the security framework is divided into two parts. One is for GATT servers and the other is for GATT clients. The part for GATT servers is implemented in the C language while the GATT clients' part is implemented in the Java language. Developers can call the classes provided by the framework when they are needed. Both parts of the security framework are very easy to use because developers only need to add a few lines of code to the code of other BLE services in order to use them.

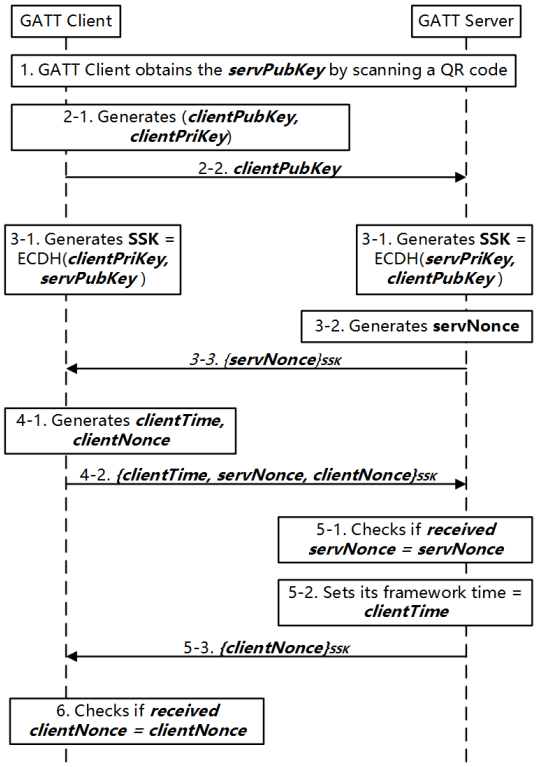

Figure 3: The sequence diagram of the synchronizing time protocol

1: *Device A* → *Device B*: $\{1, data\}_{SK}$

2: *Device B* → *Device A*: $\{2, data\}_{SK}$

3: *Device A* → *Device B*: $\{1, data\}_{SK}$

4: …

n: *Device A* → *Device B*: $\{n, data\}_{SK}$

Figure 4: The protocol of securely transmitting data between two BLE devices

2.1 Implementation of cryptography

In addition to ensuring that the designed security goals are satisfied by the security framework, we try to make the security framework as simple as possible so that it requires less computation and power consumption. AES-128 with CTR mode is used for encrypting and decrypting data. A key length of 160-bit (P-160) is used in the ECDH algorithm. The purpose of this choice is to balance efficiency, security and compatibility, because the versions 4.0 and 4.1 of BLE specification limit the Maximum Transmission Unit (MTU) of GATT to 20 bytes. The RSA algorithm is not suitable because it needs a very long key (length more than 1024 bits) and its calculation is very demanding and needs high power consumption.

2.2 Implementation of the synchronizing time protocol

The synchronizing time protocol is implemented as a GATT service. The service enables the Indicate Notify properties (in Bluetooth specification) for enabling retransmission mechanisms.

2.3 Implementation of the secure transmitting data protocol

Two functions are implemented: SecureSend and SecureRecv. Developers can add these two functions to any other BLE services demand the security of transmitting data via BLE.

3 Performance & power consumption

3.1 Testing environment

Tab. 1 lists the system setting of the tests.

Table 1: Setting of the tests

Hardware	Software
A nRF52832 development board	The security framework (server part)
A smartphone (Android 6.0, BLE 4.1)	The security framework (client part)
An Oscilloscope	N/A

3.2 Performance

The nRF52832 BLE chip provides a Real-Time Counter (RTC), which is increased 32,678 times every second, for timekeeping. In the tests, we record an RTC value before each protocol starts and record an RTC value after each protocol ends. A for loop is used to execute every protocol 10 times and then an average code execution time of every protocol is calculated. For each protocol in the security framework, formula 2 and formula 3 are used to calculate its average code execution time.

$$CodeExecutionTime = (RTC_{after} - RTC_{before})/32768 \quad (2)$$

$$AverageCodeExecutionTime = (\sum_{n=1}^{10} CodeExecutionTime)\ /10 \quad (3)$$

The results of average code execution time of each protocol are shown in Tab. 2.

Table 2: The average code execution time of each protocol

Task	Time (ms)
Synchronizing Time Protocol	0.823959
SecureRecv	3.204285
SecureSend	0.427238

3.3 Power consumption

In the experiments, a 22 Ω resistor is connected in series with the nRF52832 BLE chip in the development board. An oscilloscope is used to measure real time voltages of the resistor. The Ohm's Law formula (Eq. (4)) is used to calculate the real time currents of the nRF52832 BLE chip [Nordic Semiconductor ASA (2017)]. Since we cannot measure the real time voltages of the nRF52832 BLE chip, we cannot use Eq. (5) to calculate the power consumption. Besides, since most wearable devices use *mAh* (milliamp-hour) to represent its battery power, using current is more convenient for calculating power consumption.

$$I = U/R \quad (4)$$

$$P = I * U \quad (5)$$

Because our security framework is a module in application, it does not affect the BLE protocol stack. Therefore, its power consumption is produced by the calculation of cryptographic algorithms and does not add extra power consumption to BLE protocol stack.

In order to measure the power consumption of the calculation of cryptographic algorithms, an infinite for loop running for 5 ms is set to calculate the power consumption without using the security framework. The oscilloscope snapshot of the infinite for loop is shown in Fig. 5. We use Eq. (6) and Eq. (7) as the formulas to calculate the average current consumption of the infinite for loop; it is 4.36 mA.

A timer is set for each protocol. All these timers will be activated every 10 ms. When a timer is activated, it will use a for loop to execute every protocol 10 times and then go to sleep and wait for the next activation.

The real time current values of the nRF52832 BLE chip are measured, and the average current of every protocol is calculated. Figs. 6, 7, and 8 show the oscilloscope snapshots

of each protocol. Among them, Fig. 6 shows the details of its waveform diagram. The timer cycle value is 10 ms. The work mark represents the working period of the nRF52832 chip. The sleep mark represents the sleeping period of the nRF52832 chip. The numbers marked upon the waves correspond to each execution of the protocol. Each division (div) of X axis represents 1ms and each div of Y axis represents 25 mV. The first wave of each waveform diagram represents a mixed voltage of chip's wake up and the first-time execution of a protocol. The rest waves (9 times) of each waveform diagram represent voltage values when each protocol is being executed.

Therefore, we use Eq. (6) and Eq. (7) as the formulas to calculate an average current of the rest waves (9 times) of each waveform diagram.

$$AverageCurrent = WaveVoltage/22 \tag{6}$$

$$AverageCurrent = (\sum_{n=1}^{9} WaveCurrent)\ /10 \tag{7}$$

The results of average current consumption of each protocol are shown in Tab. 3.

The comparison of current consumption is shown in Tab. 4.

From Tab. 4, we can know that the security framework adds extra current consumption about 1.19 mA to 1.37 mA. We can calculate the power consumption (mAh) by using current multiplied by time (which is shown in Tab. 2). As shown by Tab. 5, the extra power consumption is very small, only about $8.47 * 10^{-6}$ to $1.88 * 10^{-5}$.

Table 3: The average current consumption of each protocol

Task	Current (mA)
Sync. Time Protocol	5.73
SecureRecv	5.55
SecureSend	5.55

Table 4: The comparison of the current consumption

Task	Using the framework	Without using the framework	Extra current
Sync. Time Protocol	5.73 mA	4.36 mA	1.37 mA
SecureRecv	5.55 mA	4.36 mA	1.19 mA
SecureSend	5.55 mA	4.36 mA	1.19 mA

For practical applications, the power consumption of the security framework can be ignored because it is very small and will not affect the applications' battery life.

Table 5: The extra cost caused by each task

Task	Extra current (mA)	Extra time (ms)	Extra power consumption (mAh)
Sync. Time Protocol	1.37	0.823959	$1.88137305 * 10^{-5}$
SecureRecv	1.19	3.204285	$6.35516525 * 10^{-5}$

SecureSend	1.19	0.427238	$8.473553667 * 10^{-6}$

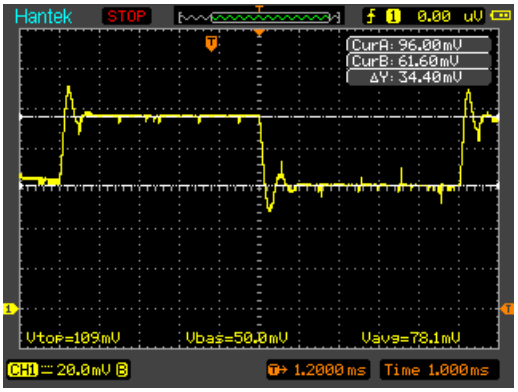

Figure 5: The oscilloscope snapshot of the infinite for loop that without using the security framework

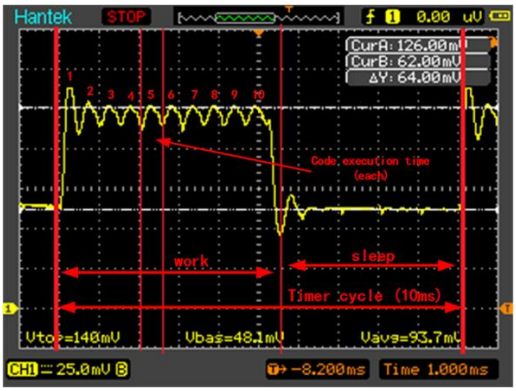

Figure 6: The oscilloscope snapshot of the synchronizing time protocol.

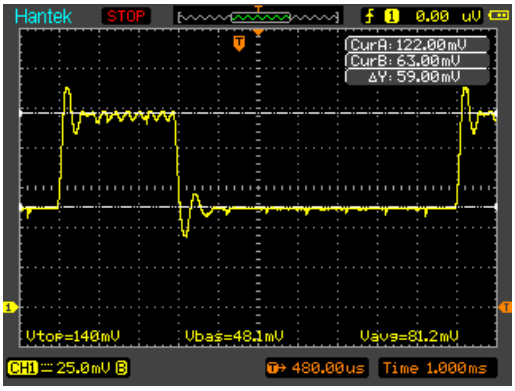

Figure 7: The oscilloscope snapshot of the secure transmitting data function (SecureRecv)

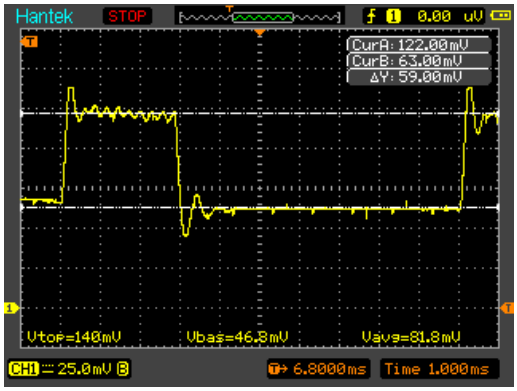

Figure 8: The oscilloscope snapshot of the secure transmitting data function (Secure-Send)

4 Summary

We have designed and implemented a framework for secure communication and usage of BLE devices. This framework has been tested using BLE wearable devices. The protocols of this framework have been formally verified. Since the framework does not assume additional support of hardware, operating system, or existing BLE architecture, it can be seamlessly integrated into the security solutions of IoT of various scenarios.

In the future we plan to apply the framework to more devices and develop applications based on this framework with more convenient user interface. It is possible to utilize some advanced method to securely transfer the public key of a GATT server, such as some smart key management scheme when more resources or options are allowed, in addition to the current way of scanning a QR code.

References

Arora, M. (2012): How secure is aes against brute force attacks? http://www.eetimes.com/document.asp?doc_id=1279619.

Blanchet, B. (2016): Modeling and verifying security protocols with the applied pi calculus and proverif. *Foundations and Trends in Privacy and Security*, vol. 1, no. 1-2, pp. 1-135.

Bluetooth Special Interest Group (2016): Security features in blue tooth low energy. https://www.bluetooth.com/~/media/files/specification/bluetooth-low-energy-security.ashx?la=en.

Bluetooth Special Interest Group (2017): Bluetooth core specification. https://www.bluetooth.com/specifications/bluetooth-core-specification.

Chang, R.; Shmatikov, V. (2007): Formal analysis of authentication in bluetooth device pairing. *(FCS-ARSPA) Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis*, pp. 45-62.

Gong, T.; Huang, H.; Li, P.; Zhang, K.; Jiang, H. (2015): A medical healthcare system for privacy protection based on iot. *Seventh International Symposium on Parallel Architectures, Algorithms and Programming*, pp. 217-222.

Guttman, J. D. (2002): Security protocol design via authentication tests. *Proceedings of 15th IEEE Computer Security Foundations Workshop*, pp. 92-103.

Li, M.; Yu, S.; Zheng, Y.; Ren, K.; Lou, W. (2013): Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption. *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131-143.

Liang, Z.; Verma, R. M. (2008): Improving techniques for proving undecidability of checking cryptographic protocols. *Third International Conference on Availability, Security and Reliability*, pp. 1067-1074.

Liang, Z.; Verma, R. M. (2009): Correcting and improving the np proof for cryptographic protocol insecurity. *Information Systems Security*, pp. 101-116.

Lowe, G. (1995): An attack on the needham-schroeder public-key authentication protocol. *Information Processing Letters*, vol. 56, no. 3, pp. 131-133.

Nick (2015): The hitchhikers guide to ibeacon hardware: a comprehensive report by Aisle-labs. http://www.aislelabs.com/reports/beacon-guide/.

Nordic Semiconductor ASA (2017): nRF51 development kit user guide v1.2. http://infocenter.nordicsemi.com/pdf/nRF52_DK_User_Guide_v1.2.pdf.

Padgette, J.; Bahr, J.; Batra, M.; Holtmann, M.; Smithbey, R. et al. (2017): Guide to bluetooth security: NIST special publication 800-121 revision 2. https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-121r2.pdf.

Qu, Y.; Chan, P. (2016): Assessing vulnerabilities in bluetooth low energy (BLE) wireless network based IOT systems. *IEEE 2nd International Conference on Big Data Se- curity on Cloud, IEEE International Conference on High Performance and Smart Computing, and IEEE International Conference on Intelligent Data and Security*, pp. 42-48.

Ryan, M. (2013): Bluetooth: with low energy comes low security. *7th USENIX Work-Shop on Offensive Technologies.*

Sun, W.; Cai, Z.; Li, Y.; Liu, F.; Fang, S. et al. (2018): Security and privacy in the medical internet of things: a review. *Security and Communication Networks*, vol. 2018, no. 1, pp. 9.

Wikipedia (2017): Elliptic curve diffie-hellman. https://en.wikipedia.org/wiki/Elliptic-curve_Diffie%E2%80%93Hellman.

Zhang, Q.; Liang, Z. (2017): Security analysis of bluetooth low energy based smart wristbands. *2nd International Conference on Frontiers of Sensors Technologies*, pp. 421-425.