2008

Improved Bluetooth Key Exchange using Unbalanced RSA

Saif Rahman
University of Windsor

Improved Bluetooth Key Exchange Using Unbalanced RSA

by

Saif Rahman

A Thesis
Submitted to the Faculty of Graduate Studies through the Department of Electrical and Computer Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Applied Science at the University of Windsor

Windsor, Ontario, Canada
2008

Your file Votre référence
ISBN: 978-0-494-47043-5
Our file Notre référence
ISBN: 978-0-494-47043-5

Author's Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

In this thesis, a new protocol is proposed for the Bluetooth Key Exchange. The proposed key exchange will make use of a public-key algorithm as compared to the currently existing key exchange which only uses symmetric ciphers. The public-key algorithm to be used is a modified version of the RSA algorithm called "Unbalanced RSA". The proposed scheme will improve on the currently existing key exchange scheme by improving the security while trying to minimize computation time. The proposed protocol will also improve on a recent work which used the Diffie-Hellman algorithm for Bluetooth key exchange. In using the Diffie-Hellman algorithm the security was increased from the original Bluetooth key exchange but the computation time and difficulty of computations was also increased. Two Bluetooth devices that are trying to communicate can have a wide range of processor speeds and the use of the Diffie-Hellman protocol can cause a large delay at one user. The use of Unbalanced RSA in the proposed protocol will aim to remedy this problem. The aim of the proposed protocol is to eliminate the security risks from the original Bluetooth key exchange and also address the computation time issue with the enhanced Diffie-Hellman key exchange.

Dedication

First and foremost, I would like to dedicate this thesis to my parents who have provided me with unconditional support throughout this process. Without their support and guidance, completing my degree would have been a much more difficult task. I would also like to dedicate this thesis to my brother (Zia), sister (Saba), brother-in-law (Naveed), and my nephew (Saad). All of whom have been a great source of motivation and inspiration. Finally, I would like to thank all of my friends and extended family.

Acknowledgement

I would like to thank my advisor, Dr. Huapeng Wu, for his patience and encouragement during the difficult times, as well as for his insights and suggestions that helped to enhance my research skills. Dr. Wu's constructive feedback contributed greatly towards the completion of this thesis. I would also like to thank Dr. Jessica Chen and Dr. Mitra Mirhassani for their valuable views and comments.

Table of Contents

List of Tables

List of Figures

Abbreviations

ACO	Authenticated Ciphering Offset
AES	Advanced Encryption Standard
BD1	Bluetooth Device 1
BD2	Bluetooth Device 2
BT	Bluetooth
DH	Diffie-Hellman Key Exchange
IEEE	Institute of Electrical and Electronics Engineers
IDBD1	Address of BD1
IDBD2	Address of BD2
IR	Infrared Technology
MAC	Message Authentication Code
MD5	Message Digest 5
RSA	Rivest, Shamir, Adelman (Public-Key Encryption)
URSA	"Unbalanced RSA"
WLAN	Wireless Local Area Network

1. Introduction

Wireless network communication has been paid more and more attention during the past decade since it makes it much easier to access a network compared to its wired counterparts. On one hand, wireless networks can provide great convenience for the people in communication; one the other hand, it proposes a tougher task of network security since it also make it much easier for an attacker to intercept a message transmitted in wireless. There are many types of wireless networks, for example, wireless LAN (WLAN), wireless MAN (WMAN), wireless PAN (WPAN), and ad hoc wireless network. In this thesis, the security of WPAN is our main concern.

In recent years the Wireless Personal Area Network or WPAN has become a popular method for two devices in close proximity to exchange information. The WPAN consists of technologies such as Infrared (IR), Bluetooth, UWB, and Zigbee. When PAN first became popular IR technology was the popular way to exchange information between two wireless devices [1]. Infrared technology uses wireless technology in devices that convey data through IR radiation. IR technology is used for short and medium range communication. Many devices operate in the "line-of-sight mode", which means that the two devices must have a straight line of sight between them. This is not very helpful for people who are always on the move. Another mode for infrared devices is called the scatter mode where a device does not have to be in direct sight of the device but has to be in the same room or just outside the room with a door open. Another limitation of IR technology is that its signal cannot pass through walls so it cannot communicate between two different rooms in a house. Nowadays Infrared is being phased out of devices in exchange for Bluetooth (BT)

technology. BT technology improves on the mobility of infrared devices by allowing users to be farther away and in different rooms when they are exchanging information. Bluetooth can allow devices to be up to 100 meters away and still be able to connect to one another [2]. It also allows for a data transmission rate of up to 3 megabits per second. Overall, Bluetooth is a good solution to the problems faced by infrared technologies and that is why it is taking over as the predominant technology for wireless file transfer.

Over the past few years the number of Bluetooth users has increased rapidly. From 2003 to 2006, the number of Bluetooth users has nearly doubled every year going from 125 million to 1 billion users as predicted by the November 2006 issue of SDA magazine and [3]. The number of Bluetooth equipped devices keeps increasing. In 2009, 80% of mobile phones will be Bluetooth enabled [4]. According to Microsoft, in 2008, greater than 54% of all laptops that are shipped are equipped with Bluetooth technology. As of right now Bluetooth is not really being used for exchanging very vital information. As more powerful devices become equipped with Bluetooth, it will be used to exchange important information and therefore there needs to be no uncertainty in the security of Bluetooth. Bluetooth security right now is acceptable for the way it is being used but as Bluetooth is being used to exchange secretive information, the security needs to be improved.

Security in Bluetooth is a major issue and failure to use proper security measures can cause several problems for Bluetooth users. First of all, the Bluetooth standard describes three modes of security which are shown in Table 1.1 [5]. Modes 2 and 3 require two devices to complete the pairing process whereas mode 1 does not require it.

Security Mode	Description
1	No Security.
2	Service Level Security.
3	Link Level Security.

Table 1.1: Bluetooth Security Modes

Failure to employ a proper mode of security can lead to several vulnerabilities in the Bluetooth device. These vulnerabilities are listed below [5]:

- Sensitive data is available for browsing
- An attacker can use a compromised telephone to make calls
- Denial of Service attacks can be launched against the compromised device
- Address lists can be downloaded
- Malware can be installed for later infection of other devices, including network attached systems
- An attacker can install malware with the intent to gain ongoing control of the device

To combat these vulnerabilities several different approaches can be taken. First of all one should not operate Bluetooth Devices in Mode 1 because no security is provided. Also when Bluetooth is not being used, turn it off so that it is not discoverable. Turn it on when trying to connect to somebody. An easy way to ensure that the device is safe is to minimize the distance between it and the other Bluetooth device. It is also a good idea to install anti-virus software to keep you safe from Malware. These Bluetooth vulnerabilities have already been addressed and users for the most part are aware of them and can combat them. Some types of attacks on Bluetooth however use complex methods to obtain confidential information. These attacks are known as passive eavesdropping, active eavesdropping, and bluedumping. These attacks occur during the key exchange operation and obtain the link keys so that the attacker can decrypt all information sent form one device to the other. In [6] the author agrees that eavesdropping attacks are a major problem in current Bluetooth security. The key exchange scheme proposed in this Thesis will help to fight against these attacks.

The scheme proposed in this Thesis uses public-key cryptography during the Bluetooth key exchange to provide better security while also trying to minimize the time delay. The proposed scheme will help to protect against the

common key exchange attacks of passive eavesdropping, active eavesdropping, and bluedumping because of the use of public-key cryptography. The proposed algorithm uses a slightly modified version of the RSA algorithm which has considerably smaller time delay than other public-key algorithms while still providing a high level of security.

The next chapter reviews the existing Bluetooth Privacy and key exchange schemes. Chapter 3 revisits previously proposed work on the Bluetooth key exchange using public-key cryptography. Chapter 4 reviews the RSA algorithm, and introduces a modified version of RSA known as "Unbalanced RSA". The new protocol using Unbalanced RSA is proposed in chapter 5. This chapter will also discuss the security aspects of the proposed protocol. Chapter 6 will go over in detail the simulation results and also provides analysis on these results. Conclusions and ideas for future work are given in the final chapter.

2. An Overview on Bluetooth Privacy and Key Exchange

IEEE 802.15 is a working group of the IEEE 802, which specializes in Wireless Personal Area Networks. This chapter focuses on the security portion of the IEEE 802.15 standard and more specifically on the Key exchange.

2.1 An Overview of Bluetooth

Bluetooth was created by Ericsson in 1994 to essentially replace wired networks. In 1998 another step was taken and a Bluetooth special interest group (SIG) was formed, the founders were from Ericsson, IBM, Intel, Toshiba and Nokia [7]. By the year 2004 Bluetooth was already supported by over 2100 companies all over the world [8]. Bluetooth allows devices to connect and exchange information without any wires or external devices. The only requirement is that the device is Bluetooth enabled. Bluetooth has many different uses and is used in a variety of devices such as cell phones, laptops, printers, headsets, video games etc. The introduction of Bluetooth has been quite revolutionary and actually helps people in everyday life. The use of Bluetooth enabled headsets, allows people to connect the headset to their phone so they can take calls even while driving without getting distracted from the road. Bluetooth does of course have a limited range and the range is not that of a WLAN. However, depending on the class of the device being used the range can be quite good. For example, for a Bluetooth headset, a large range is not required therefore a class 2 or 3 device can be used which can provide up to 10 meter range. However, for a laptop, a larger range is required so a class 1 device with 100m range would be appropriate [9]. Prior to Bluetooth, Infrared technology

was commonly used to exchange information between two wireless devices. This technology proved to be quite rudimentary when compared to Bluetooth. With IR technology, the two devices usually had to have a direct line of sight to one another or there couldn't be any walls or other obstacles between them [10]. Once Bluetooth was introduced IR technology was quickly faded out.

Another wireless service that has constantly been compared to Bluetooth is Wireless internet (WLAN). Wireless internet or Wi-Fi is very common these days and is standard in laptops. Bluetooth is different from Wi-Fi in that it does not require any external products in order to be used. The only requirement is that the two devices have the Bluetooth technology and can connect to one another and exchange information. The similarity between Bluetooth and Wi-Fi is that they are both used in households and offices. Wi-Fi covers a wider range than Bluetooth, whereas Bluetooth is less expensive and has lower power consumption [11]. In general, Bluetooth is efficient for exchanging small files or documents when two devices are in close proximity of one another.

Security concerns play a major role in Bluetooth technology. The goal of Bluetooth security is to allow devices to connect to one another and/or exchange information without being compromised. In the early years of Bluetooth, there were some very lethal attacks such as bluejacking, bluesnarfing, or bluebugging which enabled attackers to use the Bluetooth device without the user's knowledge [12]. However, security patches have been used to remedy these problems [13]. In recent years Bluetooth security has become a more serious issue as it is being used to exchange confidential information. The main issue is the key exchange process where an attacker can use passive/active eavesdropping or bluedumping attack to obtain secret keys. Once an attacker is able to obtain the secret keys he/she will be able to decrypt all information being sent between the two Bluetooth devices. These attacks will be discussed in a later section and how they can be used to obtain the secret keys. The next section will go over the initial stages of Bluetooth security. The third section will review the Bluetooth key exchange process as well as the attacks possible on the key exchange.

2.2 Bluetooth Security

The Bluetooth Security process consists of Key generation and Initialization followed by encrypting and sending of messages. The initialization process is the fundamental base of Bluetooth security because that is where the secret keys are created and exchanged. The focus of this thesis will be on the first three parts of the initialization process and more specifically the key exchange. The initialization process consists of five steps [14]:

1) generation of initialization key
2) generation of link key
3) link key exchange
4) authentication
5) generation of encryption key

The first three steps of the initialization process make up the key exchange and that is where the main focus will be. Prior to discussing the initialization process a brief discussion on Bluetooth security services will be provided.

2.2.1 Security Services

Bluetooth provides five different kinds of security services using different kinds of mechanisms. The five security services provided by Bluetooth are authentication, access control, data confidentiality, data integrity, and non-repudiation. Authentication is provided through a challenge-response scheme in which the claimant is asked to prove its knowledge of the secret key using a symmetric cipher. The authentication procedure uses the AES candidate SAFER+ as its base and modifies it to fit the needs of Bluetooth [14]. Access control is provided to Bluetooth through the use of PIN numbers entered in the devices. When two devices are trying to connect to one another it is required that they each enter the same PIN in their devices so that it is known that these devices want to be matched with one another. If the same PINs are not entered into the devices they will not be allowed to connect to one another. Data

confidentiality is provided through an encryption scheme. The encryption of the payload information is done by a stream cipher. The key stream generator will generate and stream through the key which will then be XORed with the plain text or cipher text depending on whether encryption or decryption is being performed. In order to make sure data integrity is maintained Bluetooth has many error codes which will occur if there is any problem with the two devices that are trying to communicate. These error codes are used to inform the parties if the message could not be delivered for some reason. The errors will let the parties know what problem has occurred so that they can fix it and resend the data. The unique address of each Bluetooth device provides accountability in terms of the last security service, non-repudiation.

2.2.2 Key Types

There are four different key types that can be used as an authentication key, encryption key, or initialization key. The four different key types [14] are 1) Combination key (K_{1-2}, K_{2-1}), Unit Key (K_{unit}), Temporary Key (K_{master}), and Initialization key (K_{init}). The unit key K_{unit} is derived solely from information from Bluetooth Device 1. The combination Key is derived from information in both Bluetooth Device 1 (BD1) and Bluetooth Device 2 (BD2). The master key K_{master} will be used during the session currently in progress and it will replace the original link key temporarily. The initialization key is used during the initialization process before any combination or unit keys have been defined or exchanged. Initialization key is derived from a PIN code and a device address. The PIN can be a fixed number that came with the device or can be selected by the user and entered into both devices that are trying to connect to one another.

2.2.3 Key Generation

For authentication purposes there are two different algorithms used for generating keys. In figure 2.1 both these algorithms are shown as well as their inputs and outputs. When trying to generate a unit key or combination key the E_{21} function is used, this will produce a 128-bit key using a 128-bit random number

and 48-bit address. When trying to generate an initialization key or a master key the E_{22} function is used, this will produce a 128-bit key from a 128-bit random number and an L octet user PIN.

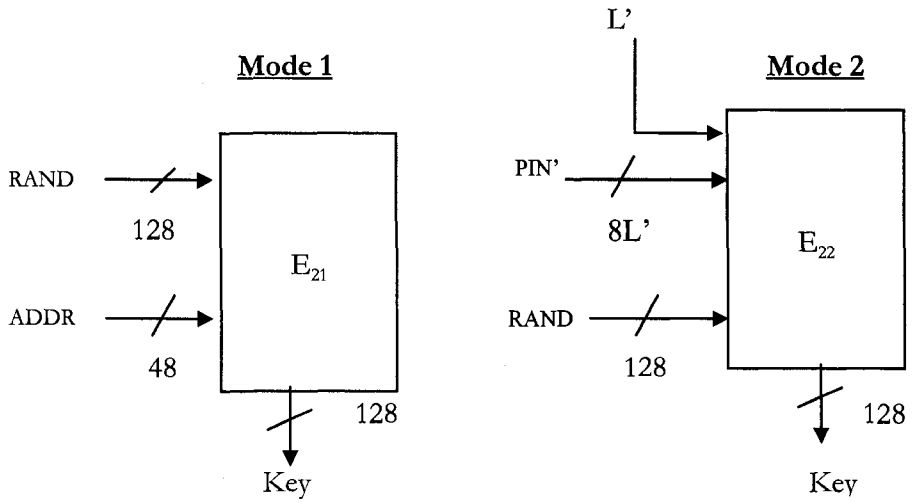

Figure 2.1: BT Key Generating Algorithm E_2 [14]

Figure 2.2 shows how the encryption key is generated using the E_3 algorithm. The E_3 algorithm produces 128-bit key using a 128-bit random number, a 96-bit Cipher offset, and the 128-bit link key.

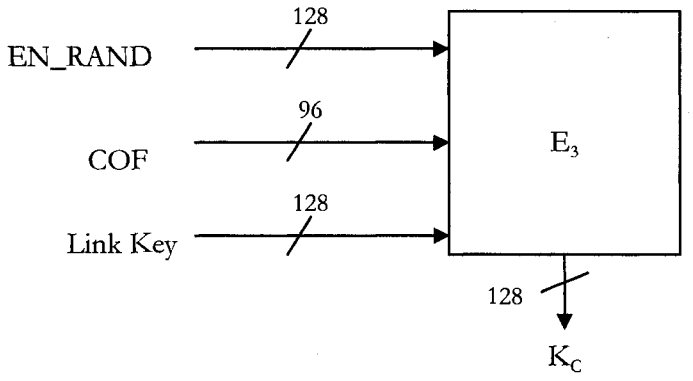

Figure 2.2: Generation of BT encryption key [14]

2.2.4 Initialization Key

The initialization key is generated using the E_{22} algorithm. This algorithm takes as input the device address, PIN code, length of PIN (in octets), and a random number. During each authentication process a new random number will be issued to provide better security. It is also possible to do mutual authentication so that each device knows exactly who it is communicating with. Once a successful authentication has been completed an auxiliary parameter, the authenticated ciphering offset (ACO) will be computed and the ACO will be used for ciphering key generation.

2.2.5 Unit Key

To generate a unit key the only information that is needed is from Bluetooth Device 1 (BD1). Figure 2.3 below shows how BD1 will send the unit key to BD2 so that they can communicate with each other. As can be seen from the figure, BD1 will generate K_{unit} and then XOR with the initialization key so that K_{unit} is not transported in the clear. Although this is a very simple encryption it does provide some type of security. Once the encrypted key reaches BD2, it will decrypt the key by XORing it with the initialization key to obtain K_{unit}, which means that this key will always be used by BD2 when it is trying to communicate with BD1.

Figure 2.3: Generation of BT Unit Key [14]

2.2.6 Combination Key

When generating a combination key it is slightly more complicated than generating a unit key because both Bluetooth Device 1 (BD1) and Bluetooth Device 2 (BD2) need to be involved in the generation and they both need the

others information. Figure 2.4 below gives a diagrammed version of the steps that need to be taken in order to create a combination key K_{1-2}. Firstly, BD1 will create its part of the combination key AK_K_1 by inputting a random number (AK_RAND_1) and the address of BD1 (BD_ADDR_1) into the E_{21} algorithm. Once BD1 has created AK_K_1 it will create C_1 by XORing AK_RAND_1 and the initialization key. Once this is completed BD1 will send C_1 to BD2. On the other side BD2 will be calculating AK_K_2 and C_2 using another random number (AK_RAND_2), device address of BD2 (BD_ADDR_2), and the initialization key (K_{init}). Once BD2 completes its calculations it will send C_2 to device BD1. Once BD1 receives C_2 it can calculate AK_RAND_2 by taking C_2 and XORing it with K_{init}. Now that BD1 knows AK_RAND_2 it can calculate AK_K_2 by putting AK_RAND_2 and BD_ADDR_2 into the E_{21} algorithm. BD1 now has AK_K_2, it can calculate the combination key which is done by XORing AK_K_1 with AK_K_2 to obtain K_{1-2}. Device B does exactly the same thing to calculate AK_K_1 and then it will XOR AK_K_1 and AK_K_2 to obtain K_{2-1} which is the same as K_{1-2}. The combination key is by far the commonly used method to obtain a Bluetooth link key.

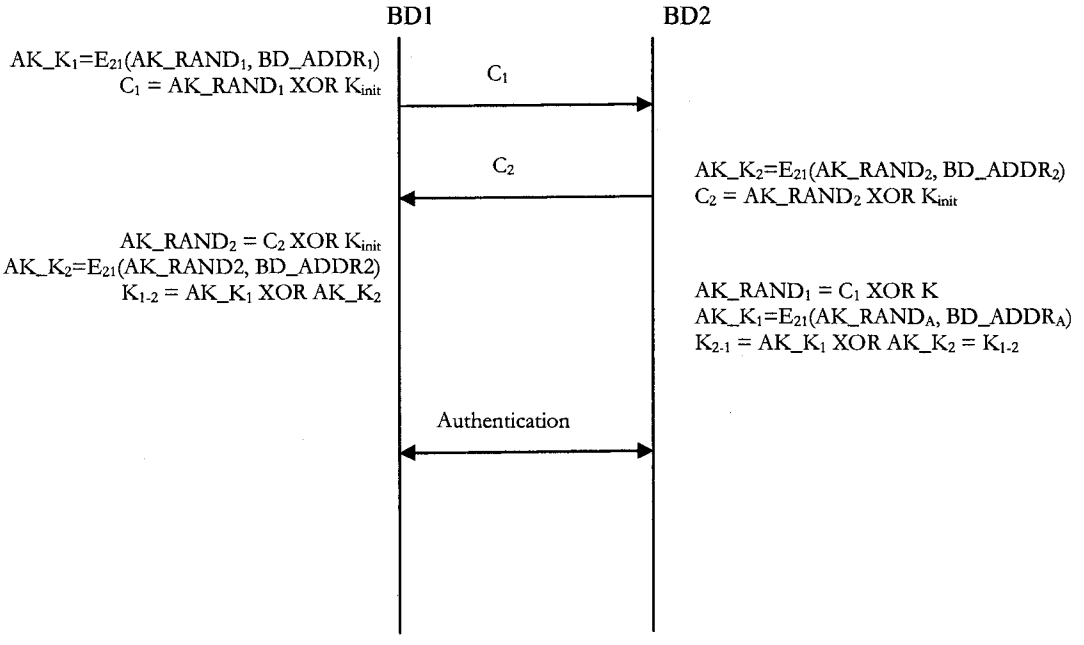

Figure 2.4: Generation of BT Combination Key [14]

2.2.7 Master/Temporary Key

The process for generating a master key for the current session is a little bit simpler than that for a combination key. When the master (BD1) wants to create a master key and deliver it to BD2 it follows the steps outlined in Figure 2.5 below. The figure shows that Bluetooth Device 1 will first calculate K_{master} by using the E_{22} algorithm with inputs of 2 random numbers and the length of the random number in octets. The next step is for BD1 to send a random number to BD2. This number is used to calculate a value called the overlay, which is obtained by inputting the link key, the random number and the length of the random number into the E_{22} algorithm. BD1 will also calculate this overlay and then the overlay and the master key will be XORed together to form C. The value C will be sent to BD2 and it can now XOR the overlay it calculated with the value C that was sent in order to obtain K_{master}.

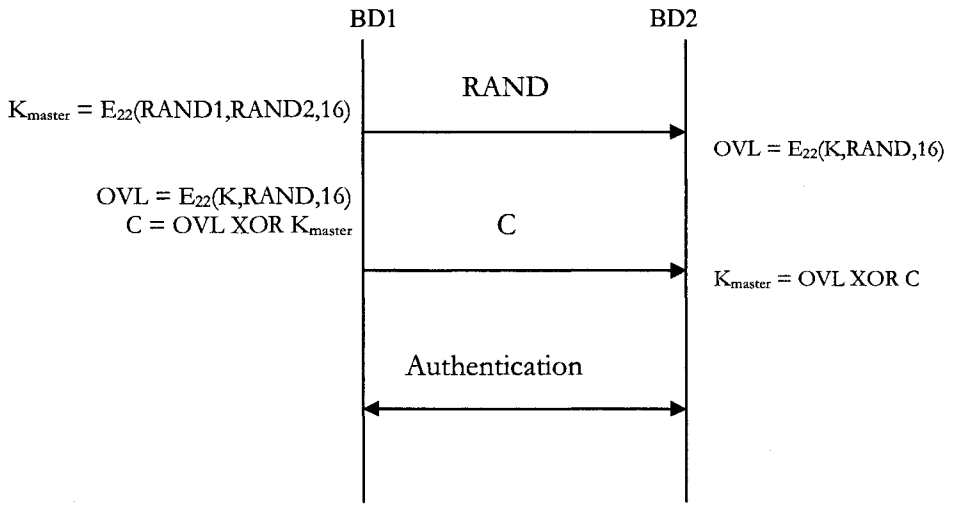

Figure 2.5: Generation of BT Master Key [14]

2.3 Bluetooth Key exchange

The Bluetooth key exchange combines the first three steps of the key generation and initialization process. It includes the generation of an initialization key by both devices, the generation of link keys, and the exchange of link keys.

Figure 2.6 shows these three steps together in one diagram. This is the entire process which will be replaced with the proposed protocol.

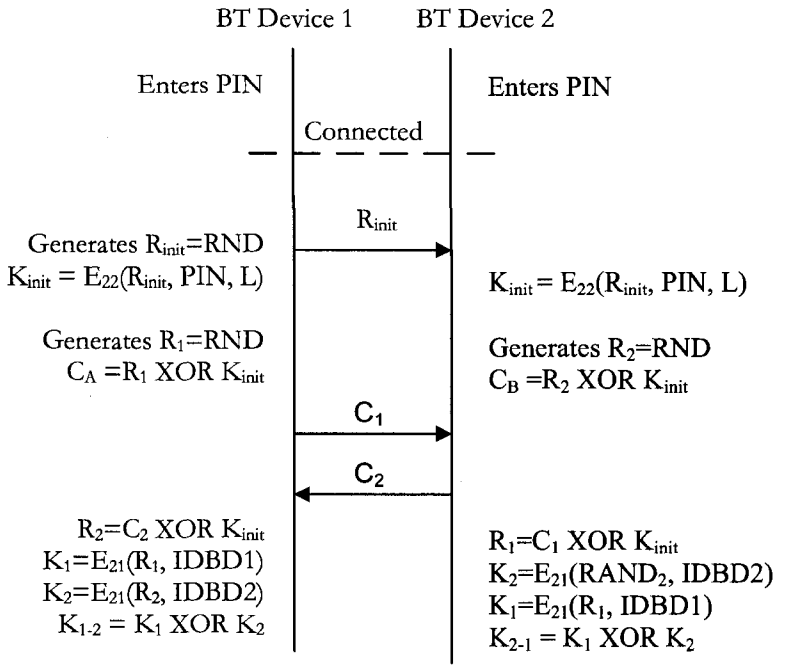

Figure 2.6: Bluetooth Key Exchange

The first step in the Bluetooth key exchange scheme is for the two users to enter the agreed upon PIN into each of their respective devices. Once the PINs have been entered into the devices, the two devices connect to one another. These two devices will be called Bluetooth Device 1 (BD1) and Bluetooth device 2 (BD2). BD1 will generate a random number R_{init} using a random number generator. R_{init} will be used to generate the initialization key. Once BD1 generates R_{init} it will send it to BD2. Now that both devices have R_{init} they can calculate the initialization key (K_{init}) using the E_{22} function. The input to the E_{22} function will be the R_{init}, PIN, and L (length of PIN), which will produce the output K_{init}. Once both devices have obtained the K_{init} they have to begin the process for exchanging the Link Key. BD1 and BD2 will generate the random numbers R_1 and R_2 respectively. The next steps are almost the same as generating a combination key. BD1 and BD2 will generate C_1 and C_2 respectively by XORing the

initialization key with the random numbers they generated. BD1 will then send C_1 to BD2 and BD2 will send C_2 to BD1. Using C_2, BD1 will obtain R_2 and similarly BD2 will obtain R_1. Both devices now have R_1 and R_2 as well as IDBD1 and IDBD2. They each generate K_1 ($K_1=E_{21}(R_1, IDBD1)$) and K_2 ($K_2=E_{21}(R_2, IDBD2)$). By XORing K_1 and K_2, they have each obtained the shared link key.

2.3.1 Attacks on Bluetooth Key Exchange

In the early years of Bluetooth technology, several flaws were found in its implementation. These flaws were known as bluejacking, bluesnarfing, and bluebugging. Bluejacking is the sending of unwanted messages through Bluetooth to Bluetooth-enabled devices such as mobile phones and laptops [15]. Bluesnarfing allows users to access the information on a wireless device through a Bluetooth connection. Bluebugging allows a user to take control of the victim's Bluetooth enabled phone to make calls or send text messages. Bluesnarfing and Bluejacking are illegal in most countries and these attacks are usually only possible on older Bluetooth-enabled devices as the newer devices have updated software to avoid these problems.

Although these problems were of concern back then they have for the most part been taken care of. However, there is still one major problem that needs to be addressed. This major problem is the transfer of confidential information. With the pairing scheme in place right now, it would not be difficult for a person to eavesdrop (passively or actively) on a connection between two Bluetooth devices and obtain the link key. In [16] it was noted that Bluetooth had three major vulnerabilities: 1) Spoofing through keys, 2) Spoofing through address, and 3) PIN Length. The key exchange plays a major role in trying to counter these vulnerabilities and with the system in place now it is not too difficult to attack. In [17], [18], [19], [20] it is documented the use of only a PIN as secret entity is not sufficient to provide total security and a better pairing scheme is needed. In the near future it will be necessary for Bluetooth to provide better security.

Passive Eavesdropping

If a person is trying to send confidential information through Bluetooth, a passive eavesdropper would only need to find out the PIN both devices are using in order to be able to decrypt the messages being sent back and forth. Let's start at the very beginning. In order for the two devices to connect they need to establish a PIN that they both enter into their devices. Usually these PINs are 4 digit numbers and therefore are not that difficult to figure out. If an eavesdropper is able to obtain this PIN he can follow through all the Bluetooth security steps to obtain the encryption key that is going to be used. A figure of a passive eavesdropping attack is shown in Figure 2.7.

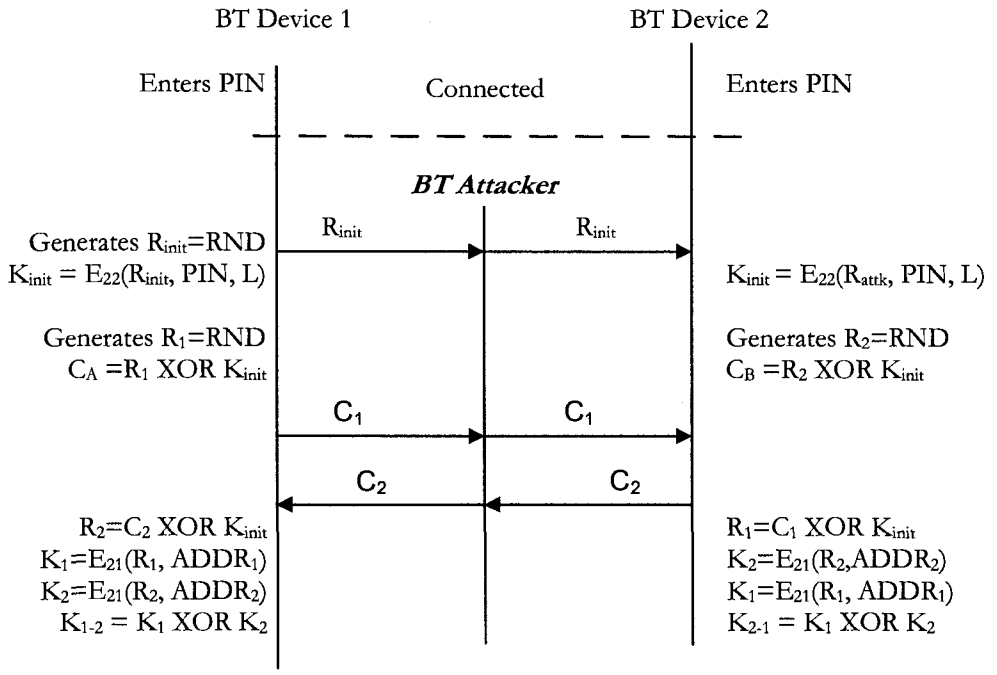

Figure 2.7: Passive eavesdropping attack on Bluetooth Key exchange

The steps an eavesdropper would use are as follows:

1. Obtain PIN by guess or obtaining from use
2. intercept random number sent from one Bluetooth device to another
3. Use random number to derive initialization key through the E_{22} function

4. Use initialization key and combination key format to obtain random numbers from both Bluetooth device users
5. Use random numbers to derive the portions of the authentication key from both users ($AK\_K=E_{21}(RAND, ADDR)$)
6. Use AK_K_1 and AK_K_2 to obtain the link key (K_{link} = AK_K_1 XOR AK_K_2)
7. eavesdroppers can put link key through authentication scheme to obtain SRES and compare it with the SRES of the Bluetooth devices
8. The link key is then put through E_3 function along with the COF and a random number to obtain the ciphering key
9. The ciphering key is input to the payload key generator
10. The payload key is then input to the key stream generator and the key stream is generated
11. The eavesdropper can now use the key stream and XOR it with the cipher text to obtain all the plaintext.

Active Eavesdropping

An active eavesdropper is similar to a passive eavesdropper with one exception; an active eavesdropper will actually go in and alter messages being sent from one user to the other. An active eavesdropper will alter messages so that it seems as though the BT users are communicating with the right person but actually there is an attacker intercepting their messages and altering them. Figure 2.8 below shows that two Bluetooth devices are trying to create a combination key. However there is an attacker intercepting the messages and altering them to make it seem like the two devices are still communicating with each other. This type of attack is also known as the man-in-the-middle attacks.

In this scheme the attacker starts intercepting the messages after the PINs are entered by both users. As the diagram shows, every message that is supposed to go from Device 1 to Device 2 is intercepted by the attacker and the attacker then sends its own message to device 2. Same thing happens when device 2 is sending a message to device 1. All the items that are bolded and

italicized in the figure are the ones that are manipulated by the attacker. The figure shows that the two Bluetooth devices still believe they are speaking to the correct person without any interference because they are receiving the proper messages. The attacker is the only device with all the proper values and at the end after the attacker performs the same process on authentication he/she will be able to decrypt any messages exchanges between Device 1 and Device 2. This attack also ensures that any message sent from device 1 to device 2 will not be decrypted by device 2 since it did not receive the proper values during key exchange. In passive eavesdropping however, device 2 is still able to decrypt the messages sent by device 1 because none of the information during key exchange had been altered.

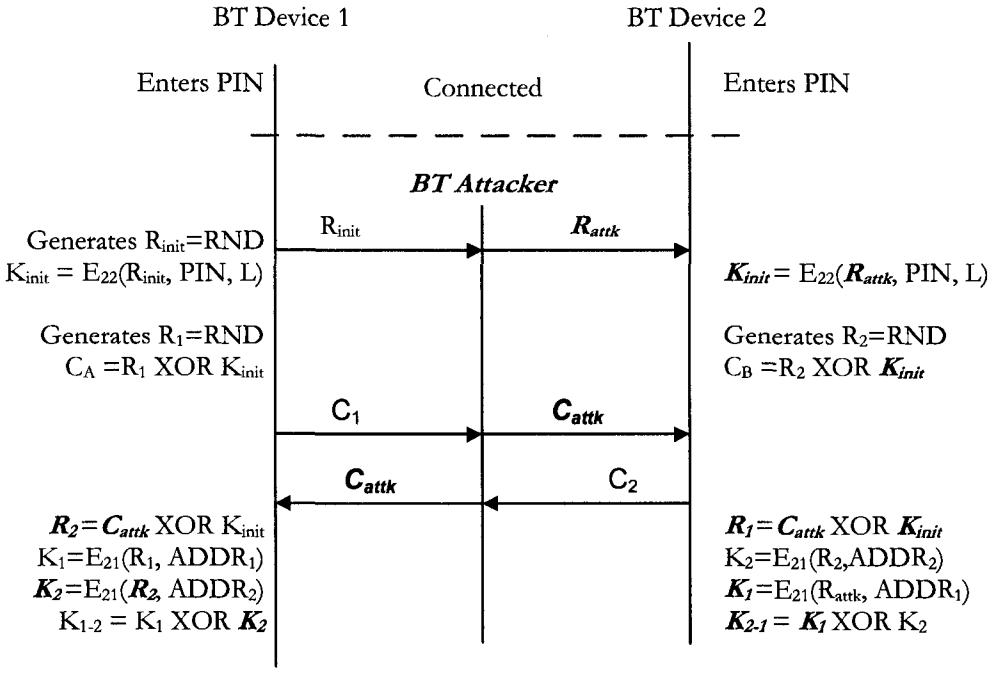

Figure 2.8: Man-in-the-middle attack on BT Key Exchange

Passive and Active Eavesdropping (bluedumping)

A combination of passive and active eavesdropping is called bluedumping. This attack causes a Bluetooth device to unload its stored link key to a specific device, which forces these two devices to perform their key exchange and

therefore the attacker has another opportunity to eavesdrop and obtain the pairs link key [21]. The process of this attack is quite simple. The attacker will send one of the Bluetooth devices a message claiming to be another device and claiming to have lost the pairs link key. Once the Bluetooth device receives this message, it will reinitiate the key exchange with the other Bluetooth device. This will give the attacker an opportunity to see the messages and obtain the link key. Again this attack does depend on the attacker being able to obtain the PIN that is entered into both BT devices.

2.4 Summary

Bluetooth security in the past has been adequate for the users of Bluetooth. It provides users with the basic security services of authentication, access control, data confidentiality, data integrity, and non-repudiation. The current key exchange scheme may not be adequate in the near future as Bluetooth becomes commonly used for transferring important and confidential information. The key generation and key exchange process can be a weak link in the Bluetooth security architecture, which could lead to the leaking of confidential information.

The main objective of an attack on the key exchange portion of Bluetooth security is to obtain the link key that is created. The three types of attack described above are the main ways to obtain a link key from two unsuspecting Bluetooth devices. As can be seen from the above explanations the security of the key exchange depends purely on the secrecy of the PIN. Most Bluetooth users tend to use easy PINs and therefore making it not too difficult to crack the PIN. For example in 2005 it was found that to crack a 4-digit PIN on Pentium IV computer took only 0.063 seconds [22].

3. Recent Work on Bluetooth Key Exchange

A new proposal was introduced in 2004 to improve the Bluetooth key exchange scheme by Selim Aissi, Christian Gehrmann, and Kaisa Nyberg [23]. The authors of this protocol were from Intel, Ericsson and Nokia. This protocol proposed to use Diffie-Hellman in order to exchange the link keys between two Bluetooth devices. This protocol uses Diffie-Hellman and Hash functions to enhance the security of the link key while still allowing devices to have user-friendly PINs. It would provide strong protection against off-line attacks (passive eavesdropping), active eavesdropping and bluedumping.

3.1 Security Enhancement Using DH scheme

The Diffie-Hellman scheme was first published by Whitfield Diffie and Martin Hellman in 1976 [24]. The paper was the first that clearly defined public-key cryptography. The objective of Diffie-Hellman is to enable two users to safely exchange a key that can be used later for encryption of messages. The following steps are taken to exchange a key using Diffie-Hellman [25]:

1. A prime number q is selected and given to both users
2. α is chosen which is smaller than q; α must be a primitive root of q and given to both users
3. User A selects private key X_A
4. User A calculates public key $Y_A = (\alpha \hat{} X_A) \bmod q$
5. User A sends Y_A to User B
6. User B selects private key X_B
7. User A calculates public key $Y_B = (\alpha \hat{} X_B) \bmod q$

8. User B sends Y_B to User A
9. User A generates secret key $S = (Y_B \hat{} X_A) \bmod q$
10. User B generates secret key $S = (Y_A \hat{} X_B) \bmod q$

Once these steps are completed both users have the shared secret key S which can now be used for encryption of messages the users would like to send to one another. The use of private and public keys makes this scheme less susceptible to certain types of attacks.

The Diffie-Hellman enhanced key exchange protocol makes use of cryptographic one-way functions and hash functions to protect it from certain types of attacks. Although it does increase the number of protocol steps from 3 to 4, the level of security is increased. The DH protocol consists of two stages the registration stage and the key establishment stage. The registration stage contains the initial key generation, exchange of identities, and exchange of cryptographic verification values [23]. The key establishment stage consists of the DH key exchange. [23] The steps of the protocol are shown in Figure 3.1 and 3.2.

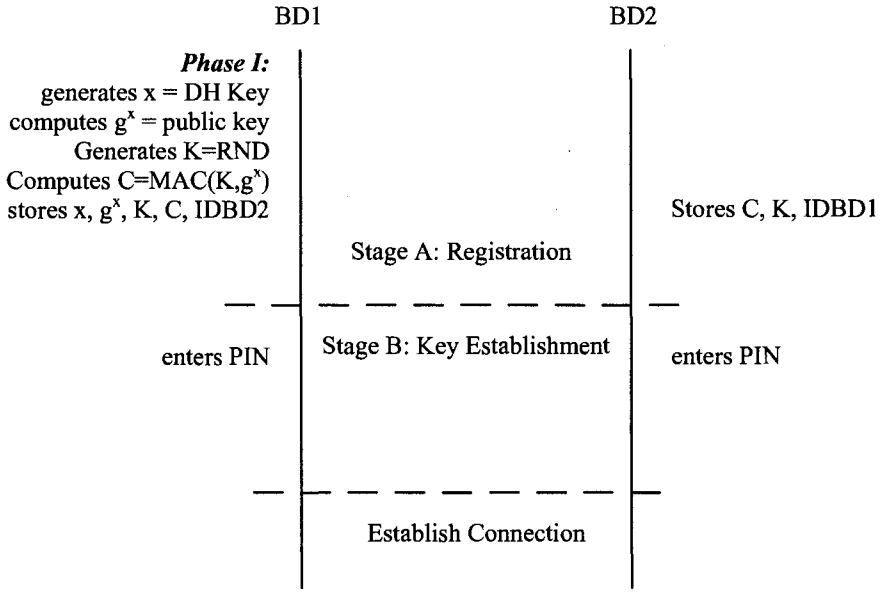

Figure 3.1: Enhanced BT Key exchange using DH Phase 1

Phase 1 will take place at Bluetooth Device 1. It starts off with Bluetooth device 1 (BD1) generating its private key x. After the generation of its private key, BD1 computes the public key g^x mod q. BD1 now needs to generate a random number for the authentication key (K) which will be used as the key to the MAC function. The next step for BD1 is to generate a message digest using a MAC function. The message digest is created by using the K as the key and the message used to create the message digest is g^x. Once all these steps are taken care of, BD1 will store x, g^x, K, C, and IDBD2. BD1 will securely transfer the values of C and K to BD2 using its online interface. User of BD2 will enter these values into its device and store them.

Prior to the beginning of phase 2 several steps need to be taken. First of all, both users for the Bluetooth devices need to agree upon a PIN. Once the PIN has been agreed upon, the users enter the PINs into their respective Bluetooth devices. Once the PINs have been verified, the connection is established. The next step is for BD2 to send its address (IDBD2) to BD1. In return, BD1 will send a message to BD2 containing its public key (g^x) and its address (IDBD1). BD2 will now try to compute the same message digest as BD1 by using the key it received earlier (K) and the g^x it was sent (C' = MAC(K,g^x)). After the computation, BD2 will compare the calculated C' to the C it has stored. If both values are the same then BD2 will send a success message to BD1.

Phase 2 can now start after the C and C' have been confirmed to be the same. At the beginning of phase 2, BD2 generates it private key, y. BD2 will now use this key to generate its public key (g^y mod q). After the calculation of the public key, BD2 will compute a key (K') by using the key derivation function (KDF). The KDF is simply a hash function which takes as input a variable size message and outputs a fixed size message. BD2 will compute K' by using the authentication key (K' = KDF(K)). BD2 will finally calculate the link key using all the values it has. The link key is also calculated by using the key derivation function (K_{Link} = KDF(PIN,S, C K, IDBD1, IDBD2)). After all these steps have been completed, BD2 sends a message to BD1 containing g^y and $E_{K'}$(K,IDBD2).

$E_{K'}(K,IDBD2)$ is simply an XOR encryption function that is encrypting K and IDBD2 using K' as the key. At the conclusion of this step phase 2 is finished.

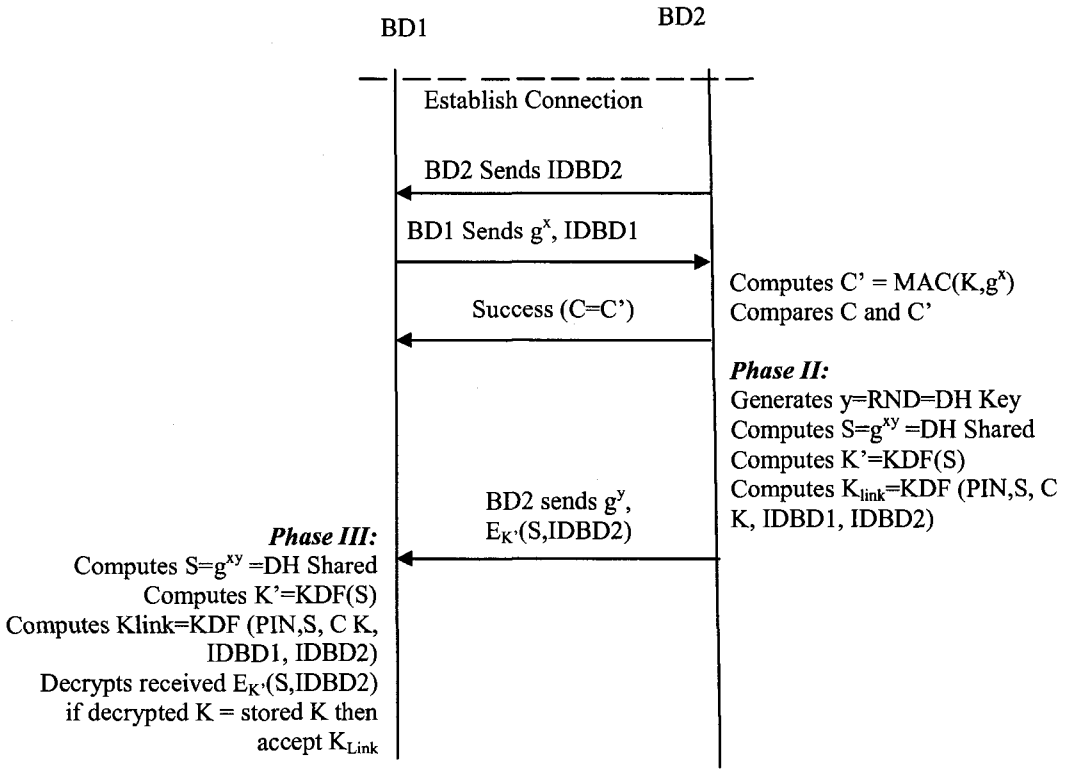

Figure 3.2: Enhanced BT Key exchange using DH Phase 2and 3

Phase 3 begins with BD1 computing K' in the same way that BD2 did (K' = KDF(K)). After the calculation of K', BD1 will generate the link key once again in the same manner as BD2 (K_{Link} = KDF(PIN,S, C K, IDBD1, IDBD2)). The final calculation for BD1 is to decrypt the message that was sent by BD2 ($E_{K'}(K,IDBD2)$) to recover the value of K. When BD1 recovers the value of K it will compare it to the stored value of K it has. If the stored value of K matches the decrypted value of K then BD1 will accept the value of K_{Link}.

3.2 Security Analysis

The Diffie-Hellman protocol was proposed in order to avoid the security risks present in the original Bluetooth key exchange scheme. As was mentioned earlier, the main security threats were passive eavesdropping, active eavesdropping, and bluedumping.

In the passive eavesdropping attack, the attacker will not be changing any messages. They will just be trying to obtain the link key by observing the messages being sent from one device to the other. In the Diffie-Hellman protocol, the only entities that are exchanged publicly are g^x, g^y, IDBD1, IDBD2, $E_{K'}$(S,IDBD2), and a success signal. The information that an attacker needs to obtain the link key is PIN, S, C K, IDBD1, and IDBD2. An eavesdropper would be able to obtain IDBD1 and IDBD2 by simply listening to the conversation. The PIN of course is again a problem. The attacker needs to be able to guess the PIN somehow because there is no real way to calculate it unless you go through all possibilities. The shared secret S is also a problem because the attacker will need to find out either user 1's or user 2's private key in order to calculate S. Since these values are generated randomly, it would be quite difficult. The values of C and K are usually exchanged between the two devices right when they discover one another and an attacker would have to be present and waiting for these devices to connect to be able to get these values. Another option would be for the user to enter the values of C and K to the second BT device using human operable interface [23]. This way it would not be possible for the hacker to obtain these values. The Diffie-Hellman enhanced protocol would completely eliminate the possibility of an off-line attack by using the Diffie-Hellman Key Exchange and Hash functions.

The active eavesdropping attack would have a greater chance of success in this protocol. However, it would be quite difficult. The main issue in the man-in-the-middle attack would be the discovery of the values of C and K and then using those values to manipulate the DH public key to match those values. Of course, first and foremost, if the attacker is unable to obtain the PIN then the attack will be useless. In the DH protocol when BD1 sends g^x to BD2, an active

eavesdropper would intercept this and for a successful attack would need to find another public key that would give the same value of C. If somehow the attacker is able to obtain the values of K and C, this would become much easier. The values of K and C are exchanged very early and again the attacker would have to be present as soon as the two devices discover one another. Another precaution would be for BD1 to send the K and C to BD2 using its output interface and the user of BD2 would enter the information into the device using the human operable interface [23]. With these precautions although it is possible for a man-in-the-middle attack to work, it is highly unlikely.

Bluedumping, as was explained earlier, is a combination of active and passive eavesdropping [21]. The main portion of this attack relies on initiating a new key exchange and then listening to the messages being exchanged. The attacker will force the two devices to discard their link keys and start a new key exchange. However, it is not necessary for the devices to change any of the values they used previously. If new values are not created for everything, then the attacker will have an even smaller chance of obtaining the link key because the K and C values do not need to be transferred from one device to the other. If new values of K and C are used again it will be quite difficult for the attacker to obtain these if they are being entered into the Bluetooth device using the human operable interface. At the end the success of the bluedumping attack depends on the same factors as the passive eavesdropping attack.

3.3 Problems in Recent Work for BT Key Exchange

Although it does provide better security than the original Bluetooth key exchange, there are some issues with this protocol that could make it undesirable for some users. As the protocol diagram shows, each user is required to carry out the same amount of computation which can be a problem when one device has a processor that is much slower or smaller than the other. For example, if a laptop is communicating with a handheld device (cell phone, PDA, etc.), this protocol could cause significant delay at the handheld device end and the laptop would just have to wait. Another issue would be the increased amount of computations,

each user is to perform two modular exponentiations during each key exchange again this would cause significant delays at the end of the handheld device. The result of these problems would be significant delay for the users with slower or less powerful processors which is quite tedious for the user that is trying to communicate with them.

3.4 Summary

The Diffie-Hellman protocol was able to address some of the issues with the original Bluetooth key exchange scheme. This protocol was able to provide high security against the passive eavesdropping, active eavesdropping, and bluedumping attacks. Although this is a very good feature, it did suffer from some undesirable features as well. The Diffie-Hellman key exchange protocol will require each Bluetooth device to perform two exponentiations, which will greatly increase the difficulty of calculations required as well as the computation time. Another issue would be that smaller and slower processors have to perform exactly the same computations as their powerful counterparts, which can cause an even longer delay. In the next chapter another algorithm is introduced called "Unbalanced RSA" which addresses the problems related to the Diffie-Hellman protocol.

4. RSA and its Modified Version

This chapter will first go over the basics of the RSA algorithm followed by an explanation of the modified version of RSA called "Unbalanced RSA". Finally the security features of both RSA and Unbalanced RSA will be reviewed.

4.1 RSA

RSA was publicly described the first time in 1977 in a paper written by Rivest, Shamir, and Adleman for *Communications of the ACM* [26]. RSA has become the most popular approach to public-key encryption. The RSA algorithm consists of five main entities shown in the table below.

Entity	Description
n	Modulus (public)
p, q	Prime factors of modulus n (private)
e	Public key (public)
d	Private Key (private)

Table 4.1: RSA entities

The RSA algorithm is shown in Figure 6.1 on the next page. The figure shows the basic idea on how RSA works. One user (User A) will select two prime numbers p and q. Using those numbers he will calculate its public and private key. User A will then broadcast its public key so User B can obtain it. User B will select a message and encrypt it using User A's public key (e) then send the cipher text to User A. User A will decrypt the ciphertext using its private key to obtain the message.

Key Generation	
Select p, q	p and q are prime
Calculate $n = p \times q$	
Calculate $\phi(n) = (p-1)(q-1)$	
Select integer e	$gcd(\phi(n),e)=1; 1<e<\phi(n)$
Calculate d	$e*d=1 \bmod \phi(n)$
Public key	$KU = \{e,n\}$
Private key	$KR = \{d, p, q\}$

Encryption	
Plaintext:	$M < n$
Ciphertext:	$C = M^e \bmod n$

Decryption	
Ciphertext	C
Plaintext:	$M = C^d \bmod n$

Figure 4.1: RSA Algorithm [27]

4.2 "Unbalanced RSA"

"Unbalanced RSA" (URSA) was introduced in a paper called "RSA for paranoids" by Adi Shamir in 1995 [28]. In 1995, a modulus size of 512 bits for RSA started to become considerably insecure. However, if the modulus size was increased some problems could arise. Firstly, by increasing the modulus size, the computation complexity and delay would rise very quickly and thus make it difficult to choose a modulus size that is efficient and will provide long term security. For example, if an RSA encryption operation with a 1024 bit modulus takes 1 second, then the same encryption operation with a 5120 bit modulus on the same processors would take approximately 2 minutes. Unbalanced RSA will

allow people to increase the modulus size to improve security without any time penalty. Unbalanced RSA can also be used to keep the modulus size the same and decrease the computation time when compared to RSA.

In the explanation of "Unbalanced RSA" a modulus size of 1024 bits will be used as an example as this value is considered a common value for RSA modulus. In normal RSA the prime factors for such a modulus would be 512 bits each. In Unbalanced RSA however, the two prime factors p and q are 256 and 768 bits in length respectively. This difference in the size of the prime factors is what is being referenced in the name "Unbalanced RSA". A later discussion will show why a 1024 bit modulus and its factors of size 256 and 768 in Unbalanced RSA are still as secure as normal RSA with its 512 bit factors.

RSA is usually used to exchange keys for symmetric cryptosystems therefore the cleartexts that are being encrypted are usually quite short. Looking at the earlier example of p which was 256 bits, it is very unlikely that someone would use RSA to exchange a key greater than 256 bits. Even three keys for 3DES require only 168 bits [28]. An assumption can be made that the cleartext that needs to be encrypted is in the range of 0 to p.

When comparing RSA to Unbalanced RSA one only needs to consider the decryption operation because the encryption operation remains the same in both cases. In normal RSA, decryption is performed by $M = C^d \bmod n$. In Unbalanced RSA, decryption is by performed $M = C^d \bmod p$. If RSA decryption and Unbalanced RSA decryption is compared it will show that the moduli are 1024 bits and 256 bits respectively. Using these two moduli it can be shown that the decryption in RSA will take $(1024/256)^3=64$ times longer than that of URSA [28].

The decryption process of Unbalanced RSA is based on the use of the Chinese Remainder Theorem (CRT). In normal RSA when trying to decrypt the ciphertext using CRT is the most efficient way to do so [29]. The process is split up to calculate $M_1=C^d \bmod p$ and $M_2 = C^d \bmod q$, where p and q are of the same size. For the URSA case where p is 256 bits and q is 768 bits, the calculation of M_2 would take $(768/256)^3 = 27$ times as long as the calculation of M_1. The purpose of this discussion was to show there is no need to calculate the much

more expensive M_2. Since the size of the message M is smaller than p, it can be concluded that M_1 is simply equal to M. This would mean that there is no need to calculate the much more expensive M_2 in order to retrieve the original message.

4.3 RSA and "Unbalanced RSA" Security Analysis

RSA security is mainly dependant on the ability of an attacker to factor the modulus n into its prime factors, p and q. These days a normal accepted value of an RSA modulus is 1024 bits making the two prime factors 512 bits each. There are two types of factoring algorithms that can be used on the modulus of RSA. The first type of algorithm is one whose running time depends on the size of the factors (Type 1) and the second type of algorithm is one whose running time depends on the size of the factored number, n (Type 2) .

The fastest factoring algorithm which depends on the size of the factors is the elliptic curve method. This method was invented by Lenstra in 1987 [30], it was an improvement on the p-1 method put forward by Pollard [31]. The asymptotic running time of this method is $\exp(O((\ln(p))^{0.5} \cdot (\ln\ln(p))^{0.5}))$ [28]. However, the basic operations of this method are very slow. In 1995 the largest factor ever found using the elliptic curve method was 145 bits long. The largest factor ever found so far using ECM was 67 digits long [32]. According to Paul Zimmerman, using the elliptic curve method should find factors 70 digits long by 2010 and 85 digits long by 2018 [33]. Therefore, in the near future, it will be very unlikely the elliptic curve method will be able to find factors 256 or 512 bits long.

Factoring algorithms that depend on the size of the factored number are much faster because they can use a variety of mathematical techniques. The best algorithm of this type is called the general number field sieve. This method consists of a sieving step and a matrix step [34]. This algorithm has an asymptotic complexity of $\exp(O((\ln(n))^{1/3} \cdot (\ln\ln(n))^{2/3}))$ and is said to be able to factor a 512 bit modulus 10,000 – 15,000 MIPS-years (1 MIPS-years is about 31.5 trillion instructions) [28].

Since RSA was introduced all major factorizations of the modulus have been achieved by the algorithms depending on the size of the modulus. It can be

assumed that this trend will continue in the future. The earlier example has Unbalanced RSA with the modulus size 1024 bits and the size of p is 256 bits, whereas RSA has n being 1024 bits and p being 512 bits. By looking at the results of these factoring algorithms it is safe to say that the sizes chosen for the URSA entities provide the same level of security as the RSA entities. Since the largest factored numbers using ECM is 67 digits long, the p value which 256 bits in "Unbalanced RSA" is just as safe as the 512 bit value of RSA. The second type of algorithm depends on the size of the modulus and since both RSA and Unbalanced RSA have a modulus size of 1024 bits, they are equally secure to this type of algorithm.

By looking at these facts one can say that the "Unbalanced RSA" algorithm can be kept as secure as regular RSA if certain conditions are met. If the modulus, n, of both algorithms is of the same size then RSA and URSA will not be susceptible to a type 2 algorithm because the size of their moduli is the same. Now since Unbalanced RSA uses the prime factor p as its decryption modulus, the size of the factor should be greater than 67 digits. This is because that largest factor found using a type 1 algorithm is 67 digits long. As long as these conditions are met, using Unbalanced RSA should not compromise the security when comparing it to normal RSA.

4.4 Application of Unbalanced RSA

Unbalanced RSA has been proposed to be applied to other wireless areas. In [35], [36], it is proposed to use Unbalanced RSA for authentication and key distribution in Wireless Local Area Networks. Public-key cryptography needs to be used in WLAN because it also suffers from the same security threats as Bluetooth (passive and active eavesdropping) [37]. The main idea of this proposal was for the client and server in the communication to exchange certificates and a shared secret key. The first step is for the client to send its certificate to the server, the server will verify the certificate authority's signature on the certificate and if approved it will move forward. The server will generate a random secret key and encrypt it with the client's public key. This ciphertext

would be sent to the client along with the server's certificate. The client will now verify the server's certificate and if approved will decrypt the ciphertext to obtain the secret key. Once both the client and server have the secret key a finished message is sent from client to server. This protocol using Unbalanced RSA addresses the flaws in key-distribution and authentication in the 802.11 standard and also reduces the time delay for key-distribution.

4.5 Summary

In this chapter the "Unbalanced RSA" algorithm was introduced to show its advantages over RSA and Diffie-Hellman. The purpose of "Unbalanced RSA" is to decrease the size of decryption modulus in order to reduce computation time, while providing the same security level. For example, it was shown that the decryption process for "Unbalanced RSA" was 64 times shorter than that of RSA when both have a modulus of 1024 bits. "Unbalanced RSA" can also be used to greatly increase the security level, by increasing the size of n, while keeping the same computation time as RSA. Another advantage of "Unbalanced RSA" is that it can be used to let slower and smaller processor do the decryption portion of the algorithm so that they don't have to deal with the large modulus size.

5. Proposed Bluetooth Key Exchange with Unbalanced RSA

In this chapter the proposed protocol will be introduced. "Unbalanced RSA" will be applied to the Bluetooth Key exchange in order to improve the security from the original Bluetooth scheme and reduce computation time and reduce the number of exponentiations from the Diffie-Hellman protocol. The first section will describe the proposed protocol in detail using its three separate phases. The next section will describe how the security features of the protocol defend against the common key exchange attacks discussed earlier. Lastly the three key exchange protocols in various fields will be compared.

5.1 The Proposed Protocol

First of all, there are two scenarios that can take place during Bluetooth communication. The first scenario is when both devices have human operable interfaces or there can be a communication between two devices one of which has no human operable interface. The device with no human operable interface should have a suitable output interface so that it can display important values to the other device. The public keys and message digest are securely transferred to each device at the very beginning of the connection between the two devices.

The proposed protocol consists of three messages being exchanged between Bluetooth Device 1 (BD1) and Bluetooth Device 2 (BD2). Other than these messages there are several calculations that take place at each device. There are some variables and abbreviations that will be used during this discussion. In Table 5.1 below are descriptions of all these items.

e_2, d_2	Public key, Private key pair for BT device 2
n	Modulus used for RSA
p, q	prime factors of modulus n
Hash	MD5 Hash function
MAC	Message authentication code. Input is secret key and message, output is MAC
RND	Random number generator

Table 5.1: proposed protocol entities

The public key of BD2 is used only by BD1. BD1 will perform an encryption function; BD2 will perform a decryption function. Both devices need to perform 2 Hash functions and 1 MAC function. Only BD1 will need to generate random numbers using the random number generator. Figure's 5.1 and 5.2 show the protocol along with all its steps divided into 3 phases.

5.1.1 Phase 1

First and foremost, the public key of BD2 is given to BD1 secretly as soon as the two devices decide to connect to one another but prior to the entry of the PIN. So for example when BD1 discovers BD2 as soon as BD1 chooses to connect to BD2 and BD2 accepts, the public key of BD2 (e and n) is sent to BD1. Phase 1 (Figure 5.1) of the protocol takes place before the users enter the PINs into both BT devices. Phase 1 mainly consists of operations/calculations that do not require values from BD2. BD1 will firstly generate a 128-bit random number (K) to use as the initialization key. This key will then be encrypted using the public key of BD2 ($C = (K^{e_2}) \bmod n$). A message digest is calculated using the initialization key (K) and the ciphertext C. This message digest will be used later for authentication. Once all these values are calculated, BD1 will store K, H, C,

and e_2. The message digest is secretly sent to the user of BD2. BD2 will store the value of H to use at a later time for authentication. Once these operations are finished both users have agreed to a PIN and enter the PIN into their respective Bluetooth enabled devices.

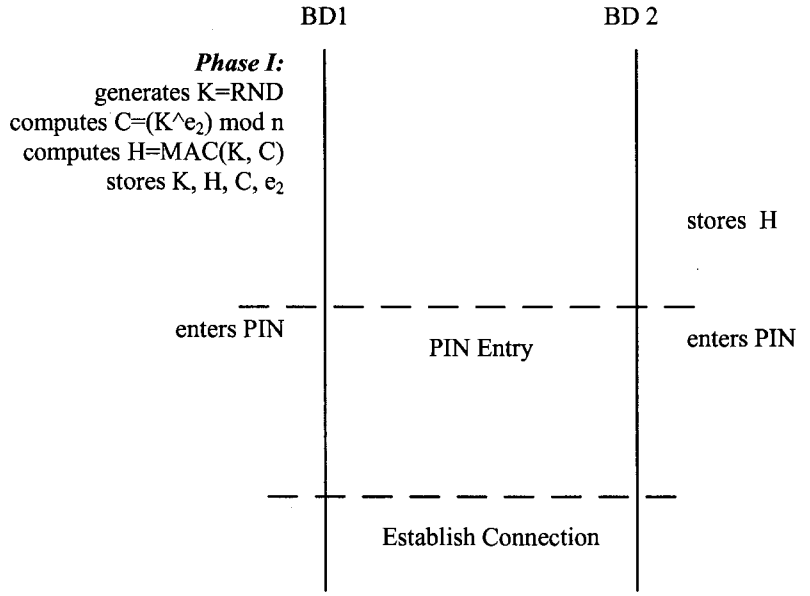

Figure 5.1: Enhanced BT Key exchange using URSA Phase 1

5.1.2 Phase 2

After the PINs have been entered, two messages need to be exchanged between the users to enable further calculations. BD2 will send its 48-bit address to BD1 and BD1 in return will send its address along with the ciphertext, C. Once these messages have been exchanged, phase 2 of the protocol can begin. Phase 2 takes place entirely on the side of BD2. Firstly, BD2 will decrypt the ciphertext to obtain the initialization key. As can be seen in Figure 5.2, the decryption process is done by using modulus p hence "Unbalanced RSA" is being used. After decrypting the ciphertext, BD2 will compute an H' using the random key K which was just decrypted and the ciphertext, which was just delivered to it following Phase 1. Once H' is calculated, BD2 will then compare this value to the H that it was given during the end of phase 1. If H and H' are the same, then BD2 knows that it is communicating with BD1 and no changes have

been made to the messages being sent. If H and H' did not match BD2 would simply shut down the communication and the key exchange would be cancelled at that point. Once the message digests match, then the initialization key is used to create another key, K'. This key is generated by using the initialization key (K) as input to the MD5 hash function. This K' is then XORed with H' and the address of BD2 and this value is sent to BD1. The last step in phase 2 is to generate the link key which will be used later to obtain an encryption key. The link key, K_{link}, is generated by entering several values into the MD5 Hash function. The inputs to the Hash function are the PIN, K, H, IDBD1, and IDBD2. As can be seen, all these values are very important to the generation of the link key and a small error in any one of them will cause the devices to have two different link keys. That is why these values all need to be checked to ensure their correctness.

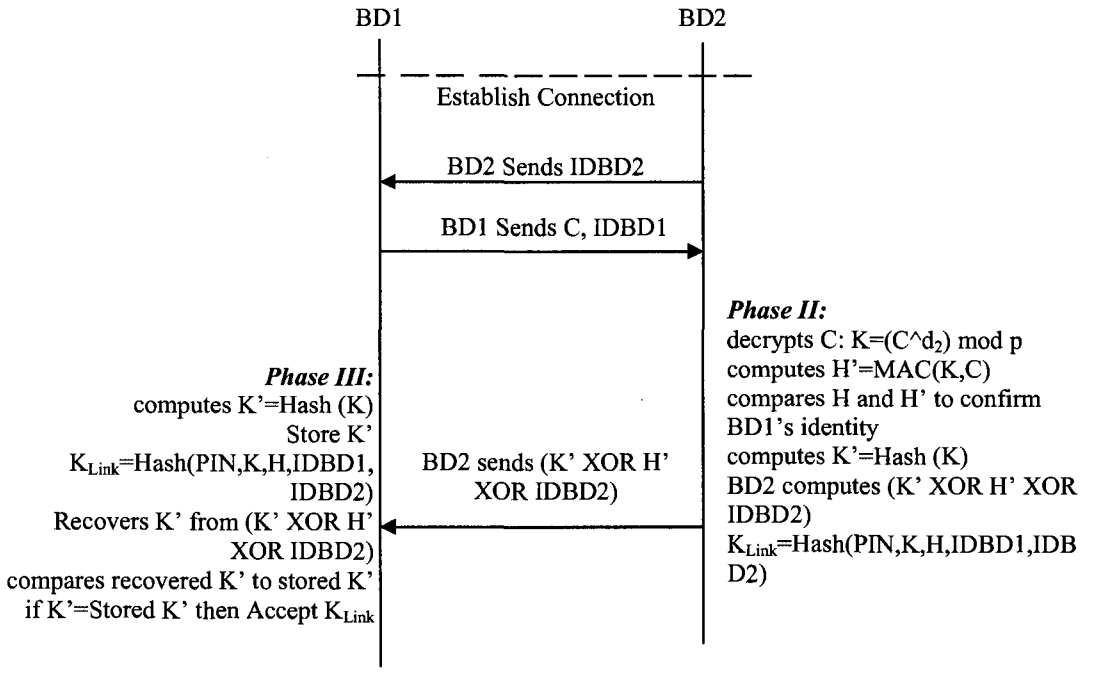

Figure 5.2: Enhanced BT Key exchange using URSA Phase 2 and 3

5.1.3 Phase 3

Phase 3 is also shown is Figure 5.2 and will begin prior to the message being sent by BD2 to BD1 containing the (K' XOR H' XOR IDBD2) value. The first step in phase 3 is to generate the same K' as was done in phase 2. This was

done by placing the initialization key (K) into the hash function. Once the K' is calculated then BD1 can go on and generate the link key. Once again, this is done in the same manner as BD2 where the values of the PIN, K, H, IDBD1, and IDBD2 are used as input to the hash function and the output is K_{link}. The next step in phase 3 is to recover the value of K' from the message that was sent by BD2 earlier. This is done by XORing the message with H and IDBD2. Once K' is recovered, BD1 will check to see if the K' that was recovered from the message is the same K' that was calculated earlier through the hash function. If they are the same then BD1 knows that BD2 received the proper value of the initialization key or else they could not calculate the same K'. After having gone through the entire protocol, both devices know they have the correct values so they can now accept the K_{link} value and continue with the security process after the key exchange.

5.2 Security analysis

The proposed protocol possesses many of the same security properties as the Diffie-Hellman protocol. The proposed protocol provides high security against the common attacks of passive eavesdropping, active eavesdropping and bluedumping. As explained earlier the proposed protocol will exchange certain values (H and public keys) secretly prior to the entry of the PIN. The combination of public-key cryptography as well as HASH/MAC functions provides enough security for the near future.

The passive eavesdropping attack would be almost useless against this protocol. As explained earlier certain values are exchanged secretly before entry of the PIN. Even if these values were to somehow get compromised a passive eavesdropping attack would still be almost impossible. In order to have a successful passive attack the attacker would need to obtain the values of H, K, PIN, IDBD1, and IDBD2. The values of IDBD1 and IDBD2 can be obtained by listening to the messages sent by BD1 and BD2 to one another. The other information an attacker would be able to gather is the value of the ciphertext, C, and the value of (K' XOR H XOR IDBD2). An attacker will not be able to obtain

the value of the initialization key K from C because it can only be properly decrypted with the private key of BD2. Since the attacker cannot obtain K it therefore cannot obtain K' because in order to get K', K is to be entered into to the Hash function. K' is needed to be able to recover the value of H from the message of (K' XOR H' XOR IDBD2). Then the attacker still has the issue of somehow finding out the PIN value. One can see that passive eavesdropper would have an almost impossible time trying to obtain these values so that he could put them into the Hash function to get the link key.

An active eavesdropping attack has a better chance of success than a passive one but still will have a difficult time. The success of the active eavesdropping attack will mainly depend on the attacker being able to get the right value for the message digest. Since the value of H and e is exchanged prior to PIN entry is extremely difficult for the attacker to retrieve these. Even if the attacker can somehow obtain these values, the attacker has to be anticipating when the devices will choose to make a connection. But since the attacker is focusing on the Bluetooth exchange it will be difficult to obtain the public keys and H. The attacker will intercept the message being sent from BD1 to BD2 which contains the value of C. Now the attacker needs to ensure that the value of C it will send to BD2 has the same MAC value as H using K as the key. The attacker will have no knowledge of the values of H or K making it very unlikely he/she can produce a C having the same message digest value. If the attacker sends an incorrect value of C to BD2 then it will cause BD2 to shut down the key exchange because it will know someone is altering the messages. If the attacker does find out the value of C then there is still the problem of finding the PIN and the initialization key. In order for an active eavesdropping attack to succeed, the attacker will need to find out all of these values, which will make it quite difficult as shown in the discussion above.

The bluedumping attack is somewhat of a combination between passive and active eavesdropping. An attacker will send a message to a Bluetooth device pretending to be another device and claiming to have lost the link key the devices shared. This will force the two devices to reinitiate the link key exchange [21].

However, once the two devices start the key exchange, the attacker becomes a passive eavesdropper and therefore can only listen to the messages being sent. Again the two devices will exchange values prior to PIN entry using their interfaces and in some cases may not need to exchange these values if they do not wish to renew them. The attacker now faces the same problems as the passive eavesdropping attack and therefore it will be highly unlikely that the attacker can obtain the link key in the end.

As one can see, the Unbalanced RSA enhanced Bluetooth key exchange protocol has not lost any security features when compared to the enhanced Diffie-Hellman Key exchange protocol. Both protocols can fend off the three main key exchange attacks by using their public-key systems and hash functions. In the simulation section, Unbalanced RSA and Diffie-Hellman will be compared to see if the proposed protocol can outperform the DH protocol in terms of computation time.

5.3 Comparing the Three Protocols

Table 5.2 compares the original Bluetooth key exchange, the Diffie-Hellman protocol and the proposed protocol ("Unbalanced RSA") in several different areas.

The first row compares the three protocols in terms of their security strength. When the description of the original key exchange scheme was given it was shown how it is quite susceptible to the common attacks of passive eavesdropping, active eavesdropping, and bluedumping [16]. On the other hand the Diffie-Hellman [23] and proposed protocols were shown to have very high security against these attacks by using one-way cryptographic functions and hash functions. Therefore, the security strength of the original scheme is low while the other two protocols have high security strength.

The entities that are kept secret by each protocol also differ. The original scheme will keep only the PIN as the secret everything else will be sent to the other device in plaintext. The Diffie-Hellman protocol will keep the PIN secret as well as the shared secret key S using Diffie-Hellman. This protocol will also keep

the message digest a secret along with the key used to create it. The "Unbalanced RSA" protocol will keep the PIN secret as well as the key exchanged using RSA, just like the Diffie-Hellman protocol the message digest and its key should also be kept secret.

	Original Scheme	Diffie -Hellman	"Unbalanced RSA"
Security Strength	Low	High	High
Secret Entities	PIN	PIN, S	PIN, K_{init}
Computation Time (delay)	Adequate	Inadequate	Adequate
Protocol Steps	3	4	3

Table 5.2: Comparing three key exchange protocols

The computation time will be related to how difficult the computations are for each protocol. The original scheme has relatively simple computations. In the original key exchange scheme only symmetric ciphers are used, more specifically SAFER+. SAFER+ was a candidate for the advanced encryption standard (AES) eventually losing out to Rjindael after three years of testing [38]. The reason for the use of SAFER+ was its practical implementation in hardware [39]. The Diffie-Hellman protocol will have a relatively high computation time because of all the modular exponentiations being done on both sides. In the case of the DH enhanced protocol it would require four modular exponentiations in the entire protocol all with the same size modulus [25]. The proposed protocol should be closer to the original scheme in terms of computation time. The encryption and decryption portion of the protocol should take the most time. Usually encryption is much faster than decryption in RSA however, with the "Unbalanced RSA" version the decryption time will be considerably smaller and should make the computation time adequate. If one of the devices in a Bluetooth communication is a much smaller processor the Diffie-Hellman protocol would cause an even larger

delay, the Unbalanced RSA protocol can cure that problem. The Diffie-Hellman protocol and the proposed protocol will be simulated for the computation time.

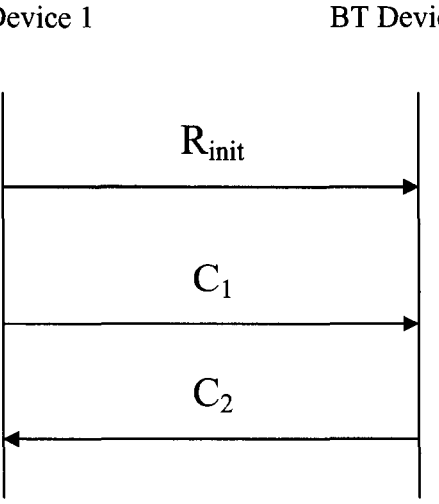

Figure 5.3: Original BT key exchange messages

The last comparison between these three protocols is made in terms of the protocol steps. In the original protocol only 3 messages are exchanged between the two Bluetooth devices during the key exchange as shown in Figure 5.2. Figure 5.3 shows the number of messages needed to be exchanged in the proposed protocol using "Unbalanced RSA". The diagram shows that the proposed protocol only uses 3 messages as well, same as the original Bluetooth key exchange scheme. The proposed protocol is therefore increasing the security from the original Bluetooth key exchange protocol without changing the number of protocol steps needed.

Figure 5.4 shows that the Diffie-Hellman protocol requires 4 messages to be exchanged, in order to complete the protocol. The Proposed protocol requires only 3 messages as can be seen in Figure 5.3 to be exchanged in order for the protocol to be completed. In the end the proposed protocol does not even increase the number of protocol steps in order to increase the security, whereas the Diffie-Hellman protocol increases the security same as the proposed protocol but needs to add one extra protocol step. Although this may not make a huge

difference in the overall design of the key exchange it does increase the complexity and will also increase the overall delay of the process.

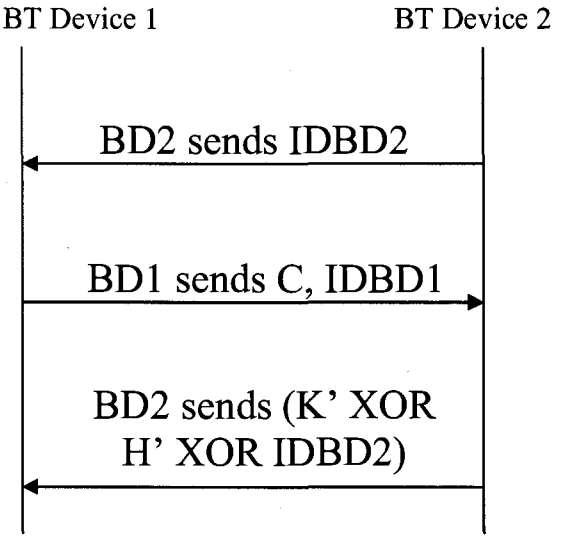

Figure 5.4: Enhanced "Unbalanced RSA" Key Exchange Messages

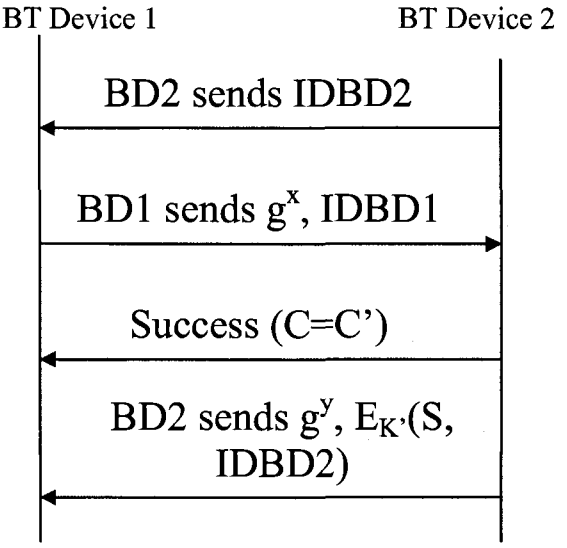

Figure 5.5: Enhanced Diffie-Hellman Key exchange messages

Comparing these three key exchange protocol in these four fields shows how successful they are. The "Unbalanced RSA" protocol is better than the Diffie-

Hellman protocol in terms computation time, and the number of protocol steps. These two protocols do however provide high security strength and have the same number of secret entities. The Unbalanced RSA protocol provides higher security strength than the original key exchange scheme and also keeps more entities a secret. They both have an adequate computation time and the same number of protocol steps. The only area where the Unbalanced RSA protocol would suffer is when it is compared to the original key exchange in terms of how complex the operations are in each protocol because the proposed protocol uses a public-key system, whereas the original scheme uses a much simpler symmetric cipher.

5.4 Summary

This chapter introduced the new protocol using the "Unbalanced RSA" Algorithm. The goal of the proposed protocol was similar to that of applying Unbalanced RSA to WLAN [35], [36], which was to improve security while trying to minimize delay. The proposed protocol provides better security features than the original scheme when related to the common key exchange attacks. The proposed protocol also improves the previously discussed Diffie-Hellman protocol by having a lower computation time. This protocol will require each user to perform only one modular exponentiation. Also the powerful processor can take care of the encryption because it has to handle the entire modulus while the other processor can take care of the decryption because it only needs to handle the prime factor, p. The next chapter will compare the Diffie-Hellman protocol, the proposed protocol, and an RSA protocol in terms of their time delay. As the complexity of the operations in the protocol and the computation time are related, some vital information should be obtained from the simulations run on these key exchange protocols.

6. Simulation Results and Analysis

The simulation was performed to check and see how long each protocol would take to go through its phases. Simulation results were performed on the Diffie-Hellman protocol, the "Unbalanced RSA" protocol, and a protocol exactly the same as the proposed protocol but using RSA. The simulation was performed for modulus sizes of 1024 bits and 2048 bits. In the URSA protocol, the size of p when n is 1024 bits was 256 bits, making q 768 bits. When n was 2048 bits, the size of p was 512 bits, making q 1536 bits. The reason for choosing these values was because as shown in [32], [33] they should be secure for the foreseeable future. The modular exponentiation was performed using the square-and-multiply method as it is one of the most popular methods to use. Simulation was performed using MAPLE version 11 on a laptop with a core duo processor with a 2GB RAM and 2GHz clock frequency. The codes for the simulations are provided in the appendices along with comments to explain how they work.

6.1 Simulation Results

The simulation results were performed in phases as shown in the diagrams for the protocols earlier. In the Diffie-Hellman protocol phase 1 consists of a modular exponentiation and a MAC function; Phase 2 consists of two modular exponentiations and two hash functions; and finally Phase 3 consists of two hash functions. In the RSA/Unbalanced RSA protocol Phase 1 consists of an encryption operation and a MAC function; Phase 2 consists of a decryption operation, a MAC function, and two hash functions; and lastly Phase 3 consists of two hash functions.

The results for the running time of these phases are shown in Table 6.1(n=1024 bits) and Table 6.2(n=2048 bits). The results show the amount of time taken to perform calculations for the entire phase and in brackets it shows how much of the time of the phase is consumed by the main operation, which in these cases was the modular exponentiation. These results will be discussed in more detail later to ensure their validity.

	Phase I (main opr.)	Phase II (main opr.)	Phase III (main opr.)
DH	3.68 ms (3.46 ms)	9.02 ms (8.76 ms)	5.3 ms (4.98 ms)
RSA	.048 ms (.045 ms)	3.75 ms (3.725 ms)	6.4 μs (6.4 μs)
URSA	.048 ms (.045 ms)	.064 ms (.0589 ms)	6.4 μs (6.4 μs)

Table 6.1: Delay for DH, RSA, and URSA schemes (n=1024 bits)

	Phase I (main opr.)	Phase II (main opr.)	Phase III (main opr.)
DH	18.6 ms (18.5 ms)	48.5 ms (48 ms)	28.2 ms (28.1 ms)
RSA	.425 ms (.421 ms)	32.9 ms (32.6 ms)	6.4 μs (6.4 μs)
URSA	.425 ms (.421 ms)	.562 ms (.552 ms)	6.4 μs (6.4 μs)

Table 6.2: Delay for DH, RSA, and URSA schemes (n=2048 bits)

By looking at the results in the two tables above, one can clearly see the advantages "Unbalanced RSA" has over the other two protocols. As explained in the chapter on "Unbalanced RSA", decreasing the size of one factor does not compromise the security of the algorithm and therefore the tables provide an accurate way to compare the protocols [28]. The calculations in each phase were explained earlier and by looking at the results one can see that Unbalanced RSA is clearly superior to the other two algorithms when it comes to computation time. The Diffie-Hellman protocol is clearly the least efficient protocol for either modulus size. The amount of time it takes in each phase is much greater than that of RSA or "Unbalanced RSA". The comparison of RSA and "Unbalanced RSA" is a little bit tricky as in both protocols; Phase 1 and Phase 3 delays are exactly the same. Phase 1 and Phase 3 is where the encryption and hash functions take place. However, when looking at phase 2, Unbalanced RSA's efficiency is much greater than that of RSA. Taking all these results into account, one can say the "Unbalanced RSA" is the best choice when one needs a combination of security and efficiency.

6.2 Analysis of Results

The simulation results can be tested by comparing them with Shamir's discovery that the time delay of a RSA computations grows cubically with the size of the modulus [28]. Table 6.3 shows the results for the encryption process and how their time delay ratio compares to that of Shamir's theory. First of all one should take note that the table shows RSA and Unbalanced RSA have the same time delay. This is not an error because as mentioned earlier, the encryption operation does not change between RSA and "Unbalanced RSA". In theory, when the size of the modulus changes from 1024 bits to 2048 bits, the time delay should be about $(2048/1024)^3=8$ times slower. As the table shows the tested time is about 9.3 times slower which is quite a good estimate when compare to Shamir's theory.

	n = 1024 bits	n = 2048 bits	Ratio (Theory)
RSA	.045 ms	.421 ms	9.3 (8)
Unbalanced RSA	.045 ms	.421 ms	9.3 (8)

Table 6.3: Encryption time delay and ratio (RSA and URSA schemes)

Table 6.4 shows the results of the decryption process and compares the results to Shamir's theory. As mentioned in the beginning of the chapter, when the modulus is 1024 bits, the p value for URSA is 256 bits, and when the modulus is 2048 bits, the p value for URSA is 512 bits. According to Shamir, when n is 1024 bits, Unbalanced RSA decryption should be $(1024/256)^3$=64 times faster than that of RSA. In the simulation, results showed Unbalanced RSA to be 63 time faster than RSA. When n is 2048 bits Unbalanced RSA decryption should also be $(2048/512)^3$=64 times faster than that of RSA. In the simulation, with modulus being 2048 bits, results showed Unbalanced RSA decryption to be 59 times faster than RSA. By looking at these results and comparing them to Shamir's theoretical results, it can be deduced that the results obtained are valid and make sense.

	n = 1024 bits	n = 2048 bits
RSA	3.725 ms	32.6 ms
Unbalanced RSA	0.0589 ms	.552 ms
Ratio: RSA/URSA (Theory)	63 (64)	59 (64)

Table 6.4: Decryption time delay and ratio (RSA and URSA schemes)

6.3 Summary

This chapter displayed and analyzed the results achieved when the Diffie-Hellman protocol, the proposed protocol, and the RSA protocol were simulated. By looking at the results it is clear to see that the proposed protocol that uses Unbalanced RSA has the lowest computation time. The differential between the proposed protocol and the Diffie-Hellman protocol is enormous whereas differential between the proposed protocol and RSA is smaller. This is due to the fact that the only difference that occurs between the proposed protocol and the RSA protocol is in Phase 2 of the protocol (decryption). Even with this fact, by looking at just the phase 2 portion of the simulation Unbalanced RSA is much better than the RSA protocol. The results were validated by showing that they adhere to Shamir's theory that the RSA computation time is proportional to the size of the modulus cubed. The results presented in this chapter were obtained on a laptop and should be considered useful for Bluetooth on laptops only. The use of the proposed protocols on handhelds would require further investigation and testing. The next chapter will state some conclusions to the thesis and also discuss some future work that could be considered for this area.

7. Conclusions

In this thesis, the problems with IEEE 802.15 standard in terms of key exchange have been reviewed. New suggestions to the key exchange scheme and their problems have also been reviewed. The problem of exchanging keys reliably in a Bluetooth network is very critical. The IEEE standard key exchange tends to have several security issues when it comes to common attacks (passive/active eavesdropping, bluedumping). The Enhancement to the key exchange which uses Diffie-Hellman [23] aimed to help fix these security issues. The Enhancement using Diffie-Hellman however also had its own problems with computation time and complex calculations.

In order to find a middle ground between security and efficiency between two Bluetooth devices, a new protocol is proposed here for Bluetooth key exchange. The proposed protocol uses a modified version of RSA called "Unbalanced RSA". If Unbalanced RSA is used, the goal of providing security and efficiency can be achieved. By using Unbalanced RSA the computation time can be reduced because in this algorithm the decryption process is done modulo p, not modulo n as in regular RSA. This fact alone should greatly improve its efficiency over the Diffie-Hellman protocol. The changing of the modulus for decryption can affect the security level if one is not careful. There are certain conditions that need to be met in order for Unbalanced RSA to have the same security level as regular RSA.

Even though Unbalanced RSA reduces the size of the prime factor p and uses it as the modulus for decryption, this does not decrease the security level when compared to normal RSA. A modulus can be divided up into its factors using two different types of algorithms. The first is dependent upon the size of the

factored number n and the other is dependent upon the size of the factors. Since the size of n does not change from RSA to Unbalanced RSA, the first type of algorithm will not make a difference. The second type of algorithm has only found factors up to 67 digits so far [32] and is estimated to find 85 digits in the year 2018 [33]. Looking at these facts it is safe to assume that a p size of 256 bits is still safe for a few years and if needed can even be increased slightly when it is needed.

By using detailed descriptions and simulations, it was shown that the proposed protocol using Unbalanced RSA is the best combination of security and efficiency out of the three protocols. It was shown that the original Bluetooth key exchange was susceptible to the common key exchange attacks whereas the two modified version could defend against these attacks. So the original scheme was deemed to be insecure when compared to the enhanced schemes. The Diffie-Hellman enhanced protocol and the Unbalanced RSA enhanced protocol were simulated to check their computation time. At the end, the Unbalanced RSA protocol was better in terms of computation time than the Diffie-Hellman protocol.

In summary, the proposed key exchange protocol using Unbalanced RSA was able to increase the security strength when compared to the original Bluetooth key exchange. It was also able to decrease the computation time when compared to the enhanced Diffie-Hellman Key exchange. The use of cryptographic one-way functions and hash functions allows the Unbalanced RSA key exchange to provide high security against the common key exchange attacks. The Diffie-Hellman enhanced protocol and the proposed protocol using Unbalanced RSA are quite similar in terms of their layout. However, a closer look reveals that the DH enhanced protocol requires twice as many modular exponentiations than the proposed protocol. Also, one can see the Unbalanced RSA enhancement keeps the same number of protocol steps as the original key exchange whereas the DH enhancement increases the number of protocol steps.

The proposed protocol turns out to be a suitable replacement to the original key exchange scheme and to the Diffie-Hellman enhanced key exchange protocol. The proposed protocol did perform up to the expectations on the laptop

on which it was simulated. The results obtained from the simulation are relevant to laptops and cannot easily be converted to results on handheld processors. It would be a good idea to implement this proposed protocol on different types of processors so that it can be seen if this protocol as efficient for other types of processors. It would be wise to include various types of commercially available processors for handheld devices as well as other devices, as a greater number of handhelds come equipped with Bluetooth. The use of elliptic curve cryptography in Bluetooth security should also be explored. These suggestions can be treated as future work after this thesis.

Appendix A: Maple code for BT Key Exchange using URSA

A.1: System Setup

```
M1 := (rand(2^256))();      // generates random integer of size 256 bits
M2 := (rand(2^768))();      // generates random integer of size 768 bits
P := nextprime(M1);         //finds next prime number after M1
Q := nextprime(M2);         // finds next prime number after M2
n := P*Q;                   //Multiplies P and Q to obtain modulus n
n2 := (P-1)*(Q-1);          //generates the toitent function
e := 2^16 + 1;              // sets public key value
isprime(e);                 // check to make sure e is prime
gcd(e,n2);                  //check to see if e is relatively prime to n2
d := eval(1/e mod n2);      //evaluate value of d using modular arithmetic
IDBD1 := (rand(2^48))();    // address for Bluetooth device 1
IDBD2 := (rand(2^48))();    // address for Bluetooth device 2
```

A.2: Phase 1 Testing

```
st := time():
for i from 0 by 1 to 100 do
K := (rand(2^128))();
C := K&^e mod n;
end do;
time()-st
```

This loop will see how long it takes to run phase 1 of the proposed scheme 100 times. He result obtained from this must be divided by 100 to get the actual result to run phase 1 once. Phase 1 includes 1 exponentiation and 2 random number generations.

A.3: Phase 2 Testing

```
st := time():
with(StringTools):
for i from 0 by 1 to 75 do
C := K&^e mod p;
K' = Hash(K);
K_Link = Hash(PIN,K,H,IDBD1,IDBD2);
end do;
time()-st
```

This loop will see how long it takes to run phase 2 of the proposed scheme 75 times. Since the decryption process will take longer than the encryption we used a smaller loop. Phase 2 includes 1 exponentiation and 2 hash functions.

A.4: Phase 3 Testing

```
st := time():
with(StringTools):
for i from 0 by 1 to 1000 do
K' := Hash(K);
K_Link := Hash(PIN,K,H,IDBD1,IDBD2);
end do;
time()-st
```

This loop will see how long it takes to run phase 3 of the proposed scheme 1000 times. We needed to greatly increase the size of the loop because the hash functions take very little time to execute. Phase 3 includes 2 hash functions.

Appendix B: Maple code for BT Key Exchange using RSA

B.1: System Setup

```
M1 := (rand(2^512))();      // generates random integer of size 256 bits
M2 := (rand(2^512))();      // generates random integer of size 768 bits
P := nextprime(M1);         //finds next prime number after M1
Q := nextprime(M2);         // finds next prime number after M2
n := P*Q;                   //Multiplies P and Q to obtain modulus n
n2 := (P-1)*(Q-1);          //generates the toitent function
e := 2^16 + 1;              // sets public key value
isprime(e);                 // check to make sure e is prime
gcd(e,n2);                  //check to see if e is relatively prime to n2
d := eval(1/e mod n2);      //evaluate value of d using modular arithmetic
IDBD1 := (rand(2^48))();    // address for Bluetooth device 1
IDBD2 := (rand(2^48))();    // address for Bluetooth device 2
```

B.2: Phase 1 Testing

```
st := time():
for i from 0 by 1 to 100 do
K := (rand(2^128))();
C := K&^e mod n;
end do;
time()-st
```

This loop will see how long it takes to run phase 1 of the RSA key exchange scheme 100 times. The result obtained from this must be divided by 100 to get the actual result to run phase 1 once. Phase 1 includes 1 exponentiation and 2 random number generations.

B.3: Phase 2 Testing

```
st := time():
with(StringTools):
for i from 0 by 1 to 20 do
K := C&^d mod n;
K' := Hash(K);
K_Link := Hash(PIN,K,H,IDBD1,IDBD2);
end do;
time()-st
```

This loop will see how long it takes to run phase 2 of the RSA Key exchange scheme 20 times. Since the decryption process will take longer than the encryption we used a smaller loop. Phase 2 includes 1 exponentiation and 2 hash functions.

B.4: Phase 3 Testing

```
st := time():
with(StringTools):
for i from 0 by 1 to 1000 do
K' := Hash(K);
K_Link:= Hash(PIN,K,H,IDBD1,IDBD2);
end do;
time()-st
```

This loop will see how long it takes to run phase 3 of the RSA key exchange scheme 1000 times. We needed to greatly increase the size of the loop because the hash functions take very little time to execute. Phase 3 includes 2 hash functions.

Appendix C: Maple code for BT Key Exchange using DH

C.1: System Setup

```
q1 := (rand(2^1024))();                          //generates random 1024 bit number
q := nextprime(q1)                               // next prime number after q1 for
                                                 modulus
XA := (rand(2^512))();                           //generates user A private key
XB := (rand(2^512))();                           //generates user B private key
IDBD1 := (rand(2^48))();                         //User A address
IDBD2 := (rand(2^48))();                         //User B address

for g from 50 to 1000 while α <> q-1 do          //selects range from which to get g
with(numtheory);
i := order(g, q);                                //finds the proper g as primitive root
g;                                               //displays alpha value
end do;
```

C.2: Phase 1 Testing

```
st := time():
for i from 0 by 1 to 100 do
XA := (rand(2^512))();
YA := ALPHA&^XA mod q;
K := (rand(2^128))();
end do;
time()-st
```

This code will run Phase 1 of the enhanced security using DH. This loop calculates the time needed to run this phase 100 times. Phase 1 includes 2 random number generations and 1 exponentiation.

C.3: Phase 2 Testing

```
st := time():
with(StringTools):
for i from 0 by 1 to 20 do
YA := (rand(2^512))();
YB := ALPHA&^XB mod q;
S := YB&^XA mod q;
K' = Hash(S);
K_Link := Hash(PIN,K,H,S,IDBD1,IDBD2);
end do;
time()-st
```

This code will run Phase 2 of the enhanced security using DH. This loop calculates the time needed to run this phase 20 times. Phase 2 includes 2 exponentiations, 1 random number generation and 2 hash functions.

C.4: Phase 3 Testing

```
st := time():
with(StringTools):
for i from 0 by 1 to 100 do
YA := (rand(2^512))();
S := YA&^XB mod q;
K' = Hash(S);
K_Link := Hash(PIN,K,H,S,IDBD1,IDBD2);
end do;
time()-st
```

This code will run Phase 3 of the enhanced security using DH. This loop calculates the time needed to run this phase 100 times. Phase 3 includes 1 exponentiation and 2 Hash functions.

References

[1] M. Krolak and M. Novak, "An Introduction to Infrared Technology: Applications in the Home, Classroom, Workplace, and Beyond ...," Technical Report, Trace R&D Center, University of Wisconsin, Madison, WI, 1995.

[2] M. Bialoglowy, "Bluetooth Security Review, Part 2," SecurityFocus, May 2005, Web publication available at: http://www.securityfocus.com/infocus/1836.

[3] A. Edlund, "Bluetooth Wireless Technology – 2005 Update," in Business Briefing: Wireless Technology 2005, 2005, pp. 28-30.

[4] "*Bluetooth* Wireless Technology Becoming Standard in Cars", Technical Report, Bluetooth Special Interest Group, February 2006, Web Publication available at: http://www.bluetooth.com/Bluetooth/Press/SIG/Test_1.htm.

[5] T. Olzak, "Secure your Bluetooth wireless networks and protect your data," White Paper, TechRepublic: A Resource for IT professionals, December 2007.

[6] P. Suri and S. Rani, "Bluetooth Security – Need to Increase the Efficiency in Pairing," in *IEEE Souteast Con*, 2008, pp. 607-609.

[7] C. Schwaderer, "History, technology, and product for Bluetooth," Technical Report, CompactPCI Systems, March 2001.

[8] P. Wells, "What is Bluetooth?," in *IEEE Potentials*, Volume 23, Issue 5, December 2004, pp. 33-35.

[9] "Bluetooth Range in Relation to Different Power Classes," Technical Report, Palo Wireless: Bluetooth Resource Center, March 2000, Web publication available at: http://www.palowireless.com/INFOTOOTH/knowbase/general/10.asp.

[10] J. Kahn and J. Barry, "Wireless Infrared Communications," in *Proceedings of the IEEE*, Volume 85, Issue 2, February 1997, pp. 265-298.

[11] M. Mahn, "Bluetooth and Wi-Fi," Technical Report, Socket Communication Inc., March 2002.

[12] D. Dagon, T. Martin, and T. Starner, "Mobile Phones as Computing Devices: The Viruses are coming!," in *IEEE Pervasive Computing*, Volume 3, Oct-Dec 2004, Issue 4, pp. 11-15.

[13] "Bluetooth Security," 2008, Bluetomorrow: The comprehensive guide to everything Bluetooth related, Web Publication available at: http://www.bluetomorrow.com/content/section/177/281/.

[14] IEEE 802.15, IEEE standard for Wireless Personal Area Networks (WPAN), IEEE, 2005.

[15] G. Legg, "The Bluejacking, Bluesnarfing, Bluebugging Blues: Bluetooth Faces Perception of Vulnerability," *Techonline*, April 2005, Web publication available at: http://www.wirelessnetdesignline.com/showArticle.jhtml?articleID=192200279.

[16] C. Hager and S. Midkiff, "An analysis of Bluetooth security Vulnerabilities," in *IEEE Wireless communication and networking*, Volume 3, March 2003, pp. 1825-1831.

[17] M. Jakobsson, S. Wetzel: "Security Weaknesses in Bluetooth," in *Proceedings of the RSA Conference 2001,* San Francisco, USA, April 8-12 2001.

[18] T. Karygiannis and L. Owens, " Wireless Network Security, 802.11, Bluetooth and handheld devices," NIST special publication 800-48, November 2002.

[19] Vainio, Juha, "Bluetooth Security", May 2000, Technical Report available at: http://www.niksula.cs.hut.fi/~jiitv/bluesec.html.

[20] D. Kügler, "Preventing Tracking and 'Man in the Middle' Attacks on Bluetooth Devices," Proceedings of Financial Cryptography, 2003, pp. 149-161.

[21] V. Kostakos, "WiFi and Bluetooth Vulnerabilities," Cityware: Urban Design and Pervasive systems, January 2008.

[22] Y. Shaked and A. Wool, "Cracking the Bluetooth PIN," School of Electrical Engineering Systems, May 2005, Technical Report available at: http://www.eng.tau.ac.il/~yash/shaked-wool-mobisys05/.

[23] S. Aissi, C. Gehrmann, K. Nyberg, "Proposal for Enhancing Bluetooth Security using an Improved Pairing mechanism," *in Bluetooth All-Hands Meeting*, April 19-23, 2004.

[24] W. Diffie and M. Hellman, "New Directions in Cryptography," in *IEEE Transactions on Information Theory*, Volume 22, Issue 6, November 1976, pp. 644-654.

[25] W. Stalling, *Network Security Essentials*, New Jersey: Prentice Hall, 2002.

[26] S. Robinson, "Still Guarding Secrets after Years of Attacks, RSA Earns Accolades for its Founders," in *SIAM News*, Volume 36, Number 5, June 2003.

[27] W. Stallings, *Cryptography and Network Security: Principles and Practice Third Edition*, New Jersey: Prentice Hall, 2003.

[28] A. Shamir, "RSA for Paranoids," RSA Laboratories' Cryptobytes, Volume 1, Number 3, Autumn 1995.

[29] A.V. Sarad, "Application to Chinese Remainder Theorem," Technical Report, AU-KBC Research Center, 2005.

[30] H. Lenstra, "Factoring integers with elliptic curves," *Annals of Mathematics*, Vol. 126, pp. 649 - 673, 1987.

[31] P. Montgomery, "Speeding the Pollard and elliptic curve methods of factorization," *Mathematics of Computation*, 48(177), pp. 243 - 264, 1987.

[32] P. Zimmerman, "50 Largest factors found by ECM," Inria Lorraine, Nancy, France, 2006, Web Publication available at: http://www.loria.fr/~zimmerma/records/top50.html.

[33] P. Zimmerman, "The Elliptic Curve Method," Inria Lorraine, Nancy, France, April 2003, Web Publication available at: http://www.loria.fr/~zimmerma/papers/ecm-entry.pdf.

[34] G. Meulenaer, F. Gosset, G. Meurice de Dormale, J. Quisquater, "Integer Factorization Based on Elliptic Curve Method: Towards Better Exploitation of Reconfigurable Hardware," in *IEEE Symposium on Field-Programmable Custom Computing Machines,* 23-25 April 2007, pp. 197-206.

[35] Z. Zheng, K. Tepe, H. Wu, "Applying Unbalanced RSA to Authentication and Key Distribution in 802.11", CWIT 2005, June 5-8th, 2005, Montreal, Quebec.

[36] Z. Zheng, "A New Protocol with Unbalanced RSA for Authentication and Key Distribution in WLAN," M.S. thesis, University of Windsor, Windsor, On, Canada, 2004.

[37] Z. Longjun, H. Wei, Z. Dong, and C. Kefei, "A Security Solution of WLAN on Public Key Cryptosystem," in *11th International Conference on Parallel and Distributed Systems*, Volume 2, 20-22 July 2005, pp. 422-426.

[38] C. Sanchez-Avila & R. Sanchez-Reillo, "The Rijndael Block Cipher (AES Proposal): A Comparison with DES," in *IEEE 35th International Carnahan Conference on Security Technology*, 8-10 April 2002, pp. 229-234.

[39] B. Schneier, "Performance comparison of the AES Submissions," Technical Report, Counter Pane Internet Security Inc., August 2000.

VITA AUCTORIS

Saif Rahman was born in 1983 in Doha, Qatar. He graduated from the American School of Doha High School in 2001. From there he went on to the University of Windsor where he obtained a B.App.Sc degree in Electrical Engineering in 2005. He is currently a candidate for a Master's degree in Electrical Engineering at the University of Windsor and hopes to graduate in the summer of 2008.