RESEARCH ARTICLE OPEN ACCESS

Dual Authentication For Bluetooth Connection

Diallo Alhassane Saliou[1], Wajdi Fawzi Mohammed Al-Khateeb[1] Rashidah Funke Olanrewaju[1] and Sado Fatai[2]
[1](Department of Electrical and Computer Engineering, Kulliyyah of Engineering, International Islamic University Malaysia, P.O. Box 10, 50728 Kuala Lumpur, MALAYSIA)
[2](Department of Mechatronics Engineering, Kulliyyah of Engineering, International Islamic University Malaysia, P.O. Box 10, 50728 Kuala Lumpur, MALAYSIA)

ABSTRACT
Recently, Bluetooth technology is widely used by organizations and individuals to provide wireless personal area network (WPAN). This is because the radio frequency (RF) waves can easily penetrate obstacles and can propagate without direct line-of-sight (LoS). These two characteristics have led to replace wired communication by wireless systems. However, there are serious security challenges associated with wireless communication systems because they are easier to eavesdrop, disrupt and jam than the wired systems. Bluetooth technology started with a form of pairing called legacy pairing prior to any communication. However, due to the serious security issues found in the legacy pairing, a secure and simple pairing called SPP was announced with Bluetooth 2.1 and later since 2007. SPP has solved the main security issue which is the weaknesses of the PIN code in the legacy pairing, however it has been found with some vulnerabilities such as eavesdropping and man-in-the-middle (MITM) attacks. Since the discovery of these vulnerabilities, some enhancements have been proposed to the Bluetooth Specification Interest Group (SIG) which is the regulatory body of Bluetooth technology; nevertheless, some proposed enhancements are ineffective or are not yet implemented by Manufacturers. Therefore, an improvement of the security authentication in Bluetooth connection is highly required to overcome the existing drawbacks. This proposed protocol uses Hash-based Message Authentication Code (HMAC) algorithm with Secure Hash Algorithm (SHA-256). The implementation of this proposal is based on the Arduino Integrated Development Environment (IDE) as software and a Bluetooth (BT) Shield connected to an Arduino Uno R3 boards as hardware. The result was verified on a Graphical User Interface (GUI) built in Microsoft Visual Studio 2010 with C sharp as default environment. It has shown that the proposed scheme works perfectly with the used hardware and software. In addition, the protocol thwarts the passive and active eavesdropping attacks which exist during SSP. These attacks are defeated by avoiding the exchange of passwords and public keys in plain text between the Master and the Slave. Therefore, this protocol is expected to be implemented by the SIG to enhance the security in Bluetooth connection.
Keywords- Authentication, Bluetooth Security, HMAC Algorithm, Legacy Pairing, Secure and Simple Pairing.

I. INTRODUCTION

A wireless personal area network (WPAN) "Fig. 1," is a short-distance wireless network specially designed to support portable and mobile computing devices such as personal computer (PC), personal digital assistants (PDA), cell phones, printers, pagers, storage devices, and a variety of consumer electronic equipments [1]. Bluetooth technology which was developed to replace the existing wire line connections is used in WPAN with short-range interconnectivity. Moreover, Bluetooth radio operates in the license-free and globally available Industrial, Scientific, and Medical (ISM) band at 2.4 GHz [2] using Frequency-Hopping Spread Spectrum (FHSS) and are capable of transmitting voice and data [3]. Bluetooth provides enough bandwidth that enables data exchange between several mobile devices at a rate up to 1 Mbps [1] for version 2.0 (and earlier) and up to 3 Mbps for version 2.1 (and later) [4][5]. Bluetooth standard is designed for downward compatibility which means that the latest versions can support all features available in old versions. In Bluetooth connection, a piconet is a small network created on an ad hoc basis that includes one master device and up to seven slaves while a scatternet is chain of piconets that allows one or more Bluetooth devices to be a slave in one piconet and act as the master for another piconet, simultaneously [4].

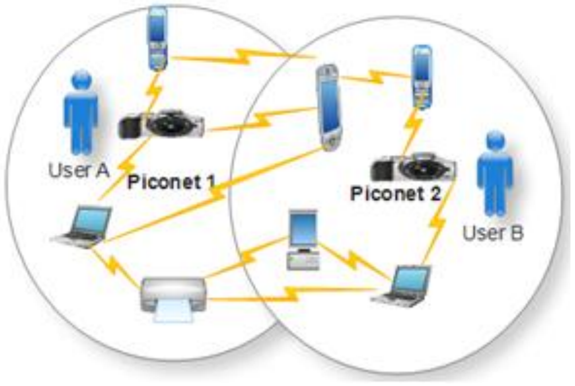


Fig. 1. Introduction to Bluetooth connection

In order for two Bluetooth devices to communicate within a piconet, they need to perform a mutual authentication. During the mutual authentication, pairing is performed in order to establish the connection.

The rest of the paper is organized as follows: section II discusses the existing pairing methods and the corresponding limitations. Section III provides related works on the actual pairing methods. The proposed authentication scheme is described in section IV followed by an implementation result in section V. Section VI gives the discussion and a conclusion is shown in section VII.

II. EXISTING PAIRING METHODS AND THEIR LIMITATIONS

2.1 Legacy Pairing

The legacy pairing is vulnerable to different security issues such as weak PIN code, passive and active eavesdropping. This method of pairing requires each device to enter a Personal Identity Number (PIN) code in order to perform pairing. Pairing is successful only if both devices enter the same code. In [6], it is stated that many Bluetooth devices today use a 4-digits PIN or a fixed PIN of commonly known values which significantly limit the security of the link key. Therefore, during the pairing procedure there is a very high probability for an attacker to get the used PIN as in [7] [8] [9].

2.2 Secure and Simple Pairing (SSP)

In SSP the pairing process is enhanced and became simple and more secure due to the non-use of a fixed PIN code. However, several attacks have been reported recently on its four pairing methods:

Attacks on the Just Work Model: A Bluetooth non-input non-out man-in-the middle attack (BT-NINO-MITM) in the just work model was identified in [12] and implemented in [11]. In [13], it is published a novel Bluetooth MITM attack called BT-SPP-Printer-MITM attack against the just work model. Besides that, an attack called Bluetooth-Secure and Simple Pairing- Headset/Hands-Free-Man in the middle attack (BT-SSP-HS/HF-MITM) was proposed in [14]. To perform the attack, the authors exploited the fact that most Headset/Hands-Free Bluetooth devices can be forced to choose the less secure just work model[12][13].

Attacks on the Passkey Entry Model: The possibility of successful eavesdropping and MITM attack on the passkey entry model has been mentioned in [10] and the implementation in the GNU radio software framework using the universal software radio peripheral (USRP) as hardware can be found in [5].

Attacks on the Out of Band Model: In [14], it is published a Bluetooth-Secure and Simple Pairing-Out of Band-Man in the middle attack (BT-SSP-OOB-MITM) and it is shown that the attack can be performed if the attacker succeeds to have visual contact to the legitimate user's device. The OOB model was suggested to be used as a mandatory model as in [12] [15]; nevertheless, in [11] it is also mentioned that this proposal cannot work.

Case of the Numerical Comparison Model: This model is not directly attacked; however attackers can force legitimate users to select a less secure model instead of this secure model. For this reason, in [15] it is mentioned that the numerical comparison model is also found to be not secure.

III. RELATED WORKS

3.1 Legacy Pairing

In [16], it was suggested an enhancement of Bluetooth authentication using the concatenation of a master's Clock and a Low Address Part (LAP) to be xored with the least 42 bits of the Authentication Random Number (AU_RAND) before being fed into the E_1 algorithm, where the signed response (SRES) is computed. However, this improvement has its drawbacks since it relies on a symmetric key which is not securely shared.

Moreover, in [17] it was designed an improved authentication algorithm using the concatenation of a clock and a part of address values (PAV) to compute the authentication random number: AU_RAND' = f (AU_RAND, Clock, PAV). However, AU_RAND which is a public parameter does not need to be changed because this does not prevent to guess the PIN code. Therefore, this enhancement is ineffective.

Reference [18] explored the weakness of the PIN and proposed to add a parameter called authentication ID (au_id) which is 128 bits in the generation of the initialization key. This au_id is shared by using Diffie-Hellman key exchange and makes the PIN more robust: PIN'=PIN U au_id. However, this current approach remains weak due to the use of the unit key and its non-implementation to assess its performance. The Diffie-Hellman key exchange algorithm used is also prone to the MITM attack.

3.2 Secure and Simple Pairing

Reference [19] developed an improved authentication algorithm by using SSP which employs Elliptive Curve Cryptography (ECC) that is an analog of Diffie-Hellman Key Exchange. However, one of the weakness of the ECC is that if all ECC users agree on a common set of Elliptive Curve (EC) parameters, to negotiate these parameters, the additional information needed to specify the exact EC might make the effective EC key size to become very large. Another drawback of the ECC is that it increases the size of the encrypted message more than the Rivest-Shamir-Adleman (RSA) encryption. It is also mathematically subtle and more difficult to implement than the RSA.

In [13] it is proposed to add a message saying "The second message has no display and keyboard! Is this true?" in the just work model to solve the BT-NINO-MITM attack. After displaying the message, the user may choose "Proceed" or "Stop". However, it is shown that this proposal does not solve the attack as in [11].

In the view of the above, the authentication procedure in Bluetooth connection needs to be improved.

IV. PROPOSED AUTHENTICATION METHOD

The proposed model employs a dual authentication which is an authentication concept that requires two verifications prior to establishing any communication.

4.1 Description

First of all, a master device is nominated and all Bluetooth devices in a piconet are registered into the database of the master device by assigning a password and a public key to each device such that the password and the public key match the identity (ID) of the device as in "Table I.". This process of registration and updating the database is executed by the administrator of the WPAN. "Table II." describes all involved security entities in the proposed model.

TABLE I. DATABASE OF THE MASTER DEVICE

No	Identities	Passwords	Public Keys
1	ID_A	Pwd_A	KU_A
2	ID_B	Pwd_B	KU_B
3	ID_C	Pwd_C	KU_C
4	ID_D	Pwd_D	KU_D
5	ID_E	Pwd_E	KU_E
6	ID_F	Pwd_F	KU_F
7	ID_G	Pwd_G	KU_G

TABLE II. DEFINITION OF INVOLVED PARAMETERS AND SYMBOLS

No	Parameters	Description	Size	Status
1	ID = BD_ADDR	Identity = Bluetooth device address	48 bits	public
2	Pwd_A	Password of slave A	128 bits	private
3	Kc	Secret key derived from PwdA	128 bits	private
4	IV	Initial Value	128 bits	private
5	KU_A or KU_a	Slave public key	128 bits	public
6	KR_A or KR_a	Slave private key	128 bits	private
7	Ks	Session Key for AES	128 bits	private
8	K^+	HMAC secret key	256 bits	private
9	M	HMAC Authentication message	512 bits	private
10	HMAC	Authentication Algorithm HMAC		
11	C_A & C_m	Slave and master commitments values	256 bits	public
12	‖, E, D	Concatenation, encryption, and decryption Symbols		

Secondly, we list all initial parameters possessed by both devices:
Slave A: (ID_A, Pwd_A, KU_A, KR_A,).
Master: (ID_A… ID_G, Pwd_A… Pwd_G, KU_A…KU_G).

4.2 Different Phases of the Proposed Authentication Scheme

1) First Authentication Stage (Phase 1): This first phase consists of three messages between the master and the slave. It will result to a first verification called first authentication stage or handshaking.
Message 1: A slave which would like to establish a secure communication with the master device sends its ID to the master.
Message 2: The master receives the ID and checks its database to see whether the received ID exists in the data base or not. If it exists, the master will derive a secret key (Kc) from the corresponding password of the current ID. However, if the ID is not registered previously in the database, it means that none of the seven devices of the piconet has sent its ID. Therefore, the master will ignore the sent ID.

Assuming that the ID exists in the database, the master generates randomly a session key (Ks) and derive a secret key (Kc) from the stored password. The derivation of (Kc) is executed as follows:

- If the password length is less than 16 bytes, zero padding is applied to the left most significant bits in order to get a key size of 16 bytes.
- If the password length is exactly 16 bytes, it is used directly as a key without any modification.
- If the password length is greater than 16 bytes, Fanfold operation is applied to get 16 bytes.

A double encryption of the MAC address of the slave (ID_A) will be executed using the AES encryption with Cipher Block Chaining (CBC) mode which is a recommended mode due to its security. The first encryption is done by using the session key (Ks) and the second encryption is done by using the derived secret key (Kc). The master will send to the slave the double encryption (Cipher 2) with the concatenation of the encryption of the initial value (IV) (1).

$E_{Kc}\,[E_{Ks}\,(ID_A)]\ \|\ E_{KUa}\,(IV) = E_{Kc}\,[\text{Cipher 1}]\ \|\ E_{KUa}\,(IV) = \text{Cipher 2}\ \|\ E_{KUa}\,(IV)$ (1)

The purpose of the double encryption is to identify the slave by asking the user to recover cipher 1 as well as to avoid the following attack: assuming that an attacker has eavesdropped on the wireless connection and captured the cipher sent in Message 2, he cannot recover neither cipher 1 nor the IV because only the legitimate user knows the password from which Kc is derived and the private key to decrypt the encrypted IV. Upon receiving cipher 2 with the concatenation of the encrypted (IV), the user recovers the IV by using his private key as in (2).

$D_{KRa}\,(IV) = IV$ (2)

After that, the system prompts him to enter his password and derive the same secret key Kc in order to decrypt cipher 2 as in (3).

$D_{Kc}\,(\text{Cipher 2}) = D_{Kc}\,\{E_{Kc}\,[E_{Ks}\,(ID_A)]\} = \text{Cipher 1}$ (3)

If cipher 1 is successfully recovered, it means that the correct password is supplied. Failure to provide the correct password will result to abort the connection.

Message 3: The user returns to the master cipher 1 together with his device ID as in (4). Here the master will decrypt cipher 1 by using (Ks) and get the encrypted ID. After that, the master will compare the original ID and the sent ID in Message 3. If they match, it means that Message 2 was not altered and Message 3 is sent by the exact slave. Otherwise, the connection will be aborted. This is the end of the first authentication stage.

$ID_A\ \|\ \text{Cipher 1} = ID_A\ \|\ E_{Ks}\,(ID_A)$ (4)

2) Exchange of Secret Key (Phase 2): This phase consists of one message.

Message 1: For the purpose of security, it is recommended to generate a new secret key instead of re-using the same Ks. Thus, the master generates a new session key noted by K^+ to compute the HMAC algorithm. The master will transfer the concatenation of K^+ and a message M by using RSA key exchange which is a secure method to transfer the session key as in (5). With RSA key exchange, secret keys are exchanges securely by encrypting them with the public key of the intended recipient. Only the recipient can decrypt the encrypted key because it requires using his own private key. Indeed, a third party who intercepts the encrypted secret key. Thus, secrecy and privacy of the session key is well obtained. Furthermore, the integrity of the encrypted key is accomplished since there is no way to tamper the transferred key. RSA algorithm is demonstrated to be reliable with high quality, guaranteed security and strong encryption. The selection of RSA key exchange is also motivated by its simplicity on hardware implementation.

$E_{KUa}\,[K+\ \|\ M] = \text{Cipher 3}$ (5)

The slave decrypts cipher 3 using the private key as in (6):

$D_{KRa}\,[K^+\ \|\ M] = K^+\ \|\ M$ (6)

3) Second Authentication Stage (Phase 3): This phase also consists of one message.

Message 1: The slave computes the HMAC algorithm to get a commitment value (C_A) to be sent back to the master as in (7). Meanwhile the master computes C_m as in (7). The master device compares C_A and C_m. If the two values match, it means that both the session key (K^+) and the message (M) are not altered. However, failure to that requires aborting the connection. HMAC is used as the authentication algorithm because it can verify data integrity and authentication simultaneously. HMAC is employed with the hash function SHA256 thus the name HMAC-SHA256.

$C_A = C_m = HMAC\,[K^+\ \|\ M]$ (7)

Upon receiving C_A, the Master compares C_A to Cm and decides whether authentication is successful or not.

4) Exchange of Encrypted Data (Phase 4): This last phase consists of many messages depending on the data to share. Devices can exchange data securely by using a new generate secret key for encryption. The communication can be ended by either of the devices. However, in this case the slave will end the connection when it will finish exchanging data because it initiated the connection for a specific purpose. It should be noted that the session key is a temporary key; hence for the purpose of security it needs to be changed periodically. "Table III."summarizes the phases of this proposal and "Fig. 2," represents the block diagram of the model.

TABLE III. SUMMARY STEPS OF THE PROPOSED MODEL

PHASES	**MESSAGES**	**BT Devices**
First Authentication Stage	M_1: ID_A	Slave to Master
	M_2:$E_{Kc}[E_{Ks}(ID_A)]\|\| E_{KUa}$ (IV)	Mater to Slave
	M_3:$ID_A\|\| E_{Ks}$ (ID_A)	Slave to Master
Exchange of Secrete Key	M_1: E_{KUa} [$K^+ \|\|$ M]	Mater to Slave
Second Authentication Stage	M_1: C_A = HMAC [K^+ || M]	Slave to Master
	Meanwhile Master computes: C_m = HMAC [K^+ || M]	
	Master compares C_A to C_m and makes decision	
Exchange of Encrypted Data	If authentication is successful devices exchange data securely.	Slave to Master and Mater to Slave

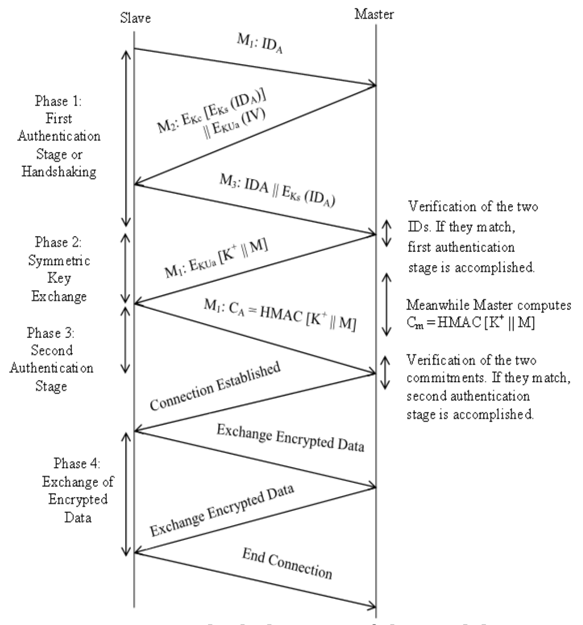

Slave
Master
M_1: ID_A
Phase 1: First Authentication Stage or Handshaking
M_2: E_{Kc} [E_{Ks} (ID_A)] || E_{KUa} (IV)
M_3: IDA || E_{Ks} (ID_A)
Verification of the two IDs. If they match, first authentication stage is accomplished.
Phase 2: Symmetric Key Exchange
M_1: E_{KUa} [K^+ || M]
Meanwhile Master computes C_m = HMAC [K^+ || M]
Phase 3: Second Authentication Stage
M_1: C_A = HMAC [K^+ || M]
Verification of the two commitments. If they match, second authentication stage is accomplished.
Connection Established
Exchange Encrypted Data
Phase 4: Exchange of Encrypted Data
Exchange Encrypted Data
End Connection

Figure 2: Block diagram of the model

V. IMPLEMENTATION RESULTS

Security consideration: For a correct and good implementation, the following security concerns need to be taken into consideration:

- Generation of secret random keys by using a strong random number generator (RNG);
- Strength of the keys;
- Changing the keys periodically;
- Secure protection of keys;
- Secure key exchange mechanism;
- Correctness of the used algorithm.

For the implementation on the mentioned hardware and software, the following three libraries have been imported: AES-CBC library, RSA library and HMAC-SHA256 library. The Arduino board and the Bluetooth shield were connected together prior to connecting the board to a laptop running Arduino IDE via a USB A to B cable. The code was developed on the Arduino IDE then linked to a GUI created in Visual studio C sharp environment.

Designed GUI: The GUI contains one main form, seven (7) windows for the First Authentication Stage (FAS1 to FAS7) and five (5) windows for the Second Authentication Stage (SAS1 to SAS5).

Running the Source Code on Arduino IDE: The source code developed on the Arduino IDE was first compiled and uploaded to the board. After the display of the message "done uploading", the code was run on the serial monitor of the IDE in order to see the output. Upon successfully running the code source, the GUI was debugged to start the authentication procedure. When the debugging was started, the main form of the GUI appeared. The main form is described as follows:

Main Form of the GUI: The main form of the GUI named "MainForm" as shown in "Fig. 3" displays the following characteristics:

COM Port or Communication Port: This label shows the port at which the Arduino board is connected. It can be port COM1, COM2, COM3, etc. The Arduino IDE displays the used port in the menu "Tools". Thus in the GUI the user will just select the exact port at which the board is connected.

Baud Rate: This parameter indicates the rate in baud (or bits per second) at which data is transferred between the board and the laptop via the serial port. The Arduino board has a serial port known as UART port that communicates with the computer via the USA cable.

Button Connect: After selecting the exact COM port and the speed at which data is transferred and clicking the button **"connect"**, the main form displays in the ride side the connected devices with corresponding characteristics. It can be seen the following:

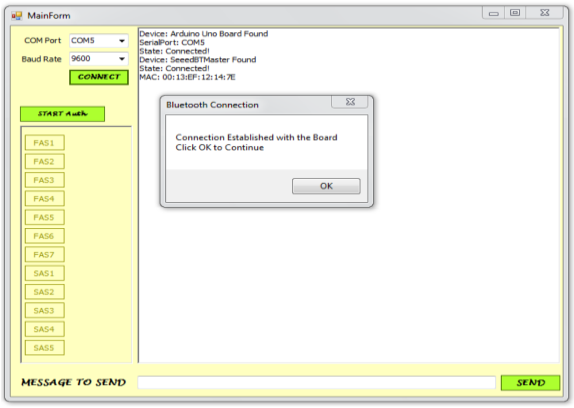
- **Device:** Arduino Uno Board Found
- **Serial Port:** COM5

- **State:** Connected!
- **Device:** SeeedBTMaster Found
- **State:** Connected!
- **MAC:** 00:13:EF:12:14:7E

In addition to that, a small window named **"Bluetooth Connection"** is displayed to show the following message **"Connection Established with the Board. Click OK to Continue"**. The user needs to click the button **"OK"** to proceed to the following step.

Button start auth: This button is used to start the authentication procedure. After clicking on it, the first window of the FAS appears and the label FAS1 which is displayed in white color shows that the procedure is at the first window. Similarly other labels FAS2 to SAS5 will be displayed in white color when the user clicks the button **"Next"** which is used to proceed to the next window.

Message to send: This label displays any message to be sent the board. The button **"Send"** executes the actual operation.



```
Device: Arduino Uno Board Found
SerialPort: COM5
State: Connected!
Device: SeeedBTMaster Found
State: Connected!
MAC: 00:13:EF:12:14:7E
```

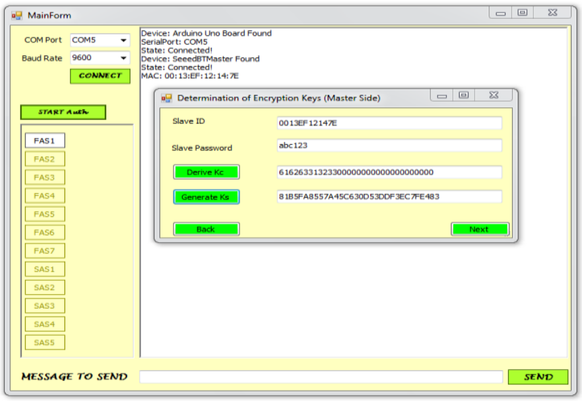
Connection Established with the Board
Click OK to Continue

Figure 3: Main form of the GUI

First Windows of the FAS (FAS1) Named Determination of Encryption Keys (Master Side): This window is displayed as soon as the user clicks the button **"star auth"**. It contains two labels which display the ID of the slave and the stored password. In addition to that, it has the following buttons:

Button Derive Kc: This button is used to derive the encryption key (Kc) from the stored password as shown in "Fig. 4". Any stored password is converted to 128 bits ASCII for the second AES encryption.
Button Generate Ks: This button is used to generate a random secret key of 16 bytes in hexadecimal (128 bits) for the first AES encryption.
Button Next: This button is used to proceed to the next window. By clicking on it, the generated parameters will be displayed on the **"MainForm"** and the following window will directly receive the necessary parameters for the actual operation.

Button Back: This button is used to come back to the previous window. Here the previous window is the **"MainForm"**.



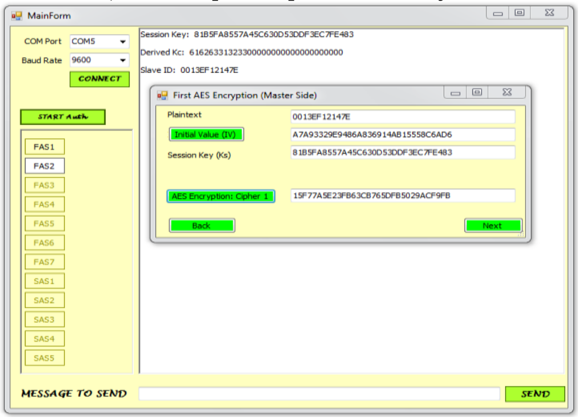
```
Device: Arduino Uno Board Found
SerialPort: COM5
State: Connected!
Device: SeeedBTMaster Found
State: Connected!
MAC: 00:13:EF:12:14:7E
```

Slave ID: 0013EF12147E
Slave Password: abc123
Derive Kc: 61626331323300000000000000000000
Generate Ks: 81B5FA8557A45C630D53DDF3EC7FE483

Figure 4: Determination of encryption keys

Second Windows of the FAS (FAS2) Named First AES Encryption (Master Side): After generating the keys in the first window and clicking the button **"Next"**, this second window appears and displays the following labels and buttons as shown in "Fig. 5":

Label plaintext: This label displays the input data to encrypt (the slave's ID).
Button Initial Value (IV): This button is used to generate randomly an IV of 16 bytes for the CBC mode encryption.
Label Session Key (Ks): This label displays the generated secret key for AES encryption.
Button AES Encryption: By clicking this button the first AES encryption is performed. The AES library with CBC mode takes the three (3) inputs (plain text, IV and Ks) then outputs Cipher 1 of 16 bytes.



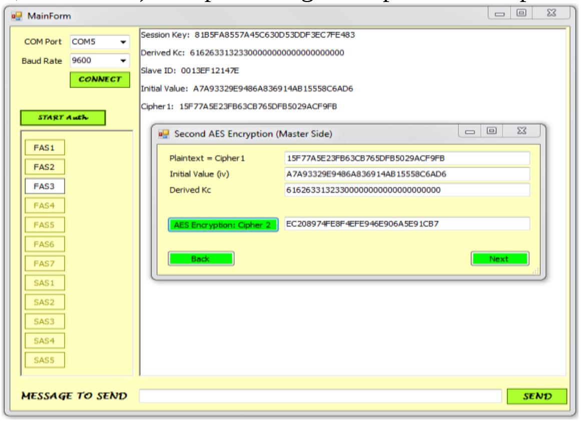
```
Session Key:  81B5FA8557A45C630D53DDF3EC7FE483
Derived Kc:  61626331323300000000000000000000
Slave ID:  0013EF12147E
```

Plaintext: 0013EF12147E
Initial Value (IV): A7A93329E9486A836914AB15558C6AD6
Session Key (Ks): 81B5FA8557A45C630D53DDF3EC7FE483
AES Encryption: Cipher 1: 15F77A5E23FB63CB765DFB5029ACF9FB

Figure 5: First AES encryption master side

Third Windows of the FAS (FAS3) Named Second AES Encryption (Master Side): By clicking the **"Next"** button in the previous window, this third window appears and takes cipher 1 as the current plaintext for encryption. Moreover, the other necessary parameters which are the IV and Kc are displayed as shown in "Fig. 6". A click on the button **"AES Encryption"** produces Cipher 2 which is also

16 bytes. It means that the AES library takes (Cipher 1, IV and Kc) as inputs and gives cipher 2 as output.

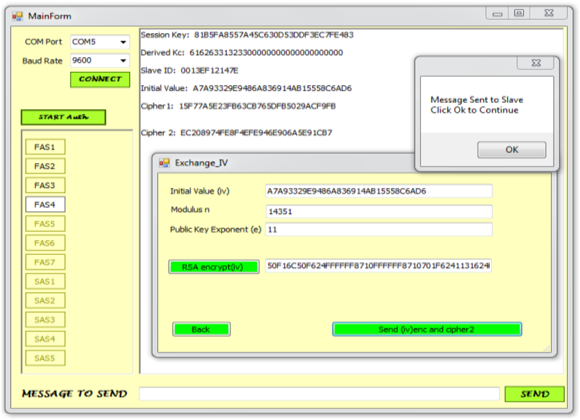


COM Port COM5
Baud Rate 9600
Session Key: 81B5FA8557A45C630D53DDF3EC7FE483
Derived Kc: 61626331323300000000000000000000
Slave ID: 0013EF12147E
Initial Value: A7A93329E9486A836914AB15558C6AD6
Cipher 1: 15F77A5E23FB63CB765DFB5029ACF9FB

Plaintext = Cipher 1 15F77A5E23FB63CB765DFB5029ACF9FB
Initial Value (iv) A7A93329E9486A836914AB15558C6AD6
Derived Kc 61626331323300000000000000000000
AES Encryption: Cipher 2 EC208974FE8F4EFE946E906A5E91CB7

Figure 6: Second AES encryption master side

Fourth Windows of the FAS (FAS4) Named Exchange IV: The objective of this window is to share the IV between the Master and the Slave. To do that, the window displays the IV and the RSA parameters for encryption as shown in "Fig. 7". The window shows the modus n and the public exponent e which are 14351 and 11 respectively. The **"RSA encrypt (IV)"** button takes the IV, n and e as input and output a cipher to be sent to the slave. It should be noted that the imported RSA library provides a cipher with a chain of variable "FFFFF" in length. The cipher itself is the hexadecimal values which appear differently to the chain "FFFFF". The button **"Send encrypted (IV) and cipher 2"** is used to send the concatenation of the encrypted (IV) and cipher 2 to the slave. Upon clicking this button the following message appears in a small window **"Message sent to the slave. Click OK to continue"**.



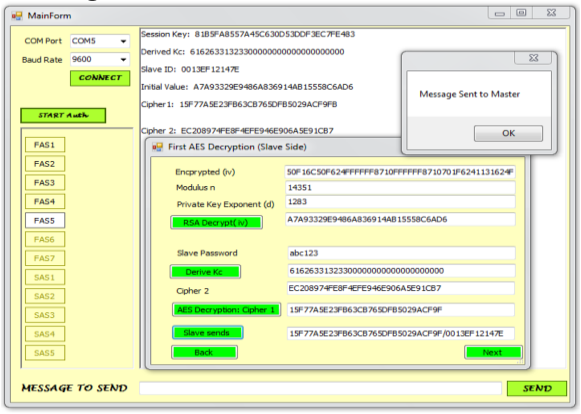
COM Port COM5
Baud Rate 9600
Session Key: 81B5FA8557A45C630D53DDF3EC7FE483
Derived Kc: 61626331323300000000000000000000
Slave ID: 0013EF12147E
Initial Value: A7A93329E9486A836914AB15558C6AD6
Cipher 1: 15F77A5E23FB63CB765DFB5029ACF9FB
Cipher 2: EC208974FE8F4EFE946E906A5E91CB7

Message Sent to Slave
Click Ok to Continue

Initial Value (iv) A7A93329E9486A836914AB15558C6AD6
Modulus n 14351
Public Key Exponent (e) 11
RSA encrypt(iv) 50F16C50F624FFFFFF8710FFFFFF871070 1F6241131624

Figure 7: Exchange of initial value

Fifth Windows of the FAS (FAS5) Named First AES Decryption (Slave Side): After clicking **"OK"** in the previous small window, the actual window is displayed in the slave side. The slave needs to recover the received encryptions. To recover the encrypted IV, the modulus n and the private exponent d are displayed. Then the user clicks the button **"RSA Decrypt (IV)"** to get back the IV as shown in "Fig. 8". Here, the RSA library takes (encrypted IV, n and d) as inputs and output the IV. Next, the slave derives the same Kc (as done in the Master side) from the stored password in the label **"Slave Password"** by just clicking on the button **"Derive Kc"**. The label cipher 2 shows the actual cipher to be decrypted. Upon recovering the IV and deriving Kc the button **"AES Decryption"** is used to recover cipher 1. Here, the AES library takes (Cipher 2, IV and Kc) as inputs and recover cipher 1. After recovering cipher 1, the slave sends back to the master the concatenation of cipher 1 and the ID. This concatenation is displayed in front of the button **"Slave sends"**. By clicking this button, a small window is displayed to show the following message **"Message Sent to Master"**.



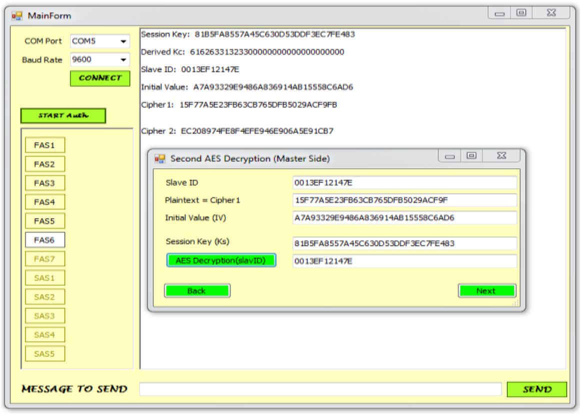
COM Port COM5
Baud Rate 9600
Session Key: 81B5FA8557A45C630D53DDF3EC7FE483
Derived Kc: 61626331323300000000000000000000
Slave ID: 0013EF12147E
Initial Value: A7A93329E9486A836914AB15558C6AD6
Cipher 1: 15F77A5E23FB63CB765DFB5029ACF9FB
Cipher 2: EC208974FE8F4EFE946E906A5E91CB7

Message Sent to Master

Encrypted (iv) 50F16C50F624FFFFFF8710FFFFFF871070 1F6241131624F
Modulus n 14351
Private Key Exponent (d) 1283
RSA Decrypt(iv) A7A93329E9486A836914AB15558C6AD6
Slave Password abc123
Derive Kc 61626331323300000000000000000000
Cipher 2 EC208974FE8F4EFE946E906A5E91CB7
AES Decryption: Cipher 1 15F77A5E23FB63CB765DFB5029ACF9F
Slave sends 15F77A5E23FB63CB765DFB5029ACF9F/0013EF12147E

Figure 8: First AES decryption slave side

Sixth Windows of the FAS (FAS6) Named Second AES Decryption (Master Side): A click on the button **"Next"** of the previous window displays this actual window in the master side where the second AES decryption is performed. FAS6 displays the received concatenation (slave's ID and cipher 1), the IV and the secret key (Ks) as shown in "Fig. 9". By clicking the button **"AES Decrypt (Slave ID)"**, the AES library takes (cipher 1, IV and Ks) to recover the ID of the slave. At, this point the master has the two IDs for verification.

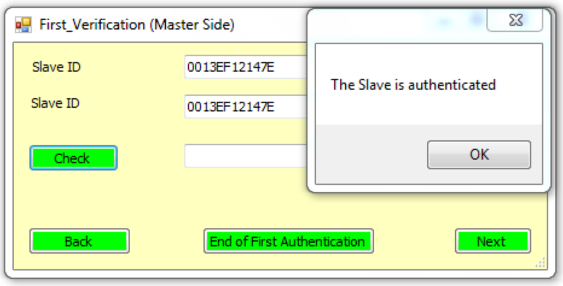


COM Port COM5
Baud Rate 9600
Session Key: 81B5FA8557A45C630D53DDF3EC7FE483
Derived Kc: 61626331323300000000000000000000
Slave ID: 0013EF12147E
Initial Value: A7A93329E9486A836914AB15558C6AD6
Cipher 1: 15F77A5E23FB63CB765DFB5029ACF9FB
Cipher 2: EC208974FE8F4EFE946E906A5E91CB7

Slave ID 0013EF12147E
Plaintext = Cipher 1 15F77A5E23FB63CB765DFB5029ACF9F
Initial Value (IV) A7A93329E9486A836914AB15558C6AD6
Session Key (Ks) 81B5FA8557A45C630D53DDF3EC7FE483
AES Decryption(slavID) 0013EF12147E

Figure 9: Second AES decryption master side

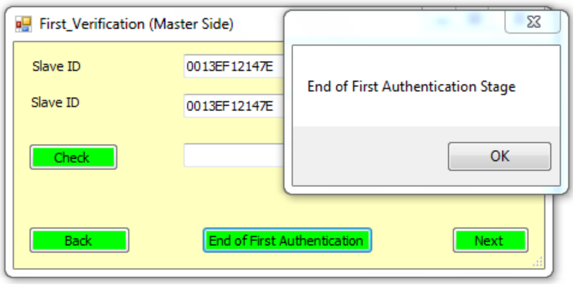
Seventh Windows of the FAS (FAS7) Named First Verification (Master Side): A click on the **“Next”** button of the previous window displays the two IDs to be compared. Here, the button **“Check”** is used to verify if the two IDs match. Upon clicking on it, either of the following messages is displayed in a small window: **“The slave is authenticated”** if the IDs match or **“The slave is unauthenticated”** if they are different as shown in “Fig. 10a)”. It should be noted that the connection will be aborted in case of unsuccessful test. The button **“End of First Authentication”** displays in a small window the message **“End of First Authentication”** which means that the FAS is done as shown in “Fig. 10b)”. By clicking the button **“OK”** in this small window, then the button **“Next”** the message **“Second Authentication Stage Click to Begin”** appears as shown in “Fig. 10c)”. After clicking **“OK”**, the first window of the SAS appears.


First_Verification (Master Side)

Slave ID: 0013EF12147E
Slave ID: 0013EF12147E

The Slave is authenticated

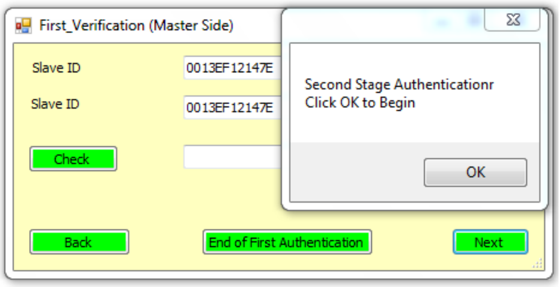
Figure 10 a): First verification master side


First_Verification (Master Side)

Slave ID: 0013EF12147E
Slave ID: 0013EF12147E

End of First Authentication Stage

Figure 10 b): First verification master side

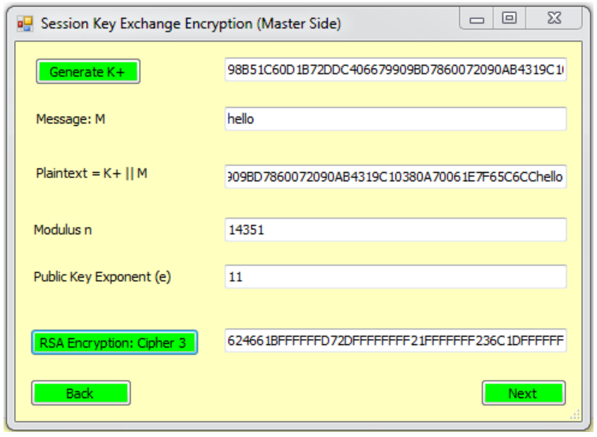

First_Verification (Master Side)

Slave ID: 0013EF12147E
Slave ID: 0013EF12147E

Second Stage Authenticationr
Click OK to Begin

Figure 10 c): First verification master side

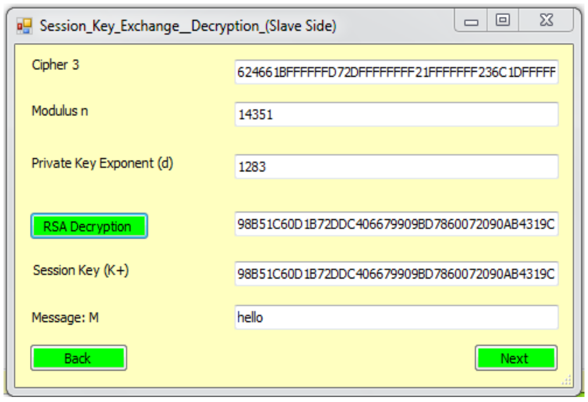
First Windows of the SAS (SAS1) Named Session Key Exchange (Master Side): This window allows the Master to generate randomly K^+ and to input a message M then share the concatenation of K^+ and M with the slave by using the RSA library. K^+ of 256 bits is generated by clicking on the button **“Generate K^+”**. The label **“Message: M”** shows the entered message M. Due to the limitation (in term of size) of the RSA library, a small message of maximum 16 bytes in hexadecimal can be entered. This is because the RSA library takes only a maximum of 48 bytes (384 bits) as input. It should be recalled at this point that M can be any length less than 512 bits because the HMAC-SHA256 algorithm will take care of the padding. The label **“Plaintext = K^+ || M”** shows the concatenation of the generated K^+ and the entered M. The RSA components n and e are displayed for the purpose of encrypting the concatenation of K^+ and M. The button **“RSA Encryption”** is used to perform the encryption of K^+ || M and produce Cipher 3. A click on the button **“Next”** displays the second window of the SAS. Figure 11 shows the first window of the SAS.


Session Key Exchange Encryption (Master Side)

Generate K+: 98B51C60D1B72DDC406679909BD7860072090AB4319C1
Message: M: hello
Plaintext = K+ || M: 909BD7860072090AB4319C10380A70061E7F65C6CChello
Modulus n: 14351
Public Key Exponent (e): 11
RSA Encryption: Cipher 3: 624661BFFFFFFD72DFFFFFFFF21FFFFFFF236C1DFFFFFF

Figure 11: Session key exchange (encryption) master side

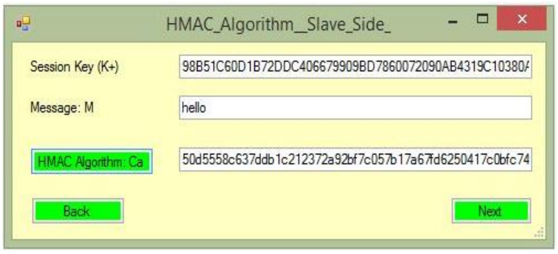
Second Windows of the SAS (SAS2) Named Session Key Exchange (Slave Side): This window displays Cipher 3 in the Slave side. The slave uses the RSA algorithm to decrypt Cipher 3 and get back K^+ || M. The required RSA components (n and d) for decryption are shown with their values. The modulus n = 14351 and the private exponent d = 1283. A click on the button **“RSA Decryption”** decrypts Cipher 3 and displays K^+ || M. At the same time the concatenation is broken and the labels **“Session Key (K^+)”** and **“Message: M”** display K^+ and M respectively. At this point the slave has K^+ and M for the computation of the HMAC-SHA256 algorithm. The button **“Next”** displays the computation of the HMAC-SHA256 algorithm in the following window. Figure 12 shows the second window of the SAS.



Cipher 3 624661BFFFFFFD72DFFFFFFFF21FFFFFFF236C1DFFFFF
Modulus n 14351
Private Key Exponent (d) 1283
RSA Decryption 98B51C60D1B72DDC406679909BD7860072090AB4319C
Session Key (K+) 98B51C60D1B72DDC406679909BD7860072090AB4319C
Message: M hello

Figure 12: Session key exchange (decryption) slave side

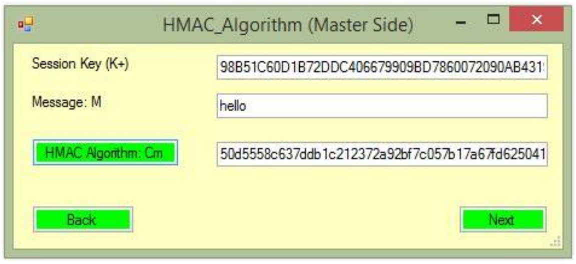
Third Windows of the SAS (SAS3) Named HMAC Algorithm Slave Side: Upon clicking the button **"Next"** in the previous window, SAS3 display K^+ and M separately. The button **"HMAC Algorithm"** serves to compute HMAC-SHA256 algorithm and output a commitment value (Ca) of 256 bits. Figure 13 shows the third window of the SAS.



Session Key (K+) 98B51C60D1B72DDC406679909BD7860072090AB4319C10380
Message: M hello
HMAC Algorithm: Ca 50d5558c637ddb1c212372a92bf7c057b17a67fd6250417c0bfc74

Figure 13: HMAC algorithm slave side

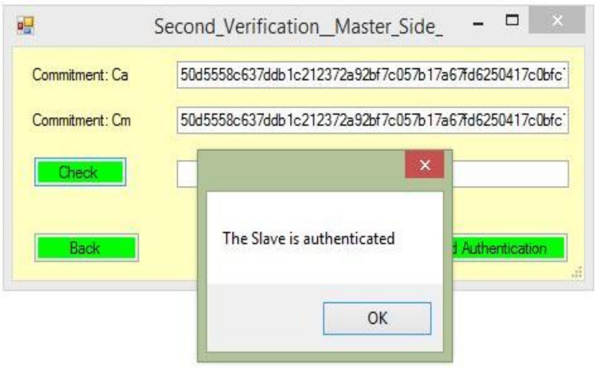
Fourth Windows of the SAS (SAS4) Named HMAC Algorithm Master Side: A click on the button **"Next"** in the SAS3 window displays K^+ and M in the SAS4 window. The button **"HMAC Algorithm"** allows computing HMAC-SHA256 algorithm and outputting a commitment value (Cm) of 32 bytes. Figure 14 shows the fourth window of the SAS.



Session Key (K+) 98B51C60D1B72DDC406679909BD7860072090AB431
Message: M hello
HMAC Algorithm: Cm 50d5558c637ddb1c212372a92bf7c057b17a67fd625041

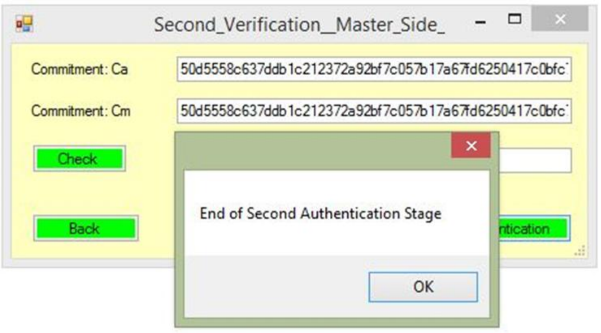
Figure 14: HMAC algorithm master side

Fifth Windows of the SAS (SAS5) Named Second Verification Master Side: A click on the button **"Next"** in the SAS4 window displays this last window with C_a and C_m for comparison. The button **"Check"** permits the master to perform the verification. By clicking on it either of the following messages will appear in a small window: **"The slave is authenticated"** if the two commitments values match or **"The slave is unauthenticated"** if they are different. In case of unsuccessful authentication, the connection will be aborted. The button **"End of Second Authentication"** displays in a small window the message **"End of Second Authentication"** which means that the SAS is completed. By clicking the button **"OK"** in this small window, devices can now establish the connection and securely exchange encrypted data. Figure 15 a) and b) show these different operations.



Commitment: Ca 50d5558c637ddb1c212372a92bf7c057b17a67fd6250417c0bfc
Commitment: Cm 50d5558c637ddb1c212372a92bf7c057b17a67fd6250417c0bfc
The Slave is authenticated

Figure 15 a): Second verification master side



Commitment: Ca 50d5558c637ddb1c212372a92bf7c057b17a67fd6250417c0bfc
Commitment: Cm 50d5558c637ddb1c212372a92bf7c057b17a67fd6250417c0bfc
End of Second Authentication Stage

Figure 15 b): Second Verification Master Side

VI. DISCUSSION

The above implementation result has shown the workability of the proposal on the listed hardware and software. All involved algorithms are computed and the result is presented a GUI. Each window is shown with different labels and button for the execution of the process. The result has shown that the security issues in SSP which are passive and active eavesdropping are completely tackled by preventing the exchange of public keys and passwords in clear text. Involved secret passwords and public keys are locally stored and the secret parameters (IV and secret keys) are generated randomly by using a strong random generator number then are securely exchanged with the RSA algorithm. Other exchanged ciphers cannot be tampered by an attacker due to the non-availability of the encryption keys to a third user and also the strength of the used encryption systems. Moreover, strong and long passwords are supported to generate K_c which is used for AES encryption. Passwords are kept secret and refreshed periodically. In addition to this good performance of the model, this model is totally different to both Legacy pairing and SSP in the sense that it employed a dual authentication for more security. It is highly expected that this proposal will replace SSP for authentication.

VII. CONCLUSION

This paper started by given an overview of Bluetooth then discussed the weaknesses in existing authentication methods. Besides that, some improved works have been reviewed prior to outlining the proposed solution for the current security issues in Bluetooth connection. The model was validated by using a BT shield connected to an Arduino board and the obtained result was observed with conformity in a GUI. The entire process of the authentication was clearly shown in snipped windows. The proposal defeats eavesdropping and MITM attacks which exist in SSP and is therefore seen to be the third authentication method in Bluetooth.

In addition, we address here an issue which has not been considered in the scheme. We believe that this issue is very crucial and can be mentioned as part of future research with this work forming the basis: the communication in a piconet is mostly between the master and slaves. However, if two slaves would like to communicate, their traffic must be relayed through the master. Therefore, it is highly important to find a way for performing mutual authenticating between the two slaves. This issue is not taken into consideration in this dissertation since the database of the master is not shared to the slaves. Hence, a slave cannot authenticate another one. We hope to tackle this issue in the future work.

REFERENCES

[1] V. Garg, *Wireless Personal Area Network-Bluetooth,* in Wireless Communications & Networking: Series Editor, David Clark, M.I.T. (Amsterdam, the Morgan Kaufmann series in networking, 2010) pp. 653-674.

[2] C. Gehrmann, J. Persson, and B. Smeets, *Bluetooth security:* Artech house, 2004.

[3] K. Saravanan and D. Yuvaraj, "*An new secure mechanism for bluetooth network,*" *in Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on,* 2010, pp. 202-205.

[4] J. Padgette, K. Scarfone, and L. Chen, "*Guide to Bluetooth security,*" *NIST Special Publication, vol. 800,* p. 121, 2012.

[5] J. Barnickel, J. Wang, and U. Meyer, "*Implementing an attack on bluetooth 2.1+ secure simple pairing in passkey entry mode,*" *in Trust, Security and Privacy in Computing and Communications (TrustCom), 2012 IEEE 11th International Conference on,* 2012, pp. 17-24.

[6] J. Linksky, *Simple Pairing Whitepaper, revision v10r00. Bluetooth Special Interest Group (SIG). Core Specification Working Group*. 2006, pp.1-23.

[7] Y. Shaked and A. Wool, "*Cracking the bluetooth pin,*" in Proceedings of *the 3rd international conference on Mobile systems, applications, and services,* 2005, pp. 39-50.

[8] M. T. Alberto, "*Sniffing the Bluetooth pairing*". 2011. Accessed on 15/03/2013 at 3:30 pm. <*http://www.seguridadmobile. com/bluetooth/ bluetooth-security/sniffing-the-Bluetooth-pairing.html*>.

[9] M. Colin "*iOS passwords: Quick tips to maximize your security*" (2012). Accessed on 15/03/2013 at 8:00 am. <*http:// www.fichnetsecurity.com/6labs/blog/ios-passwords-quick-tips-maximise-your - security*>.

[10] A. Y. Lindell, "*Attacks on the pairing protocol of bluetooth v2. 1,*" Black Hat USA, Las Vegas, Nevada, 2008.

[11] A. Iman, Al S., Mohammed, Al A Mousa, "Secure public key exchange against man-in-the middle attack during secure simple pairing (SSP) in Bluetooth". *World Applied Sciences Journal 13 (4): 769-780, ISSN 1818-4952.* Computer Science Department Computer Information System Department King Abdullah II School for IT. The University of Jordan. 2011, pp.1-12.

[12] K. Hypponen and K. M. Haataja, *"“Nino” man-in-the-middle attack on bluetooth secure simple pairing,*" in Internet, 2007. *ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on,* 2007, pp. 1-5.

[13] K. M. Haataja and K. Hypponen, "*Man-in-the-middle attacks on bluetooth: a comparative analysis, a novel attack, and countermeasures,*" in Communications, Control and Signal Processing, 2008. *ISCCSP 2008. 3rd International Symposium on,* 2008, pp. 1096-1102.

[14] K. Haataja and P. Toivanen, "*Practical man-in-the-middle attacks against bluetooth secure simple pairing,*" *in Wireless Communications, Networking and Mobile Computing, 2008. WiCOM'08. 4th International Conference on,* 2008, pp. 1-5.

[15] D. Sharmila, R. Neelaveni, and K. Kiruba, "*Bluetooth man-in-the-middle attack based on secure simple pairing using out of band association model,*" *in Control, Automation, Communication and Energy Conservation,*

2009. INCACEC 2009. 2009 International Conference on, 2009, pp. 1-6.

[16] A. Levi, E. Çetintaş, M. Aydos, Ç. K. Koç, and M. U. Çağlayan, "*Relay attacks on bluetooth authentication and solutions," in Computer and Information Sciences-ISCIS 2004, ed: Springer,* 2004, pp. 278-288.

[17] P. R. Suri and S. Rani, "*Bluetooth security-Need to increase the efficiency in pairing*," *in Southeastcon, 2008. IEEE,* 2008, pp. 607-609.

[18] T. Kumar, "*Improving pairing mechanism in Bluetooth security," Int. J. Recent Trends Eng, vol. 2,* 2009.

[19] C.-M. Fan, S. Shieh, and B.-H. Li, "*On the security of password-based pairing protocol in bluetooth*," *in Network Operations and Management Symposium (APNOMS), 2011 13th Asia-Pacific,* 2011, pp. 1-4.