2009-01-01

An Enhanced Method For The Existing Bluetooth Pairing Protocol To Avoid Impersonation Attacks

Patricia Aidee Mendoza
University of Texas at El Paso, pamendoza@gmail.com

AN ENHANCED METHOD FOR THE EXISTING BLUETOOTH PAIRING PROTOCOL TO AVOID IMPERSONATION ATTACKS

PATRICIA A. MENDOZA

Department of Electrical & Computer Engineering

APPROVED:

Virgilio Gonzalez, Ph.D., Chair

David H. Williams, Ph.D.

Heidi A. Taboada-Jimenez, Ph.D.

Patricia D. Witherspoon, Ph.D.
Dean of the Graduate School

To my husband and son, Jose Luis Saenz Ibave and Jose Luis Saenz Mendoza.
To my parents, Martha Sigala and Miguel Mendoza.

AN ENHANCED METHOD FOR THE EXISTING BLUETOOTH PAIRING PROTOCOL TO AVOID IMPERSONATION ATTACKS

by

PATRICIA A. MENDOZA, BSEE.

THESIS

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE

Department of Electrical & Computer Engineering

THE UNIVERSITY OF TEXAS AT EL PASO

August 2009

Acknowledgements

I want to express my gratitude and appreciation to my Thesis Advisor, Dr. Virgilio Gonzalez-Lozano, for his guidance, enthusiasm, encouragement, and in particular, for all his patience during the completion of my research. This work would not have been possible without his knowledge, persistence, and support, and in particular because he was always willing to help me.

I would like to thank my committee members, Dr. David Williams and Dr. Heidi Taboada-Jimenez, for taking a moment of their valuable time in order to review my thesis.

I also would like to express my deep indebtedness to Dr. Patricia Nava and Dr. Benjamin Flores for all their support during the completion of my Bachelor's and Master's degrees, and moreover because they always inspired me with all their perseverance, dedication, enthusiasm and knowledge.

I want to extend my sincere gratitude to Mr. Jerald West for always believing in me, for sharing their knowledge and job experience, for his support all the way through my undergraduate studies, and specially, for being my friend.

I also want to thank my friend Sukie Quezada, for always helping me to obtain the resources needed to complete my degree.

I am grateful to my parents, husband and sister, for their love, patience, and valuable support and motivation during my studies. Without their presence in my life, it could not be possible for me to reach my goals.

Thanks to my son, that without even knowing it, he made me understand the importance of caring, sharing and spending quality time with my loved ones. He showed me the real value of pursuing my goals and dreams, gave a real meaning to my life and made me distinguish from the things in life that are really important so I could truly appreciate them.

I would like to show my deepest gratefulness to my classmates and friends, especially to Perla P. Gonzalez, Julio C. Saenz and Sergio Velazquez, because they were always with me

through all the difficulties, challenges, achievements and celebrations found in our career; Thanks for all your help and friendship.

I want to express my gratitude to my Aunt Maria Blanca Sanchez, for trusting and believing in me, and for all the financial support she provided to me while I was pursuing my undergraduate studies, because without this help it could not be possible for me to complete to my Master's degree.

And last but not least, to God, for giving the opportunity to live and pursue my dreams and goals, for giving me health and aptitude to complete my studies, and especially, for all the blessings I have received through all my life.

This thesis was submitted and defended to my Supervising Committee on July 17th, 2009 at the Electrical and Computer Department, University of Texas-El Paso.

Abstract

As the use of the Bluetooth technology in wireless technologies such as computers, cell phones, personal digital assistants (PDAs), printers, digital cameras, PC peripherals, and headphones, among others, has enormously increased during this decade, it has also increased the need of employing a more secure and robust pairing protocol that protects the transfer of information between Bluetooth devices.

The existing Bluetooth pairing protocol starts when a Bluetooth device is set to be in discoverable mode and it is found by another device. After this step is completed, it initiates the pairing procedure by sharing some important device information, then by connecting to the other device and exchanging several link keys, and finally, by authenticating the connecting device in order to make sure is a trusted peer. However, through this document it is shown that this existing Bluetooth pairing protocol has still some security flaws that have caused impersonation attacks.

The impersonation attack is a big concern in Bluetooth security due that, in most of the cases, it directly allows the access to relevant information, such as private, confidential or even classified, and this attack is reached just by creating a connection which is fully formed from a Bluetooth pairing with a false device credentials.

Due to the concern for this type of attack, this document focuses on the implementation of an additional step on the existing Bluetooth pairing and authentication process, which enhances the Bluetooth security and also avoids the impersonation attack. This additional step was added just after the final authentication stage that a Bluetooth device commonly performs, and it is based on the use of Digital Certificates and a Certificate Authority in order to ensure that the identity of the connecting device can be trusted. Moreover, this additional authentication method is also based on three security levels, which are divided by the type of device to be connected, the application of this Bluetooth device, and the capabilities to connect to internet in order to verify the Digital Certificate that is going to be used for authentication.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

The explosive growth of the Internet and consumer demand for mobility has also increased the need for wireless communications and network technologies [1]. Every day, more users ask for remote access of services and information without the use of cables or any other similar physical medium.

Bluetooth is a solution for wireless technology for short-range communication systems which is mainly intended to replace cables while connecting portable and/or fixed electronic devices [2]. This modern type of technology is intended for fast communications that allow connecting devices such as cell phones, PDAs, desktops, laptops, printers, digital cameras, etc., and communicating to each other in a small environment such as an office or home without the use of cables or any other physical medium. Bluetooth wireless technology is specifically developed for Personal Area Networks and short range communication [3]. However, this increased amount of access and mobility into Bluetooth services also increases the risk for security threats

Nowadays, Bluetooth technology is not only used in basic applications such as the ones for connecting a cell phone with a head set. Bluetooth is also employed for relevant applications, such at universities, banking offices, government organizations and private businesses. However, due to these applications, Bluetooth is a technology that requires of robust security mechanisms in order to protect information which in many cases can be relevant and confidential. Unfortunately, in spite of the many efforts realized by Bluetooth organization and Special Interest Group (SIG), Bluetooth has still many security flaws which put in danger the safety of information and communications.

For this reason, this thesis document proposes an enhanced protocol mechanism which is intended to defend the security of Bluetooth devices, and in the same way, protect information that can be exposed while connecting with other devices.

The objective of this research is to provide the Bluetooth technology with an enhanced authentication protocol which can protect Bluetooth devices from impersonation attacks. This proposed authentication method is proposed to be performed as an additional step of the current Bluetooth pairing and authentication protocols. This enhanced authentication will require the use of a digital certificate and a Certificate Authority who can prove that this certificate is a trusted one, and it will be also based in three security levels depending on the profile the device is running.

This thesis document is organized in various chapters that explain in detail the Bluetooth technology, its definition and architecture, its security mechanisms such as pairing and authentication, its security flaws, and proposes a security method. Basically Chapter one, is an introduction to the Bluetooth technology and focuses its attention in giving a depth Bluetooth background, explaining in detail its characteristics and architecture. Chapter 2 is a Bluetooth Security overview which gives a general background of the security problem that Bluetooth currently has, and also explains some security solutions that other researchers have proposed for security flaws. Chapter 3 describes in detail the causes of the impersonation attack that this research is trying to eliminate. Chapter 4 and 5 give an comprehensive explanation of the original Bluetooth pairing and authentication, explaining the use of communication keys and how these are generated. Chapter 6 deals with the analysis and description of the proposed method in order to avoid impersonation attacks and all the definitions of the parameters used in order to succeed in this proposal. All the simulations created and used to show the proposed security method are covered in Chapter 7. And Finally, Chapter 8 is a summary of the conclusions obtained through this document development.

1.1 BLUETOOTH OVERVIEW

As already stated, Bluetooth is a short-range type of wireless communications technology which employs a radio standard technology primarily designed for low power consumption with short-range and low-cost transceivers in each device. This type of radio technology, formally

called Frequency-Hopping Spread Spectrum (FHSS), uses 79 different radio channels and constantly changes frequencies at a rate of 1,600 hops per second. In its basic mode, Bluetooth communication has a Gaussian Frequency Shift Keying (GFSK) modulation [1].

Bluetooth transceivers operate in a license-free industrial, scientific and medical (ISM) radio band at 2.45 GHz of frequency, which is similar to the band WLAN devices and IEEE 802.11 compliant devices occupy [4]. This band is available for free unlicensed used in most of the world, although some countries have restrictions on which parts of the band may be used [5].

Bluetooth allows devices to communicate to each other when they come in a permitted range, which is between 1 meter and up to 100 meters, depending on the power classification of the device (more power, longer range). The theoretical maximum bandwidth of a Bluetooth network is of about 1 Mbps [6]. Table 1.1 shows a summary of some of the characteristics of the Bluetooth technology.

Table 1.1.1: Bluetooth Technology - Key Characteristics.

Characteristic	***Description***
Physical Layer	*Frequency Hoping Spread Spectrum (FHSS)*
Frequency Band	*2.4-2.4835 GHz (ISM Band)*
Hop Frequency	*1,600 hops/sec.*
Data rate	*1 Mbps.*
Operating Range	*About 10 meters (30 feet); can be extended to 100meters.*

1.2 BLUETOOTH HISTORY

In early 1994, a team of researchers was investigating the possibility of developing a wireless connection between an earpiece and phone. As the study development proceeded, Ericsson (now Sony Ericsson) decided to further research the technology and there was when the Bluetooth concept was born. Ericsson realized the need of a standard without associated

licensing fees, where one of the greatest challenges was the interoperability between different vendors [7].

Bluetooth was officially announced in May 1998 by the Bluetooth Special Interest Group (SIG), and finally launched on July 1st, 1999. The name "Bluetooth" comes in honor of the 10th century king of Denmark, King Harold Bluetooth, who engaged in diplomacy, which led warring parties to negotiate with each other, same philosophy followed in order to design Bluetooth, also known as IEEE 802.15.1 [7].

Since its original foundation, the SIG has transitioned into a non-profit trade association. SIG membership is open to all companies wishing to develop Bluetooth market and to promote these technology products [8].

1.3 BLUETOOTH PROTOCOL ARCHITECTURE

Bluetooth internal system is defined by a layer protocol architecture where every layer or protocol ensures a smooth interoperability between the Bluetooth applications and hardware. This protocol architecture consists of a Radio Layer, a Baseband Layer, a Link Manager Protocol, a Host Controller Interface, a Logical Link Control and Adaptation Protocol, and a RFCOMM [9]. Figure 1.3 shows how these protocol layers are organized.

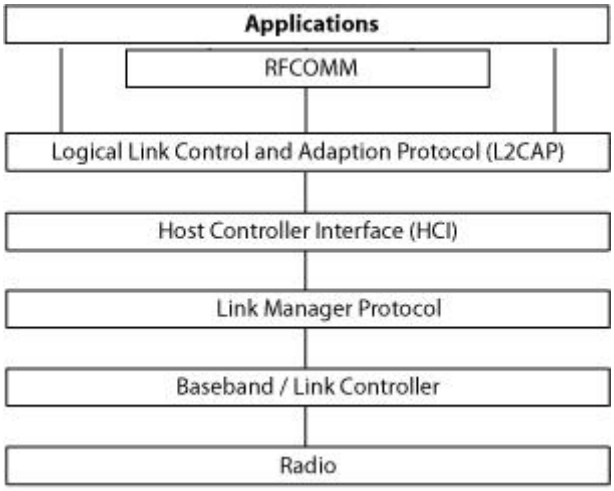


Figure 1.3.1: Bluetooth Stack Protocol

1.3.1 Radio Layer

The Radio layer is the one that is found first at the bottom level of the Bluetooth protocol stack. This layer defines the requirements of the transceiver device operating in the 2.4 GHz ISM band, establishes the requirements for using the Frequency-Hopping Spread-Spectrum (FHSS), and classifies the Bluetooth devices in three classes [8].

- Power Class 1: is designed for long range devices, up to 100 meters, with a max output power of 20 dBm and 100 mW.
- Power Class 2: for ordinary range devices, up to 10 meters, with a max output power of 4 dBm and 2.5 mW.
- Power Class 3: for short range devices, from 1/10 meters to 10 meters, with a max output power of 0 dBm and 1mW.

1.3.2 Baseband Layer

The following Bluetooth Protocol Stack level is the Baseband Layer, which is basically the physical layer of the Bluetooth. It manages physical channels and links from other services like error correction, data whitening, hop selection and Bluetooth security. This layer is used as a link controller, and works with the link manager to control routines such as creating link connections with other Bluetooth devices and power control. The baseband also manages asynchronous and synchronous links, handles packets and does paging and inquiry to access Bluetooth devices in the area [10].

1.3.3 Link Manager Protocol

The Link Manager Protocol carries out link setup, authentication, link configuration and other protocols. It is in charge of discovering other Bluetooth devices within the range area, and communicates with them via the Link Manager Protocol [10].

1.3.4 Host Controller Interface

Above the Link Manager Protocol level is the Host Controller Interface (HCI). The HCI provides a command interface to the baseband controller and link manager, and access to

hardware status and control registers [10]. This HCI level is made of three sections with different roles to play:

- HCI firmware, which is part of the actual Bluetooth hardware,
- HCI driver, which is found in the software of the Bluetooth device, and
- HC Transport Layer, which connects the firmware to the driver.

1.3.5 Logical Link Control and Adaptation Protocol

Above HCI is the Logical Link Control and Adaptation Protocol (L2CAP), which provides connection-oriented and connectionless data services to upper layer protocol. L2CAP permits higher level protocols and applications to transmit and receive L2CAP data packets. It supports higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information [9].

1.3.6 RFCOMM

The RFCOMM protocol provides emulation of serial ports over the L2CAP protocol. It is what actually makes upper layer protocols think they are communicating over a RS232 wired serial interface, so there is no need for applications to know anything about Bluetooth [9].

1.3.7 Service discovery protocol

Service Discovery Protocol (SDP) supports search of service by service class, search of services by service attributes, and service browsing, which means to extensively look at everything supported on the device.

The Service Discovery Protocol (SDP) allows a device to retrieve information on services offered by a neighboring device. A basic data connection must be set up before Service Discovery can be used. Once the connection to service discovery is established, requests for information can be transmitted, and responses received back containing information on services. This information is known as the service's attributes [9].

1.4 BLUETOOTH PROFILES

Several companies are presently manufacturing Bluetooth devices. Although the Bluetooth specification provides guidelines to ensure that the manufactures adhere to universal product standards, the interoperability between the devices is addressed differently [11]. The interoperability between devices manufactured by different vendors depends upon the services that are provided for specific usage. This specific services and their usage are defined by profiles.

Bluetooth profiles are defined as general behaviors through which Bluetooth devices communicate with other devices. These profiles are in charge of defining the possible applications that a Bluetooth device can support and operate. In order for one device to connect another, both devices must share at least one of the same Bluetooth profiles [11, 12].

Since a Bluetooth is developed, the manufacturer assigns specific Bluetooth profiles for that device, profiles that help to achieve a good level of interoperability between manufacturers and devices. Today, tens of profiles are defined by the Bluetooth SIG. In general, we should look at the profile as the requested tasks that an application needs to implement to be conformant to the specifications. Hence, it defines several parameters in different layers of the Bluetooth stack. [9]. Table 1.4 shows some examples of Bluetooth Profiles.

Table 1.4.1: Bluetooth Profiles

Some Common Bluetooth Profiles:
Audio/Video Remote Control Profile (AVRCP)
Basic Imaging Profile (BIP)
Basic Printing Profile (BPP)
Cordless telephony Profile (CTP)
Device ID Profile (DID)
Fax Profile (FAX)
File Transfer Profile (FTP)

Hands-Free Profile (HFP)
Headset Profile (HSP)
LAN Access Profile (LAP)
Phonebook Access Profile (PBAP,PBA)

Chapter 2: Bluetooth Security

2.1 Bluetooth Security Overview

As the use of wireless technology around the world has exponentially increased during the past decade, so did increase the importance of keeping secure the transference of information through wireless communication systems such as Bluetooth technology. Most of the popular communication technologies require robust security communication mechanisms in order to protect the transfer of information, and the complexity of implementing a secure communication link is greatly increased when we are dealing with wireless transmissions. Wireless technologies, such as internet and cell phones, continuously maintain transference of digital information, which in many cases could be private, confidential or even classified. A break or spoof into this transference of digital information could be the cause of damages such as partial or complete lost of information, or just a simple misuse of the same. An example of this problem could be the information managed for the U.S. Government and International Banking Businesses, where is crucial to preserve a good wireless security system due that this type of information is extremely relevant, awfully confidential, and must be maintained protected.

Bluetooth as one of the newest, most popular, and innovative wireless technologies for short distance communications, can be taken as an example of different types of security breaks. Bluetooth technology is currently employed in many relevant applications, such as computer laboratories at schools, banking offices, private businesses, and government organizations, among other important institutions [13]. Although the continuous research and improvements made through the past years by SIG on Bluetooth, this technology is frequently attacked due to the yet many vulnerabilities in its communication protocols. Despite the fact that Bluetooth has a short-range transmission specification, which in some cases could prevent from attacks, its security system can still be broken [14]. Bluetooth technology needs to employ a more robust security protection for authentication and transference of data, so no important information is compromised.

2.2 PROBLEM OVERVIEW

Since the first versions of the Bluetooth specification, this type of communication has been very vulnerable to different types of attacks, such as Bluesnarfing, eavesdropping, Bluejacking, Bluetooth Wardriving, among others [14]. However, most of these security treats only observe and listens to the information that is transferred from one Bluetooth device to the other, or just bother Bluetooth users by sending out messages without causing any real and permanent damage. However, there is one more efficient and damaging attack called impersonation, that besides listening to the information that two or more Bluetooth devices are sharing, it uses this entire information to impersonate a device and obtain and/or modify information from other Bluetooth device. As Shaked and Wool states in the Cracking the Bluetooth PIN document, "The possible damage of a successful wireless attack starts with the ability to eavesdrop on the data transferred during the communication of two devices, and ends with the ability to fully impersonate other devices [15]."

A third party, usually called man-in-the-middle, observes all the communication between two devices during a pairing procedure and obtains all parameters exchanged between them over the air interface. The intruder or man-in-the-middle usually uses this information in order to force a pairing with one of the Bluetooth devices that were trying to form a connection and communicate. This type of attack is usually performed in order to obtain some benefit from the affected device, such as having full access to private information. Basically, an intruder will successfully connect to the affected device by using other's trusted credentials.

In order to prevent Bluetooth devices from this impersonation attack, this document proposes an enhanced method in the Bluetooth security communication process, which is based in the use of a digital certificate and a Certificate Authority. This enhanced step does not modify any existing Bluetooth pairing and authentication procedure; on the contrary, this proposed method is performed in addition to and at the conclusion of the authentication procedure, just before initiating the communication and data transmission between devices. This proposed

method for Bluetooth authentication will effectively avoid impersonation attacks on this wireless technology.

2.3 PREVIOUS SECURITY RESEARCH AND CONCERNS

Bluetooth security has always been a concern for the SIG group, who has been constantly updating the Bluetooth specification core based on all the research performed in look for an improve in the Bluetooth Security mechanisms.

The first Bluetooth System enclosed the initial operating version, Bluetooth 1.0 V, which was subsequently followed by the 1.0B, 1.1 and 1.2 versions. All these older versions shared problems with poor security mechanisms, several connectivity difficulties between Bluetooth systems of different operating versions, as well as difficulties in making their Bluetooth products interoperate. The most recent Bluetooth versions, the 2.0 V., 2.1 V., and 3.0 V., eliminate many of the security issues that older versions have, as well as they increased as much as three times its transmission speed and decreased their power consumption. However, this latest versions still keep some of the past versions' weaknesses, such as the one found in the communication Key Management process [16]. Bluetooth is still potentially vulnerable to many attacks.

2.3.1 Initialization key and PIN concerns

The generation of the initialization key is still a latent concern due that its strength is purely based on the PIN code. The E_{22} generation algorithm derives the initialization key from the Personal Identification Number (PIN), the length of the PIN code, and a random number, data that is transmitted over the air. The E_{22} algorithm output is highly questionable as the only secret is the PIN code. The security of the whole system relies on the user's choice of a secret PIN, which is often too short [15]. When using a 4 digit Pin Code for the pairing process, there are only ten thousand different possibilities for this PIN number. Moreover, adding the fact that 50% of used PINs are "OOOO", the trustworthiness of the initialization key is quite low. Due to this problem, the major proposal is the use of a longer PIN for protection of further pairing and authentication parameters [17, 18].

The Bluetooth Device address, which is unique to each Bluetooth device, introduces another security problem. When a connection to a device is made and its Bluetooth device address is obtained, it is easy to track and monitor the behavior of this device. Privacy may be compromised if the Bluetooth device address (BD_ADDR) is captured and associated with a particular user. Once the BD_ADDR is associated with a particular user, that user's activities could be logged, resulting in a loss of privacy [8].

These two security flaws, PIN Code and BD_ADDR have been a concert to the Bluetooth SIG since the first Bluetooth Core Specifications. However, up to the latest version of this core, these two main security flaws still remain under the known Bluetooth security issues, with no evident solution.

2.3 Proposals from other researchers

2.3.1 Enhanced Security Layer

The Wireless Security and Cryptography book [18] proposes an enhanced security layer (ESL) for improving the security of Bluetooth technology. This proposal intends to increase the security level by replacing the original Bluetooth encryption scheme with a design based in advanced encryption standards (AES). Except for Bluetooth, this AES scheme is currently employed in all significant short-range wireless technologies such as IEEE 802.11, 802.15.3, 802.15.4 and ZigBee.

This proposed security layer is intended to be placed in top of the standard Bluetooth controller interface as can be shown in Figure 2.3.1.

For protecting data transfers, ESL supports four well-scrutinized operation modes from which the application can choose the preferred one according to its security requirements. In addition, two enhanced entity authentication and key agreement protocols are included.

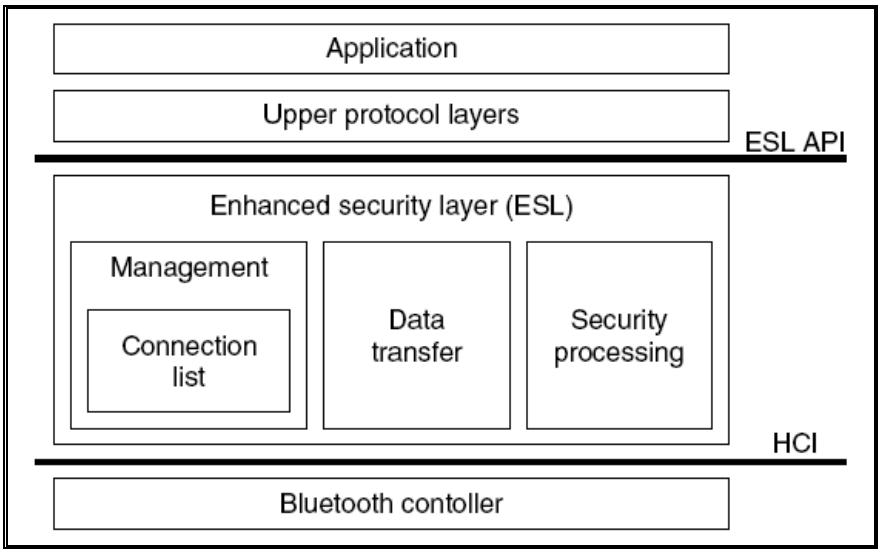


Figure 2.3.1: ESL architecture.

ESL replaces the E0 cipher used for the encryption of data, with AES. AES is a symmetric cipher that encrypts data in 128-bit blocks, with successive interaction rounds. Since ESL is located above the HCI, only HCI packet payloads can be encrypted.

The handshake portion of the widely employed and trusted transport layer security protocol was chosen as the public-key protocol and the authenticated key exchange protocol 2 (AKEP2) as the secret-key protocol. Both the protocols result in shared temporary secrets that are used as keying material for generating ESL keys as well as PIN codes. The authentication message exchanges are treated as regular HCI data transfers by the lower protocol layers. ESL requires the PIN size to be at least 12 digits. Authentication without invoking encryption and protecting connections with a unit key is not allowed. Either of the ESL key exchange protocols can be used for automatic PIN exchange to prevent poorly chosen values and to make the link setup more convenient for users. Please refer to this Wireless Security and Cryptography source edited by Nicolas Sklavos and Xinmiao Zhang for a deeper and clearest understanding of this method [18].

2.3.2 Providing Anonymity

The existing flaws in the Bluetooth security have motivated the development of a new Bluetooth mode of operation that is intended to provide protection against some location privacy threats. This new mode is called Anonymity and is currently not part of the Bluetooth standard [19].

A location attack is usually based on the BD_ADDR, the channel access code, and the device access code. A good way of protecting Bluetooth devices from this type of attack would be to regularly change the device address, which is the basic idea of this Anonymity proposed mode.

For this proposal three types of address are suggested: fixed address, active address, and alias address. Moreover, this mode also proposes a small change in the way inquiry, paging and pairing are performed.

Each Bluetooth transceiver is allocated a unique 48-bit Bluetooth device address (BD_ADDR_Fixed) from the manufacturer. This address is used to allow a device to directly page another that it has previously paired with. Without this type of address, Bluetooth devices would always need to repeat the entire inquiry procedure.

The active address (BD_ADDR) has to be regularly updated by Bluetooth devices. Since this address is intended to constantly change, it is not possible to track a device using this address. During a connection with other devices, this address is still constantly changed, but this process has to be sent to other devices connected before the change happens.

Due to this proposed method, the authentication procedure is slightly modified. The modified authentication, also called alias authentication, is based on the usage of a third key called alias addresses (BD_ADDR_alias). This key is purely used for authentication purposed.

A Bluetooth device can only operate in a nonanonymous or anonymous mode at a time. Bluetooth not supporting this anonymous node has to operate in the normal mode or nonanonymous.

Chapter 3: Bluetooth Security Attacks

3.1 SECURITY ATTACKS

The pairing and authentication inside the Bluetooth mechanisms are intended to prevent this technology from certain attacks; however, in most of the cases, these procedures are the main cause of some important security flaws.

One of the known cases of attacks is the one caused by the normal skipping of the pairing procedures. After the first time that two or more Bluetooth devices pair, each device stores the other device information in a non-volatile memory (device address and link key), so in subsequent pairings these devices do not have to repeat the complete pairing procedure, and they can jump directly to the authentication part. Unfortunately, although this shortcut in pairing creates a faster connection between devices, it also leaves an open door for possible future attacks.

If a secret key is disclosed to an adversary, there is an obvious risk of an impersonation attack, simply by using the stolen key and the Bluetooth device address from which the key was stolen. For example, once the link key of a PC and a cell phone is known, an adversary can silently connect to the cell phone Bluetooth device, impersonate it, make use of any service the Bluetooth device offers and connect to the PC by showing the cell phone credentials [20].

When a third party, or a called “man-in-the-middle,” observes all the communication between two trusted devices during the pairing procedure, this device can obtain all the parameters exchanged over the air interface. The parameters needed for an attack are the device addresses, the PIN value, and the encrypted random value called link key [1]. These parameters can be obtained by eavesdropping the pairing connection or by calculating them from the list parameters transferred. The PIN and the Link Key are the unknown parameters that need to be calculated [19].

Assume that the attacker eavesdropped (listen and capture) on an entire pairing and authentication process, and saved all the messages. Table 3.1.1 shows a list of the main messages

(without acknowledgments) that an eavesdropper can capture from an entire pairing and authentication procedure.

Sender	Destination	Data	Length	*Format*
A	B	IN_RAND_A	128 Bit	*plaintext*
A	B	LK_RAND_A	128 bit	*XORed with* K_{init}
B	A	LK_RAND_B	128 bit	*XORed with* K_{init}
A	B	AU_RAND_A	128 bit	*plaintext*
B	A	SRES	32 bit	*plaintext*
B	A	AU_RAND_B	128 bits	*plaintext*
A	*B*	*SRES*	*32 bits*	*plaintext*

Table 3.1.1: List of messages sent during pairing and authentication process.

After this listening, the attacker can now use brute force algorithms to find the PIN used, enumerating all the PIN possible values. Knowing IN_RAND and BD_ADDR, the attacker can run the E_{22} algorithm and submit these inputs and the guessed PIN, and finally find a hypothesis K_{init}. The attacker can now use this K_{init} to decode LK_RAND_A and LK_RAND_B. All this information gathered or calculated is just sufficient to perform calculations in order to obtain K_{ab}. Now, by using K_{ab} and the transmitted AU_RAND_A, the attacker calculates SRES and compares this value with the captured SRES. It is important to mention, that this type of attack is usually more possible and successful when the PIN values are small, usually of 4 digits or less. Otherwise, the amount of possible PIN values increases and the calculations in order to obtain all the pairing and authentication values is less likely to be successful [3, 21].

However, what happens if an attacker wants to impersonate a device A, but it just missed the whole pairing process between devices A with B? It is important to recall that the impersonation attack is only possible if an attacker has eavesdropped the entire pairing and authentication processes. Once a link key K_{ab} is created, each Bluetooth device stores it for future possible communications. If at a later point these devices initiate a communication, the

stored link is used and the entire pairing process is skipped. There is a second type of attack that contemplates this skipped process.

This attack forces two known devices to repeat the entire pairing and authentication process again. There are three known ways of force devices to go thru the whole process [15]:

1. Since the devices skipped the pairing process and when directly to the authentication phase, the master device sends the slave an AU_RAND message and expects SRES in return. In such a case, the attacker injects the master with an LMP_not_accepted message. This message is basically saying to the master that the slave forgot the link key and they have to go through the complete pairing process again. The master accepts this message, discards the link key for the previous connection these devices had, and the pairing process is restarted.
2. When the master and the slave start the Authentication phase, the attacker has to inject the master with an IN_RAND message before the master can send the AU_RAND message towards the slave. With this procedure the master will be convinced the slave lost its link key and the pairing will be restarted.
3. During the authentication phase, the master device sends the slave an AU_RAND message, and expects SRES in return. If the attacker injects a random SRES to the master, this will cause the authentication phase to restart and repeated attempts will be made. After certain number of failed attempts, the master declares the authentication phase as failed and the devices initiate pairing.

Now what would happen if an attacker is able to listen to all the pairing and authentication parameters from two trusted devices trying to communicate? If the attacker successfully obtained all the parameters from the pairing and authentication phases, then this intruder is able to impersonate any of those two trusted devices.

In order to make an impersonation attack, the attacker needs to access its own link key's database and insert the just acquired one. Having this key properly placed in the database, this attacker needs to go into range of the device that wants to connect and initiate the inquiry

procedure. Due that the attacker will try to connect by using some trusted credentials, the other device being inquired will just jump to the authentication procedure and the this phase will be a success. Figure 3.1.1 shows how this impersonation attack, where Case A is a trusted case, and Case B, is a case with the impersonation attack.

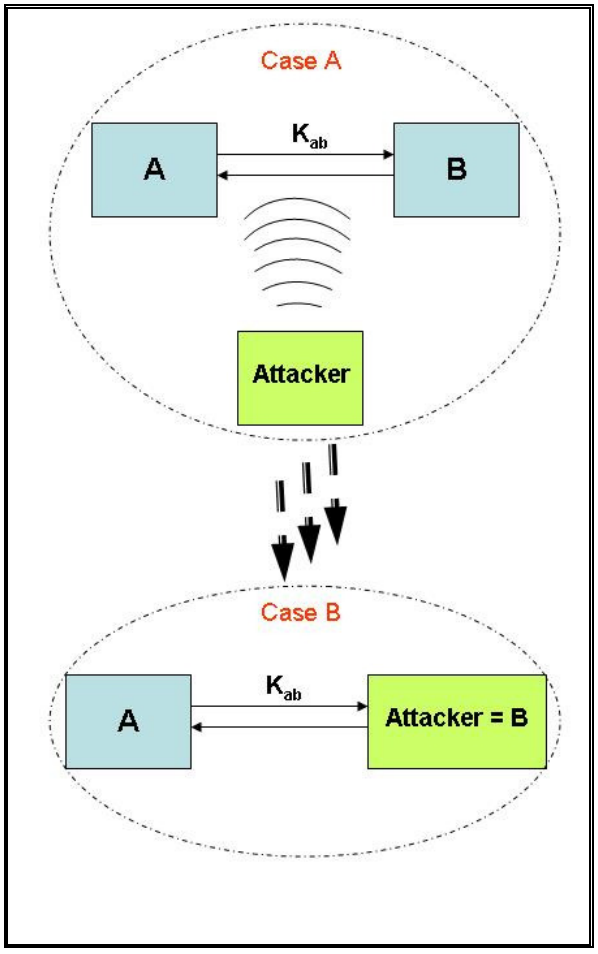


Figure 3.1.1: Impersonation Attack.

Chapter 4: Bluetooth Key Management Overview

4.1 LINK KEYS

Bluetooth technology makes use of several types of keys in order to communicate between devices. These keys, called link keys, are classified in two types, semi-permanent and temporary, depending on the type of Bluetooth Device, its application and memory capabilities, and the key volatility from the device [19].

4.1.1 Semipermanent and Temporary Keys

Semi-Permanent Keys are store in Bluetooth devices with non-volatile memory. Usually these types of Bluetooth devices have enough memory capabilities in order to store several link keys from different pairing connections with other devices. Once this key is defined by two Bluetooth devices, it may be used in several subsequent pairing between these same devices.

On the contrary, Temporary Link Keys are defined for Bluetooth devices with low memory capabilities, devices that have small applications, or with a limited user interface. The lifetime of this type of key is limited by the lifetime of the current session and this key cannot be used for future connections [2].

4.1.2 Application Keys

Link Keys for Bluetooth devices are also defined for the type of application. There are five types of keys depending on the Bluetooth device application:

1. Initialization Key (Kinit).
2. Unit Key (Ka or Kb).
3. Combination Key (Kab).
4. Master Key or Temporary Key (Kmaster).
5. Encryption Key (Kc).

All of these Link Keys contribute to the initialization, pairing and authentication of the Bluetooth System, processes that are followed in order to have a secure connection between

Bluetooth devices. Each key performs a special and unique role in this pairing and authentication process by protecting Bluetooth's parameters or another pairing information [22].

4.1.2.1 Initialization Key

Initialization key (Kinit) fully depends on the identity of the device with a variable Personal Identification Number (PIN). This key is only used during the pairing process, when no previous link key exists between devices, and is particularly generated in order to protect the transfer of initial parameters. After these parameters are transferred to another device, this key is discarded [22].

Algorithm E22 is used for deriving Kinit having as calculation parameters the BD_ADDR, Random Number (BD_RAND), Pin and the Length of this Pin (L).

When any of the two devices has a fixed PIN, the BD_ADDR of the variable PIN's Bluetooth device is used. When both Bluetooth devices have variable PIN, then they use the PIN of the slave device that receives IN_RAND (BD_RAND) [15]. Figure 4.1.1 shows how this Kinit is generated. For a deeper description of the E_{22} Algorithms, please refer to Appendix A at the end of this document.

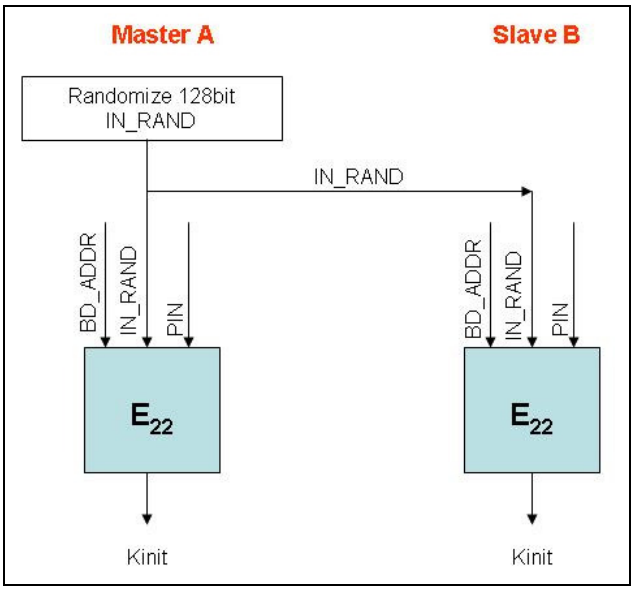


Figure 4.1.1: Kinit Generation.

4.1.2.2 Unit Key

Unit key (Ka or Kb) is created and saved on a Bluetooth's memory since its factory settings are installed. Basically, this Unit Key is created on the device since the first time the device is turned on, and after its installation, this key is rarely changed.

This type of link key is the perfect example for Bluetooth devices with low memory capabilities due that this key is identical for every pairing and dependent on a single device. Moreover, this key is installed in equipment that will be accessible to a large number of users, so their type of application requires a more secure link key. These types of devices only need to store their own key. A good example of this type of Link key is the connection generated between a PC and a Printer, where the printer is the type of Bluetooth devices that usually posses low memory capabilities.

In order to generate Ka or Kb, a Bluetooth device uses its own Bluetooth Address (BD_ADDR) and a randomly generated number, and Ka or Kb is calculated based on the encryption algorithm E21 [19].

This key is not recommended by the SIG specification core since the Bluetooth 2.1v. [18].

4.1.2.3 Combination key

Combination key (Kab) is derived new from each new communication between Bluetooth devices and requires both peers' unit keys in order to generate it. This key requires a device with more memory capabilities due that a different Combination Key must be stored for each connection to a different device. Moreover, more Higher Security Level Applications such as the encryption of formal communication also require the use of this key. This key is based on the Bluetooth device knowledge of the other parties BD_ADDR, the previously obtain Kinit, and the use of the encryption algorithm E_{21} [22]. Figure 4.1.2 shows how this Kab is generated. For a deeper description of the E_{21} Algorithms, please refer to Appendix A at the end of this document.

It is important to mention that a Combination key and a Unit Key are functionally indistinguishable because they only differ in the way they both are generated.

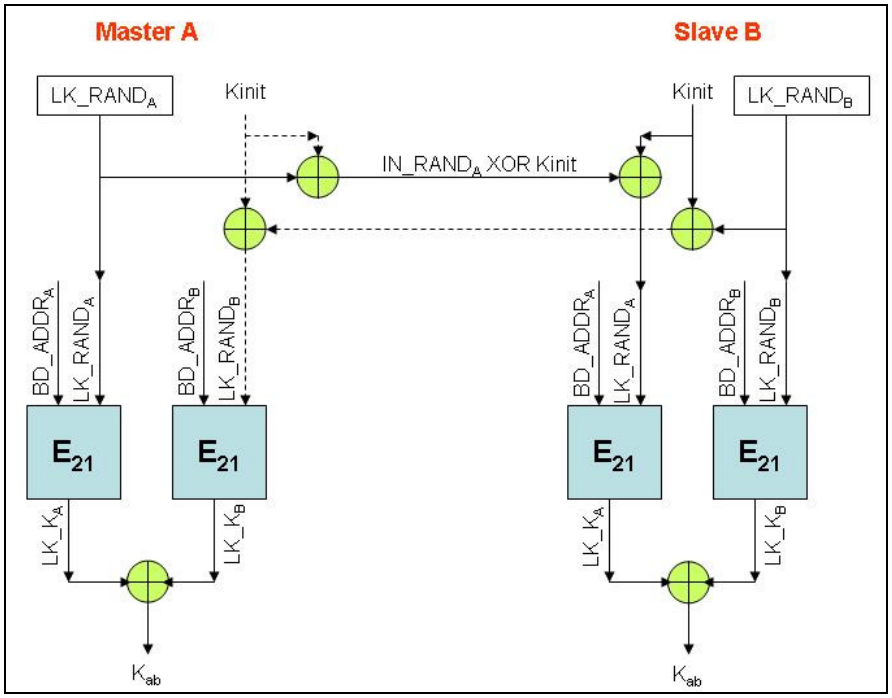


Figure 4.1.2: Kab Generation.

4.1.2.4 Master key

Master key (km) is used only when more than two Bluetooth devices are trying to communicate. This is a temporary key used in order to protect data sent in broadcast messages, where a Master device is communicating the same data to several devices known as slaves. The Master key or Km replaces the link key until the broadcast communication is terminated. This key is generated from two random numbers, a PIN and the PIN Length [22].

4.1.2.5 Encryption key

The last key, the Encryption key (ke), protects the communication packets when initialization procedures between Bluetooth parties are already completed. However, this type of key will no longer be discussed on this document due that it goes beyond this Thesis Research.

4.2 RULES FOR SELECTING LINK KEYS

The Bluetooth Device Service is the one that decides on using either a Unit Key (Ka or Kb) as Link Key or if deriving a Combination key, all this depending on the link application and the Bluetooth device's memory capabilities. In order for a Bluetooth device to take this decision, there exits some rules [19, 22, 23]:

1. If one device sends a unit key and the other sends a combination key, the unit key will be taken as the link key.
2. If both devices send a unit key, the Master's unit key will be the Link Key. During the initialization phase, the application must decide which component is going to share its own Unit Key as the paring key. Typically, this is the component which has limited memory capabilities due that this can only remember its own unit key.
3. If both devices send a combination key, the Link shall be calculated.
4. When no link key can be established between two devices, a pairing process starts where an initialization key is generated and used for authentication. This key is created by entering a common personal identification number (PIN) in each of the devices.

4.2 SECURITY OPERATION MODES

Bluetooth uses modes of operation as an additional protection of security. There are three modes of operation, and a Bluetooth device can operate only one of these modes at a particular time [14]:

- Security Mode 1: Nonsecure Mode.
- Security Mode 2: Service-level enforced security mode.
- Security Mode 3: Link-level enforced security mode.

4.2.1 Security Mode 1

Under this mode 1, a Bluetooth device will not initiate any security procedure; the security functionality is completely bypassed. This mode never demands authentication and

encryption of the link. This unsecured mode is provided for application in which security is not a relevant concern [19].

4.2.2 Security Mode 2

Security procedures are initiated after the channel establishment at the Logical Link Control and Adaptation Protocol (L2CAP) level. For this security mode, a security manager controls access to services and to devices [14].

4.2.3 Security Mode 3

Under this mode, a Bluetooth device initiates security procedures before the channel is established. This mode supports authentication (unidirectional or mutual), and encryption [19, 23].

Chapter 5: Bluetooth Pairing and Authentication Protocol

5.1 PAIRING PROCEDURE

Pairing is the process through which two or more Bluetooth devices share secret information in order to create a network link. Basically, this Bluetooth pairing procedure is best described by the following four steps, as shown in figure 4.1.1:

1. Generating an initialization key;
2. Generating a Link key;
3. Link key exchange;
4. Authentication.

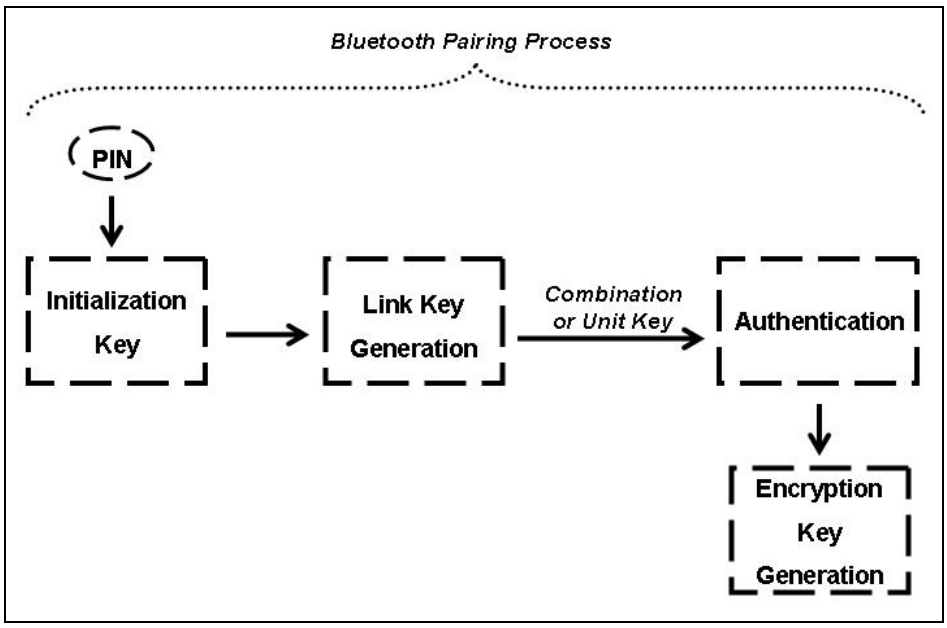


Figure 4.1.1: Bluetooth Key Management.

After two devices meet, an initialization key is created based mainly on a PIN that both devices must enter on the Bluetooth devices. These PIN values on both devices must coincide in order to continue to the following step.

Bluetooth device services are the ones that decide on using the unit key or deriving a combination key as link key, all this depending on the link application and memory capabilities

of the device. This link key is basically the most important key in the pairing process, due that it will be the parameter to authenticate in the following step which is the challenge response, and moreover, this key is a very important parameter in the final encryption communication process.

The authentication procedure, called “challenge-response” scheme, is where a Bluetooth device, called verifier, sends a random challenge to a claimant device and expects a valid response value in return in order to authenticate the connectivity [19]. A verifier device challenges a claimant device by sending a random value and expects its response. This challenge response is computed by the claimant, which uses for inputs the exchanged link key, the random value the verifier sends, and the unique address from the claimant. The claimant sends to the verifier its response, and this last compares response values. If both responses coincide, the communication is accepted; otherwise, the link is broken. Sometimes this authentication needs to be performed by both parties, so an exchange of roles is needed (verifier and claimant).

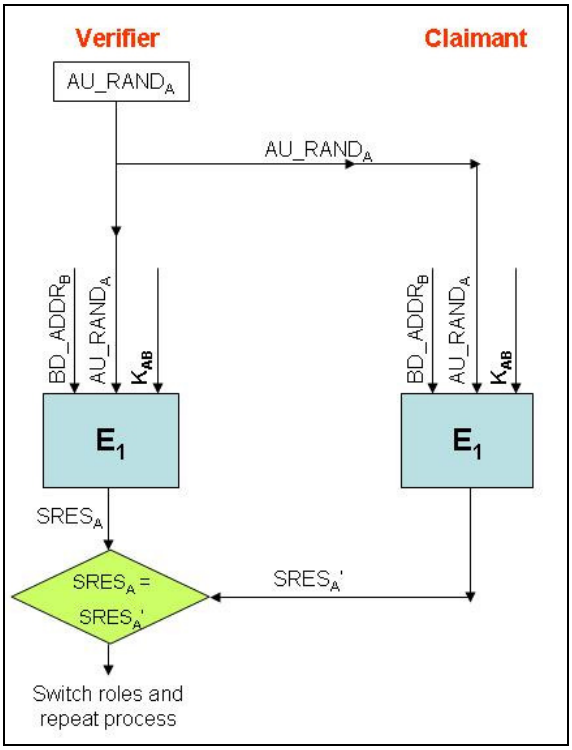


Figure 5.1.1: Challenge response scheme.

The encryption key is generated just after the authentication, and when the communication between devices is about to start. Basically, this key is generated when the device activates the encryption mode for transmitting information, and it changes every time the Bluetooth enters this encryption mode. The encryption step will not longer be covered on this thesis due that this key goes beyond this research.

5.2 General Pairing and authentication protocol

Before data can be transmitted over the air between two Bluetooth devices, it is essential that the devices discover each other and link up. In order to create a communication between two devices, first the devices must be within connection range of each other, and then one device shall be inquiring other devices (transmitting a series of inquiry packets at different frequencies) and the other device shall be listening to this inquiries (the device shall be set in discoverable mode in order to be found).

A unit is said to be in the inquiry state when it attempts to discover other devices in a piconet. An inquiry is used for fresh connections when no previous connections between devices have been made, and devices do not have stored the other Bluetooth device's address. An inquiry is skipped in case of previous connections and passes to a new step where the common link key is authenticated [11].

Only slave devices send inquiry response messages, which include information as the slave Bluetooth device address and the clock information. If an inquiry message from the master is answered and accepted by the slave, then both devices are ready to look for a common hop frequency in order to start the pairing stage.

Before two devices can establish a connection, they must be in page and page scan mode; the paging device initiates the connection, while the page scanning device responds. In order to be able to page, the paging device must know the ID of the page scanning device. This ID can be obtained from an inquire response [5].

The pairing procedure starts when the master device sends a random number (IN_RAND_A) to the slave and waits for a received acknowledgment from the slave. If the acknowledgment is received, both devices request the submission of a PIN code, which must be equal in both devices. When this PIN is entered, the devices compare both PIN by sending the entered one to the other device. If the comparison is correct, the devices send correct acknowledgments to the other device, and continue to the calculation of the initialization key (K_{init}).

With all these information, both devices calculate K_{init} by the following formula,

$K_{init} = E_{22}\{IN\_RAND, BD\_ADDR_B \text{ added to PIN, addition Length}(L')\}$.

After K_{init} is calculated, each device generates a random number LK_RAND_A and LK_RAND_B respectively, which is used in order to obtain LK_K_A and LK_K_B as follows:

$LK\_K_A = E_{21}\{LK\_RAND_A, BD\_ADDR_A\}$

and

$LK\_K_B = E_{21}\{LK\_RAND_B, BD\_ADDR_B\}$.

After this step, devices A and B now calculate C_A and C_B respectively,

$C_A = K_{INIT}$ XOR LK_RAND_A

and

$C_B = K_{INIT}$ XOR LK_RAND_B.

C_A and C_B are transferred to device B and A respectively in order to be used for device A and B to calculate the unit and combination keys, which must be equal for both devices.

Device A calculates LK_K_B and KA_B as follows,

$LK\_RAND_B = C_B$ XOR K_{INIT},

$LK\_K_B = E_{21}\{LK\_RAND_B, BD\_ADDR_B\}$,

and

$K_{AB} = LK\_K_A$ XOR LK_K_B.

In the same way, device B calculates LK_K_A and KA_A as follows,

$LK\_RAND_A = C_A$ XOR K_{INIT},

$LK\_K_A = E_{21}\{LK\_RAND_A, BD\_ADDR_A\}$,

and

$K_{AB} = LK\_K_A$ XOR LK_K_B.

After the new link key, K_{AB}, is created, the old K_{init} is discarded.

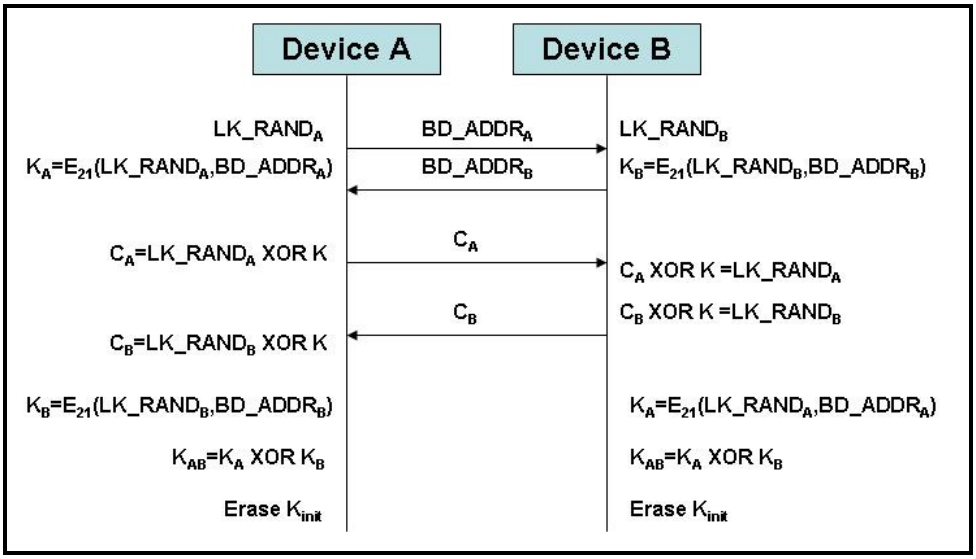


Figure 5.2.1: Unit and Combination Key Calculation.

After the link key, which is the unit or combination key depending of the case, the devices enter the mutual authentication process which is based on a challenge response transaction. The intent of this challenge response transaction is to authenticate the identity of each Bluetooth device, in order to avoid the pairing of any intruder device.

The device authentication is initiated by using a random starting point. The master sends a random number, AU_RAND_A, to the slave. The claimant uses this random number, the slave's BD_ADDR, and the link key in order to calculate the response to the challenge (SRES). This response is calculated using the algorithm E1,

$SRES = E_1\{K_{AB}, AU\_RAND_A, BD\_ADDR_B\}$,

This response is sent back to the master device in order to be compared to the master response, SRES', which is calculated with the same parameters. If SRES is equal exactly to

SRES', then the master device sends a correct acknowledgment to the slave and the challenge response transaction roles are inverted, and now the slave challenges the master device in order to authenticate its identity.

In this occasion, the slave device generates a random number AU_RAND_B and sends it to the master device. The master device receives this random number and calculated the response SRES to the challenge,

$SRES = E_1\{K_{AB}, AU\_RAND_B, BD\_ADDR_A\}$.

This response is sent back to the slave device who compares this response SRES with its own one SRES', and if both are completely equal, it sends a correct acknowledgment to the master device.

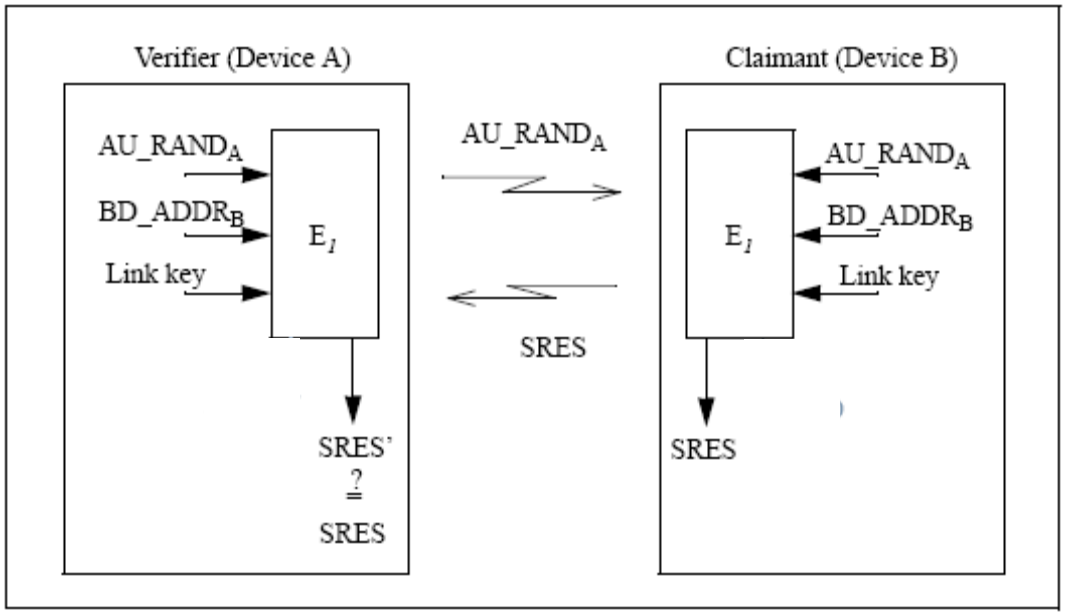


Figure 5.2.2: Challenge Response.

After all this pairing and authentication procedure, the last step is the creation of an encryption key (K_c), which is used in order to protect any communication information that is transfer between devices.

Chapter 6: Proposed Method

6.1 CERTIFICATE AUTHORITY

A Certificate Authority (CA) is a cryptographic entity or Trusted Third party, which secures digital communication based on Digital Certificates. Some of these entities offer their Cryptographic Services free of any change, but most of them have some charge over them or create their own service such as Government Institutions.

A CA issues digital certificates which contain a public key and owner identification. The CA makes sure this public key contained in the digital certificate belongs to the person, organization, server or other entity noted in the certificate. The obligation of this CA is to verify the applicant's credentials, so that users and replying parties can trust the information in the digital certificates. If the user trust the CA and can verify the CA's signature, then they can also verify that a certain public does indeed belong to whoever is identified in the certificate [24, 25].

The market of certificates used for website security is largely held by a small number of multinational companies. VeriSign and its acquisitions have approximately a 60% share of the CA market, followed by commodo (8%) and GoDaddy (6%).

6.2 PROPOSED METHOD

In order to eliminate the impersonation attacks in the Bluetooth communication system, a security mechanism based on digital certificates is proposed. This proposed mechanism is mainly based on different levels of trust:

1. The first level of trust, which is the most trustworthy or reliable, should be given to those Bluetooth devices which have an embedded authenticity certificate and the verifier device is able to check this authenticity with a Certificate Authority. This level consists of two sub-levels:
    a. This first sub-level is set when both Bluetooth devices have internet capabilities, and consequently, each device can authenticate the other device's digital certificate through the Certificate Authority. If both authentications are

successful, and the Certificate Authority accepts both digital certificates as trusted, then the actual communication starts. In the contrary, if any of the two certificates are checked as a non-trusted certificate, then the authentication fails and the devices go to the initial pairing step. If these two devices want to start a connection again, they will need to follow all the pairing and authentication steps including the enhanced one.

b. This second sub-level is set when only one of the two Bluetooth devices trying to communicate has internet capabilities. Consequently, both devices need to check their digital certificates through the device that has internet capabilities. In order to do this, the Bluetooth devices use special procedures in order to encrypt their digital certificates, so the other device can not try to modify or add any information to the certificate. This encryption and authentication procedure is fully described in the following paragraphs.

   If both authentications are successful, and the Certificate Authority accepts both digital certificates as trusted, then the actual communication starts. In the contrary, if any of the two certificates are checked as a non-trusted certificate, then the authentication fails and the devices reset and go to the initial pairing step. If these two devices want to start a connection again, they will need to follow all the pairing and authentication steps including the enhanced one.

2. The second level of trust should be given to those Bluetooth devices which have an authenticity certificate, but there is no possibility of validating it through a Certificate Authority. In this case, the devices can not go thru the step of checking their certificates with a Certificate Authority. However, these devices can follow the original Bluetooth pairing and authentication procedures, and they can also block certain applications such as address book and any other relevant documents, so an attacker can not access them and try to modify or steal them. However, due that only the original Bluetooth pairing and authentication procedures are followed, there is no

guarantee that an impersonation attack will not occur. For this reason, this level also offers the option of blocking any connection where no internet capabilities in any of the two Bluetooth devices exist.

3. The third level of trust should be given to those Bluetooth devices which do not have an embedded authenticity certificate, and for this reason, they need to recur to the usual and basic Bluetooth authentication method, where no good level of security is guaranteed. Under this level, there is always a possibility of an impersonation attack due that only the original Bluetooth pairing and authentication is followed. However, this level offers and recommends the option of blocking all the connections that are not based on the use and checking of digital certificates. If this blocking option is chosen, connections between devices with no digital certificates will not be allowed.

The Authenticity Certificate should be embedded on the Bluetooth chipset since it is manufactured. This certificate should not be modifiable due it should be hidden on the chipset since the device main installation. The certificate ought to contain a public part which is the one usable in order to authenticate the Bluetooth device, and a private one which is the one that is not modified.

An Authenticity Certificate must contain important information of the Bluetooth device such as its serial number, its manufacturer identification number and name, a private key (unique), the device address, and any other relevant information the manufacturer wants to include in order to prove the validity of the digital document and the device.

The method of authentication specified by the first level of trust requires that one of the Bluetooth devices, normally set as the verifier, has somehow access to internet. The proposed procedure states that after the common Bluetooth authentication procedure (after the “challenge response” scheme) has concluded, the claimant will send its Authenticity Certificate to the Certificate Authority, so the verifier can prove the claimant’s authenticity. The Certificate Authority will be in charge of checking if the certificate is valid and send response to the verifier.

These are the basics steps, considering that only one device has access to internet:

When device B wants to verify A,

- Device A sends its Digital Certificate to device B, so B can authenticate it.
- Device B encrypts this certificate and a random number with its private key, and sends it to the Certificate Authority.
- The Certificate Authority receives the information, decrypts it, and checks for the authenticity of device A's Digital Certificate.
- After assuring the certificate belongs to the correct device, the Certificate Authority sends an accepted response to device B. This response is the random number sent to the certificate authority.
- Device B checks the response and sends an acknowledgment to device A.

Figure 6.2.1 shows how this procedure is performed.

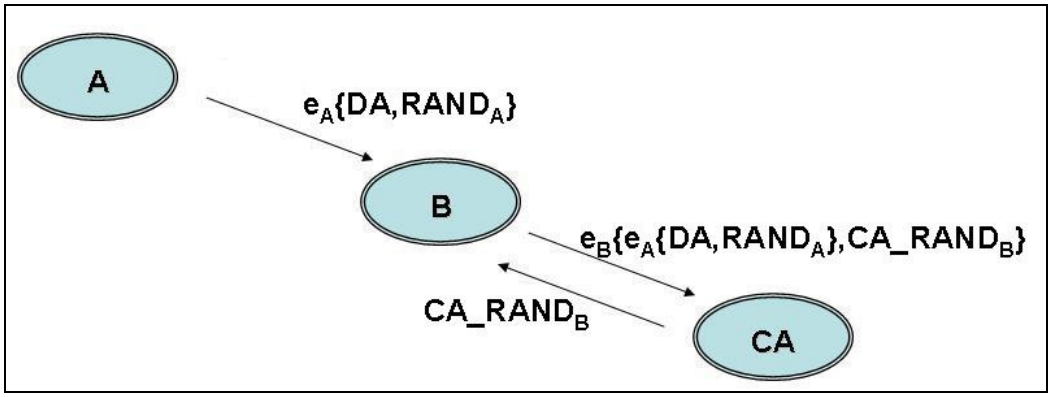


Figure 6.2.1: Device B certifies device A thru a Certificate Authority.

After device A receives the accepted acknowledgment from B, it starts to authenticate device B:

- Device B sends its Digital Certificate to device A.
- Device A receives it, encrypts the B's Digital Certificate and a random number with its private key, and sends this protected value to device B.
- Device B receives this protected information and forwards it to the Certificate Authority.

- The Certificate Authority opens decrypts this package and checks that the Digital Certificate matches its owner information.
- The Certificate Authority sends an acknowledgment response to device A through device B. This accepted acknowledgment is equal to the CA_RAND that device A encrypted with B's certificate.
- Device B receives this acknowledgment and forwards it to device A.
- Device A receives this random value and sends an LMP_Accepted to device B.
- After this the communication continues with the creation of the encryption key.

Figure 6.2.2 show how this second authentication is performed.

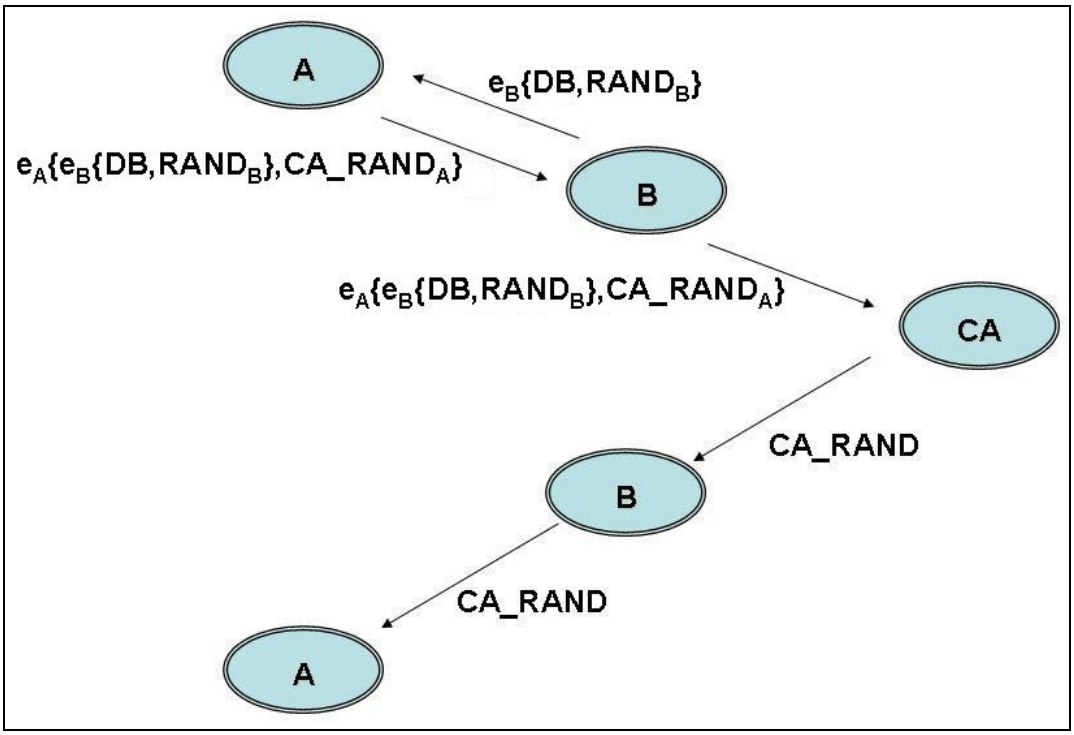


Figure 6.2.2: Device A certifies device B thru a Certificate Authority.

A Certificate Authority could be any trustworthy company such as the Bluetooth device manufacturer, or a known company like VeriSign which is a specialist in this type of verification.

When the first two proposed levels of trusts are not accessible, the non-volatile memory of the Bluetooth device might be erased completely. In such a way, the Bluetooth database, which includes all the information from previous and successful link pairings (device unique address and link key), would be entirely removed, forcing the devices to perform a full pairing process without skipping the usual pairing when two Bluetooth devices previously met.

Chapter 7: Simulations

7.1 Visual Pairing and Authentication Process

A C# code program was developed in order to show the full Bluetooth pairing and authentication process, for both the original and enhanced flow processes.

This simulation carefully shows, step by step, how Bluetooth pairing and authentication procedures happen. This simulation is divided in five simulations, each one for the type of pairing/authentication scheme needed to demonstrate the original and enhanced methods.

7.1.1 Original Bluetooth Pairing/Authentication Method

This simulation visually demonstrates the full steps followed in order to complete a pairing and authentication procedures of two devices, A and B, which have not met before. During this simulation, a third party, called hacker, will be listening to all the information that is transferred between the two devices that are trying to create a connection. The following figure shows a simulation for one of the steps in the original procedure.

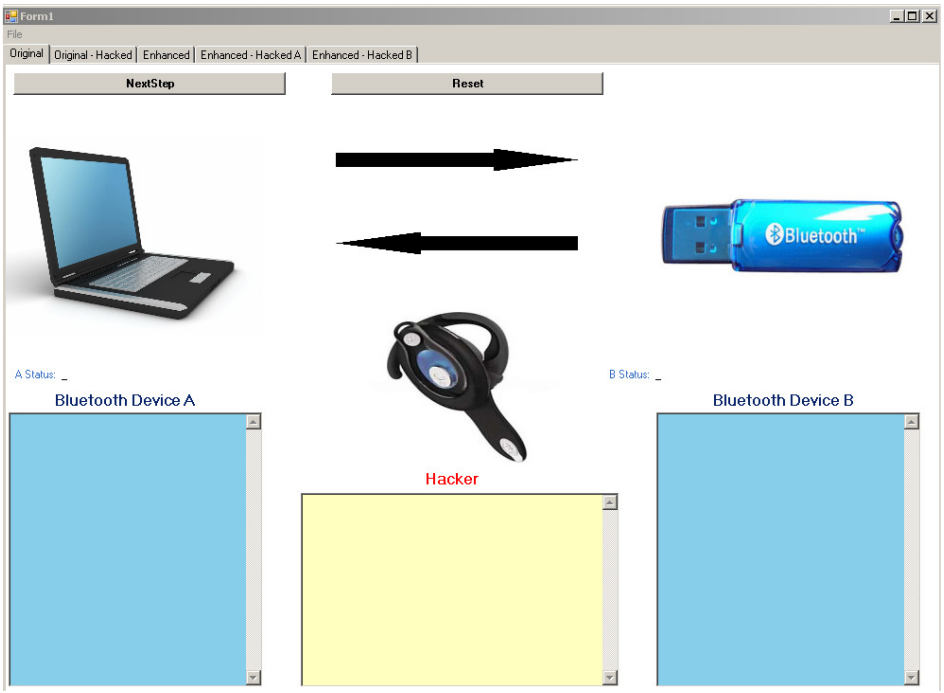


Figure 7.1.1: Original Method simulation.

7.1.2 HACKED-ORIGINAL BLUETOOTH PAIRING/AUTHENTICATION METHOD

This simulation shows how a third party or hacker can impersonate the device B by all the information obtained in the previous simulation, the Original Procedure. In this Hacked-Original procedure, device A considers the connection with the hacker as being another connection with B device, and due that A and B have previously met, all the pairing steps are skipped and they start directly from the authentication or challenge response.

Figure 7.1.2 shows a step on this authentication process with B impersonation.

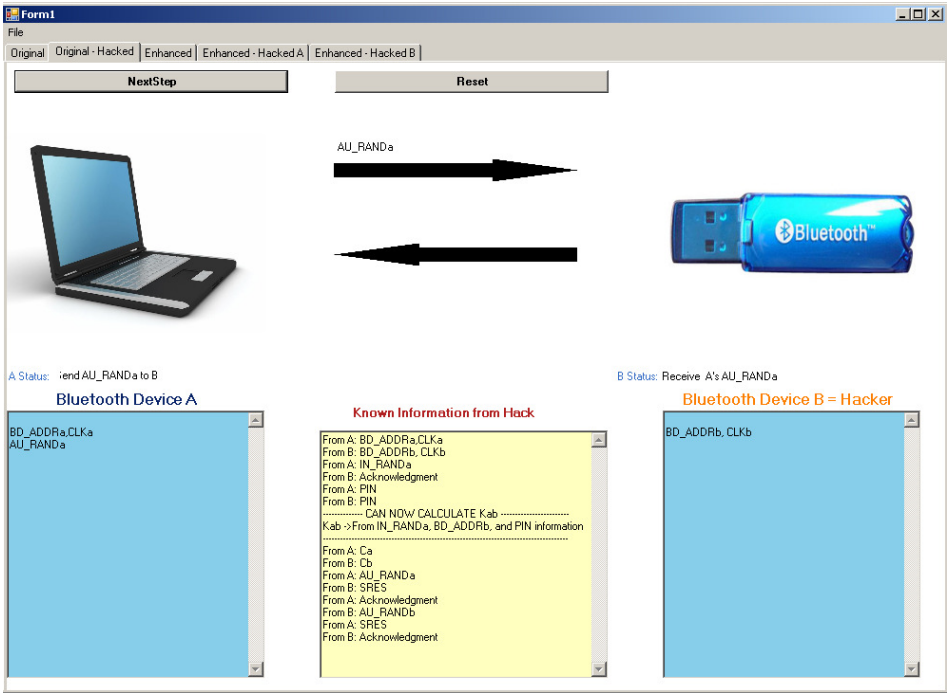


Figure 7.1.2: Original-Hacked Process Simulation.

7.1.3 ENHANCED BLUETOOTH PAIRING/AUTHENTICATION METHOD

This simulation shows the method purposed in this thesis document. Basically, this simulation is very similar to the original one showed in section 7.1.1, however, during this procedure the enhanced part is also showed. Moreover, besides device A, B, and the hacker listening to all A's and B's conversation, there is a four entity called the Certificate Authority, which duties where previously explain in chapter 6.

Figure 7.1.3 shows this enhanced process with a hacker listening to all the information being transferred over the air.

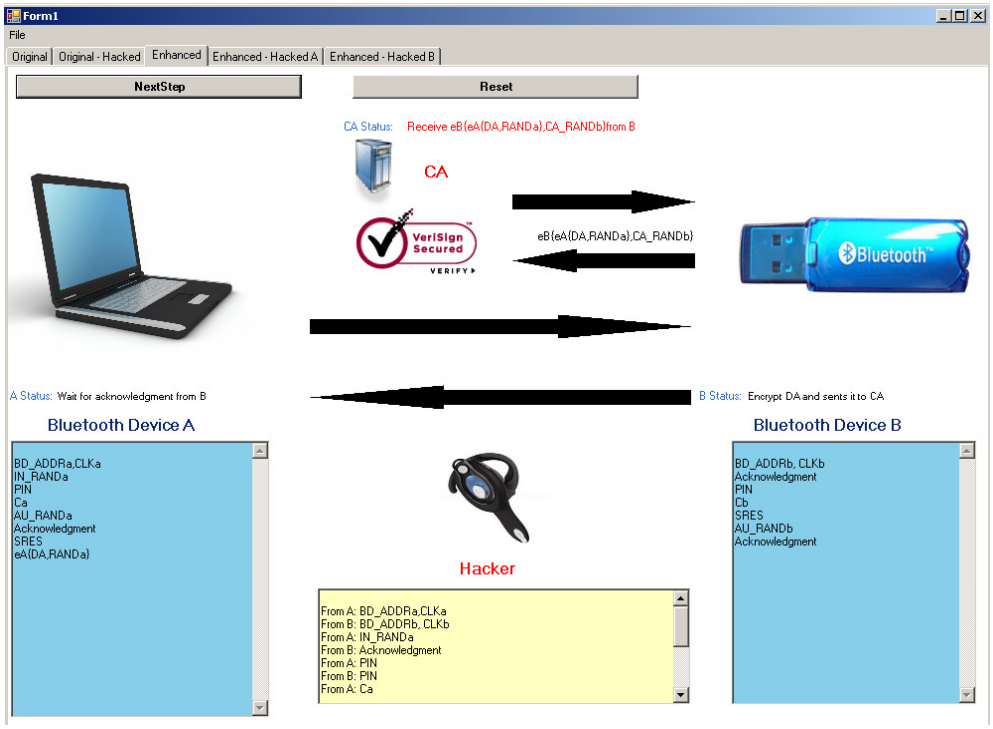


Figure 7.1.3: Original-Enhanced Process Simulation.

7.1.4 ENHANCED-HACKED BLUETOOTH PAIRING/AUTHENTICATION METHOD

This simulation represents the last step in the authentication method. This step is divided into two parts: the first one when a hacker impersonates A and connects to B, and the second one when a hacker impersonates B and connects to A.

7.1.4.1 Hacker impersonates A

During this process, the hacker impersonates device A, and creates a connection to device B. Due to the fact that devices A and B had previously met, the pairing procedure is skipped and they just go thru the authentication or challenge response process. However, this time as well as in section 7.1.3, there is an additional step to follow which is the authentication with the Certificate Authority. This additional step will cause an error in the connection due that A is a

hacker, and the connection will be lost, with no much possibilities of having a future success in case of being tried. Figure 7.1.4 shows this last simulation.

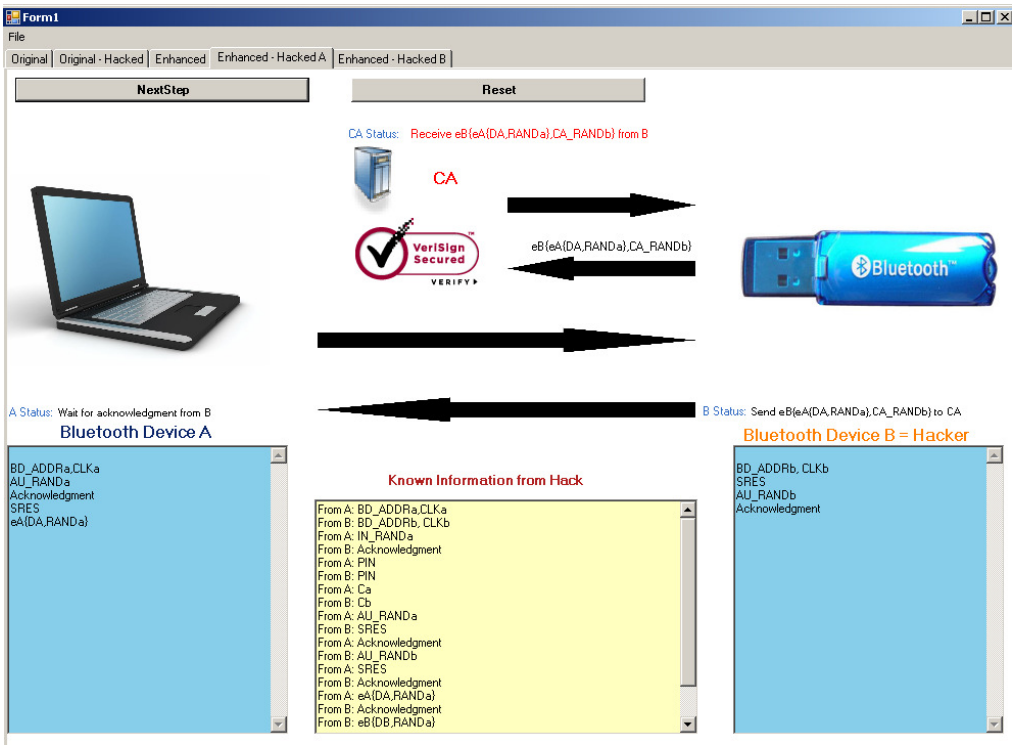


Figure 7.1.4: Enhanced-Hacked Simulation for Hacker A.

7.1.4.2 Hacker impersonates B

During this process, the hacker impersonates device B and creates a connection to device A. Similar to the procedure explained in 7.1.4.2, the pairing procedure is skipped due that devices A and B had previously met, and they just go thru the authentication or challenge response step. However, this time as well as in section 7.1.3, there is an additional step to follow which is the authentication to the Certificate Authority. This additional step will cause an error in the connection due that B is a hacker, and the connection will be lost, with no much possibilities of having a future success in case of being tried. Figure 7.1.5 shows this last simulation.

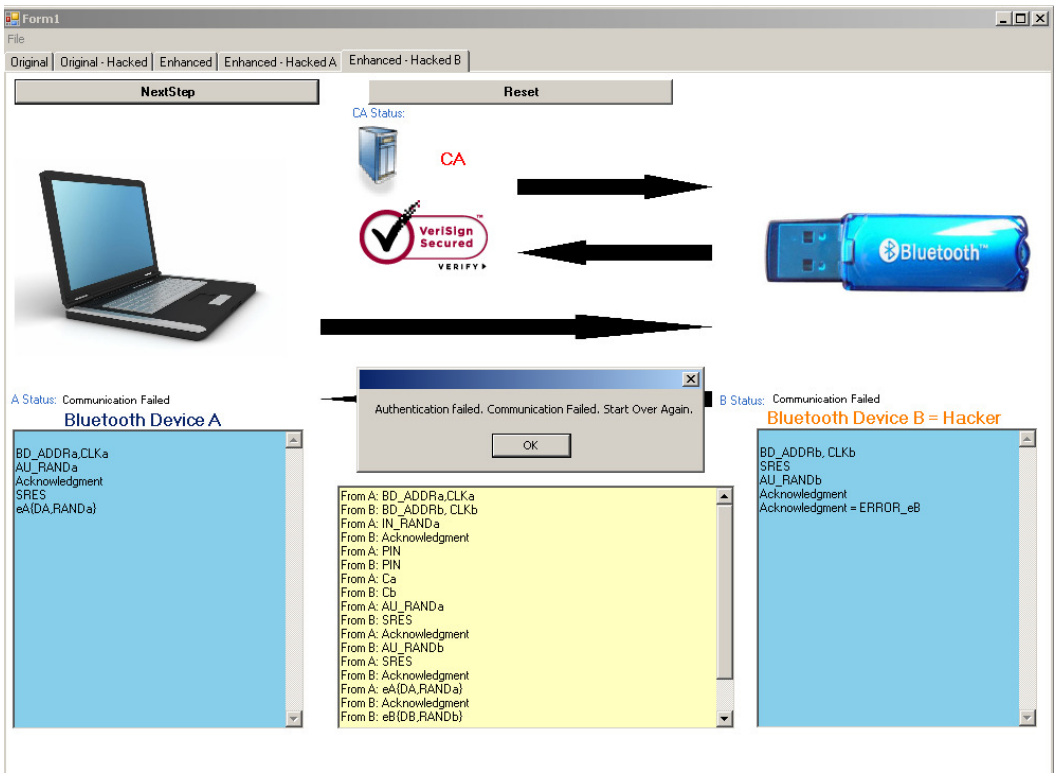


Figure 7.1.4: Enhanced-Hacked Simulation for Hacker B.

Chapter 8: Conclusion

Throughout this research, it was amply understood how Bluetooth manages its pairing and authentication phases in order to guarantee safe pairings with other devices. However, it was also understood the reasons why Bluetooth security has still some flaws in the authentication method that lead to impersonation attacks.

These impersonation attacks were carefully researched in order to be capable of having a successfully removal of the same. This removal was accomplished by adding one more step in the Bluetooth authentication procedure; the use of a Certificate Authority that ensured the authenticity of a digital certificate, and in consequence, that proved the identity of the device to be paired.

This research was concluded on diagram analysis and software simulations that compared and showed the original Bluetooth pairing protocol to the one proposed on this document. Both analyses were properly exposed to an impersonation attacks proving that the new method did protect against these attacks, on the contrary to the Bluetooth original pairing protocol.

References

[1] G. F. Gregory Lamm, Jorge Estrada, Jag Gadiyaram, "Bluetooth Wireless Networks Security Features," in *Workshop of Information Assurance and Security*, United States Military Academy, West Point, NY, 2001, pp. 265-272.

[2] S. I. G.-. SIG, "Specification of the Bluetooth System," in *Core V2.1 + EDR*. vol. 0-4: Bluetooth Org, 2007, p. 1420.

[3] d. M. Karl E Persson, "Secure Connections in Bluetooth Scatternets," in *36th Hawaii International Conference on System Sciences*, Hawaii, 2002, p. 10.

[4] L. O. Tom Karygiannis, "Wireless Network Security, 802.11, Bluetooth and Handheld Devices.," NIST National Institute of Standards and Technology, 2002, p. 119.

[5] G. M. David Kammer, Brian Senese, Jennifer Bray., *Bluetooth, Application Developer's Guide: The Shoert Range Interconenct Solution.*, 1st ed. Rockland, MA: Syngress, 2002.

[6] J. T. Vainio, "Bluetooth Security," p. 14, 2000.

[7] P. Hall, "What is Bluetooth?," Developer.com, 2002.

[8] M. Guizani, "Bluetooth: a Survey on the Technology and Security," in *China Communications*, China, 2004, p. 6.

[9] H. A. Houda Labiod, Constantino De Santis, *Wi-Fi, Bluetooth, ZigBee, and WiMax.*, 1st ed. Evry, France: Springer, 2007.

[10] A. Kansal, "Bluetooth Primer."

[11] M. Ganguli, *Getting Started with Bluetooth*, 1st ed. Cincinnati, Ohio: Premier Press, 2002.

[12] N. J. Muller, *Bluetooth Demystified*, 1st ed.: McGraw-Hill, 2000.

[13] M. Miller, *Discovering Bluetooth*, 1st ed.: Sybex Inc, 2001.

[14] S. Janssens, "Preliminary Study: Bluetooth Security," 2005, p. 28.

[15] A. W. Yaniv Shaked, "Cracking the Bluetooth PIN," Seattle, WA: ACM NY, 2005, pp. 39-50.

[16] C. Rhodes, "Bluetooth Security," www.infosecwriters.com.

[17] T. Yang, "Bluetooth Security," University Of Tennessee, Knoxville, Tennessee 2000 2000.

[18] X. Z. Nicolas Sklavos, *Wireless security and cryptography*, 1st ed.: CRC Press, 2007.

[19] J. P. Christian Gehrmann, Ben Smeets, *Bluetooth Security*, 1st ed. Boston, London: Artech House, 2004.

[20] H. Bidgoli, *Handbook of Information Security Volume 3*, 1st ed. vol. 3. Berkeley, California: John Wiley & Sons, Inc., 2006.

[21] F. S. Ford-Long Wong, Jolyon Clulow, "Repairing the Bluetooth Pairing Protocol," B. Christianson et al ed: Security Protocols, LNCS, 2005, p. 17.

[22] H. Bidgoli, *Handbook of Information Security Volume 1*, 1st ed. vol. 1. Berkeley, California: John Wiley & Sons, Inc., 2006.

[23] P. Robert Morrow, *Bluetooth Operation and Use*, 1st ed. vol. 1st ed: McGraw-Hill Professional, 2002.

[24] W. Starllings, *Data and Computer Communications*, 7th ed. New Jersey: Pearson Education, 2004.

[25] P. G. Lawrence Miller, *Security+ Certification for Dummies*, 1st ed. NY: Wiley Publishing, 2003.

Glossary

Algorithm: a procedure or formula for solving a problem. A computer program can be viewed as an elaborate algorithm. In mathematics and computer science, an algorithm usually means a small procedure that solves a recurrent problem.

Authentication: Authentication is the process of determining whether someone or something is, in fact, who or what it is declared to be.

BD_ADDR: The Bluetooth Device Address, BD_ADDR, is used to identify a Bluetooth device.

Bitwise Operation: In computer programming, a bitwise operation operates on one or two bit patterns or binary numerals at the level of their individual bits. On most older microprocessors, bitwise operations are slightly faster than addition and subtraction operations and usually significantly faster than multiplication and division operations. Example of Bitwise operations: NOT, OR, XOR, AND.

Bluetooth: Bluetooth is a wireless communication link, operating in the unlicensed ISM band at 2.4 GHz using a frequency hopping transceiver. It allows real-time AV and data communications between Bluetooth Hosts. The link protocol is based on time slots.

Bluetooth Baseband: The part of the Bluetooth system that specifies or implements the medium access and physical layer procedures to support the exchange of real-time voice, data information streams, and ad hoc networking between Bluetooth Devices.

Bluetooth Clock: A 28 bit clock internal to a Bluetooth controller sub-system that ticks every 312.5µs. The value of this clock defines the slot numbering and timing in the various physical channels.

Bluetooth Controller: A sub-system containing the Bluetooth RF, baseband, resource controller, link manager, device manager and a Bluetooth HCI.

Bluetooth Device: A Bluetooth Device is a device that is capable of short-range wireless communications using the Bluetooth system.

Bluetooth Device Address: A 48 bit address used to identify each Bluetooth Device.

Bluetooth HCI: The Bluetooth Host Controller Interface (HCI) provides a command interface to the baseband controller and link manager and access to hardware status and control registers. This interface provides a uniform method of accessing the Bluetooth baseband capabilities.

Bluetooth Host: A Bluetooth Host is a computing device, peripheral, cellular telephone, access point to PSTN network or LAN, etc. A Bluetooth Host attached to a Bluetooth Controller may communicate with other Bluetooth Hosts attached to their Bluetooth Controllers as well.

Certificate Authority: A certificate authority (CA) is an authority in a network that issues and manages security credentials and public keys for message encryption. As part of a public key infrastructure (PKI), a CA checks with a registration authority (RA) to verify information provided by the requestor of a digital certificate. If the RA verifies the requestor's information, the CA can then issue a certificate.

Challenge: The act of a sentry in halting any one who appears at his post, and demanding the countersign. It is a claim or demand, a calling into question, a demanding of proof, explanation, etc.

Connect (to service): The establishment of a connection to a service. If not already done, this also includes establishment of a physical link, logical transport, logical link and L2CAP channel.

Connectable device: A Bluetooth device in range that periodically listens on its page scan physical channel and will respond to a page on that channel.

Connecting: A phase in the communication between devices when a connection between them is being established. (Connecting phase follows after the link establishment phase is completed.)

Coverage area: The area where two Bluetooth devices can exchange messages with acceptable quality and performance.

Creation of a secure connection: A procedure of establishing a connection, including authentication and encryption.

Creation of a trusted relationship: A procedure where the remote device is marked as a trusted device. This includes storing a common link key for future authentication and pairing (if the link key is not available).

Decryption: Decryption is the process of converting encrypted data back into its original form, so it can be understood.

Device discovery: A procedure for retrieving the Bluetooth device address, clock, class-of-device field and used page scan mode from discoverable devices.

Digital Certificate: Depending on the public key infrastructure implementation, the certificate includes the owner's public key, the expiration date of the certificate, the owner's name, and other information about the public key owner.

Digital Signature: A digital signature (not to be confused with a digital certificate) is an electronic signature that can be used to authenticate the identity of the sender of a message or the signer of a document, and possibly to ensure that the original content of the message or document that has been sent is unchanged. Digital signatures are easily transportable, cannot be imitated by someone else, and can be automatically time-stamped. The ability to ensure that the original signed message arrived means that the sender cannot easily repudiate it later.

A digital signature can be used with any kind of message, whether it is encrypted or not, simply so that the receiver can be sure of the sender's identity and that the message arrived intact. A digital certificate contains the digital signature of the certificate-issuing authority so that anyone can verify that the certificate is real.

Discoverable device: A Bluetooth device in range that periodically listens on an inquiry scan physical channel and will respond to an inquiry on that channel. Discoverable device are normally also connectable.

Eavesdropping: Eavesdropping is the act of surreptitiously listening to a private conversation. This is commonly thought to be unethical and there is an old adage that eavesdroppers seldom hear anything good of themselves.

Encryption: Encryption is the conversion of data into a form, called a ciphertext, which cannot be easily understood by unauthorized people.

Impersonation: To assume the character or appearance of, especially fraudulently. As the name implies, involves gaining access to a system by impersonating a legitimate user—a feat that usually requires knowing or guessing a legitimate user's password.

Impersonator: An impersonator is someone who imitates or copies the behavior or actions of another.

Inquiry: A procedure where a Bluetooth device transmits inquiry messages and listens for responses in order to discover the other Bluetooth devices that are within the coverage area.

Key: In cryptography, a key is a variable value that is applied using an algorithm to a string or block of unencrypted text to produce encrypted text, or to decrypt encrypted text. The length of the key is a factor in considering how difficult it will be to decrypt the text in a given message.

Known device: A Bluetooth device for which at least the BD_ADDR is stored.

L2CAP Channel: A logical connection on L2CAP level between two devices serving a single application or higher layer protocol.

Link: Shorthand for a logical link.

Link establishment: A procedure for establishing the default ACL link and hierarchy of links and channels between devices.

Link key: A secret key that is known by two devices and is used in order to authenticate each device to the other.

Logical link: The lowest architectural level used to offer independent data transport services to clients of the Bluetooth system.

Logical transport: Used in Bluetooth to represent commonality between different logical links due to shared acknowledgement protocol and link identifiers.

Master: A master, in a technological context, is a device that controls one or more other devices. In networking, for example, a master/slave configuration is a communications model in which one device or process (known as the master) controls one or more other devices or processes (known as slaves).

Master/Slave: In computer networking, master/slave is a model for a communication protocol in which one device or process (known as the master) controls one or more other devices or processes (known as slaves). Once the master/slave relationship is established, the direction of control is always from the master to the slave(s).

Name discovery: A procedure for retrieving the user-friendly name (the Bluetooth device name) of a connectable device.

Page: The initial phase of the connection procedure where a device transmits a train of page messages until a response is received from the target device or a timeout occurs.

Paging device: A Bluetooth device that is carrying out the page procedure.

Paired device: A Bluetooth device with which a link key has been exchanged (either before connection establishment was requested or during connecting phase).

Pairing: Pairs of devices may establish a relationship by creating a shared secret known as a link key; this process is known as pairing.

Physical Channel: Characterized by synchronized occupancy of a sequence of RF carriers by one or more devices. A number of physical channel types exist with characteristics defined for their different purposes.

Piconet: A collection of devices occupying a shared physical channel where one of the devices is the Piconet Master and the remaining devices are connected to it.

PIN: A user-friendly number that can be used to authenticate connections to a device before pairing has taken place.

Private Key: In cryptography, a private or secret key is an encryption/decryption key known only to the party or parties that exchange secret messages. In traditional secret key cryptography, a key would be shared by the communicators so that each could encrypt and decrypt messages. The risk in this system is that if either party loses the key or it is stolen, the system is broken. A more recent alternative is to use a combination of public and private keys. In this system, a public key is used together with a private key. See public key infrastructure (PKI) for more information.

Public Key: In cryptography, a public key is a value provided by some designated authority as an encryption key that, combined with a private key derived from the public key, can be used to effectively encrypt messages and digital signatures.

Registration Authority: A registration authority (RA) is an authority in a network that verifies user requests for a digital certificate and tells the certificate authority (CA) to issue it. RAs are part of a public key infrastructure (PKI), a networked system that enables companies and users to exchange information and money safely and securely. The digital certificate contains a public key that is used to encrypt and decrypt messages and digital signatures.

Shared Secret: In cryptography, a shared secret is a piece of data only known to the parties involved in a secure communication. The shared secret can be a password, a passphrase, a big number or an array of randomly chosen bytes.

Unknown device: A Bluetooth device for which no information (Bluetooth Device Address, link key or other) is stored.

Appendix A: Ciphering algorithms

INNER DESIGN FOR E_{22}

As described in subsection 4.1.2.1 of this document, E_{22} is used to generate the initialization key or Kinit. The inputs used are:

- a BD_ADDR (48 bits long),
- the PIN Code and its length,
- a 128-bit random number IN_RAND.

The PIN and the BD_ADDR are combined to create a new word. If the PIN contains less that 16 bytes, some of the BD_ADDR bytes are appended to the PIN. If the PIN is less than 10 bytes long, all bytes of BD_ADDR are used. Let PIN' denote the new word created, and L' denotes the number of bytes the new word contains. If L' is less than 16, the new word is cyclic expanded till it contains 16 bytes. Let PIN" denote this second new word. PIN" is used as the 128 bit input key of A_r'. IN_RAND is used as the 128 bit input data, after XORing the most significant byte with L'. The following figure A-1 shows the inner design of this E_{22} algorithm.

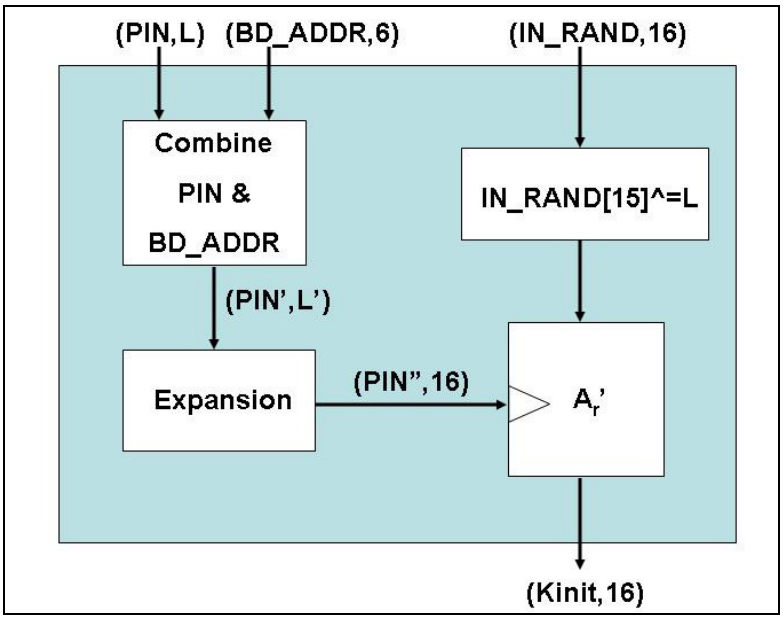


Figure A-1: Inner Design of E_{22}

INNER DESIGN FOR E_{21}

As described in subsection 4.1.2.3, E_{21} is used to generate the link key. The inputs used are:

- a BD_ADDR (48 bits long),
- a 128 bit random number LK_RAND.

The BD_ADDR is cyclic expanded to form a 128 bit word which is used as the input data of A_r'. The key used for A_r consists of the 128 bit random number LK_RAND, after XORing its most significant byte with 6(result denoted LK_RAND'). The following figure A-2 shows the inner design of this E_{21} algorithm.

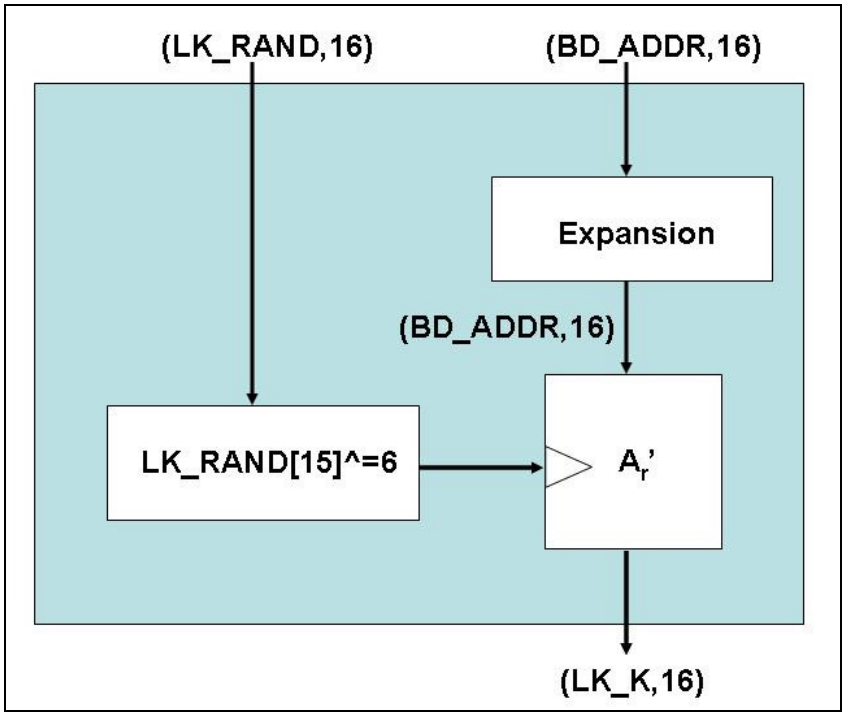


Figure A-2: Inner Design of E_{21}

INNER DESIGN FOR E_1

As described in section 5.2, E_1 is used to perform the mutual-authentication or challenge response. The inputs used are:

- A random word AU_RANDA,
- The link key Kab,
- A BD_ADDR (48 bits long).

The inner design of E_1 contains both A_r and A_r', and the link key used twice. Once, it is supplied as is for the key input of A_r. Later, it goes through a transformation denoted Offset and supplier as the key input of A_r'. The Offset transformation consists of adding and XORing its bytes with some constants. As for the BD_ADDR, it is cyclic expanded to form a 128 bit word denoted BD_ADDR'. The following figure A-3 shows the inner design of this E_{21} algorithm.

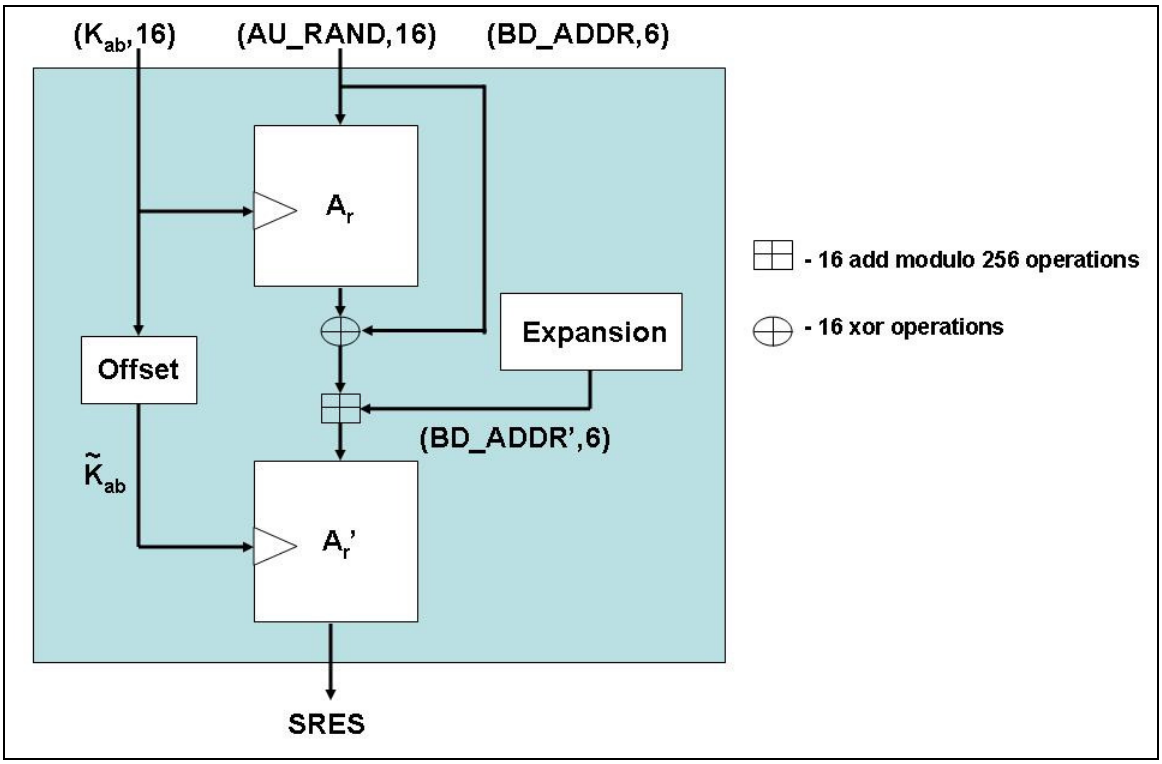
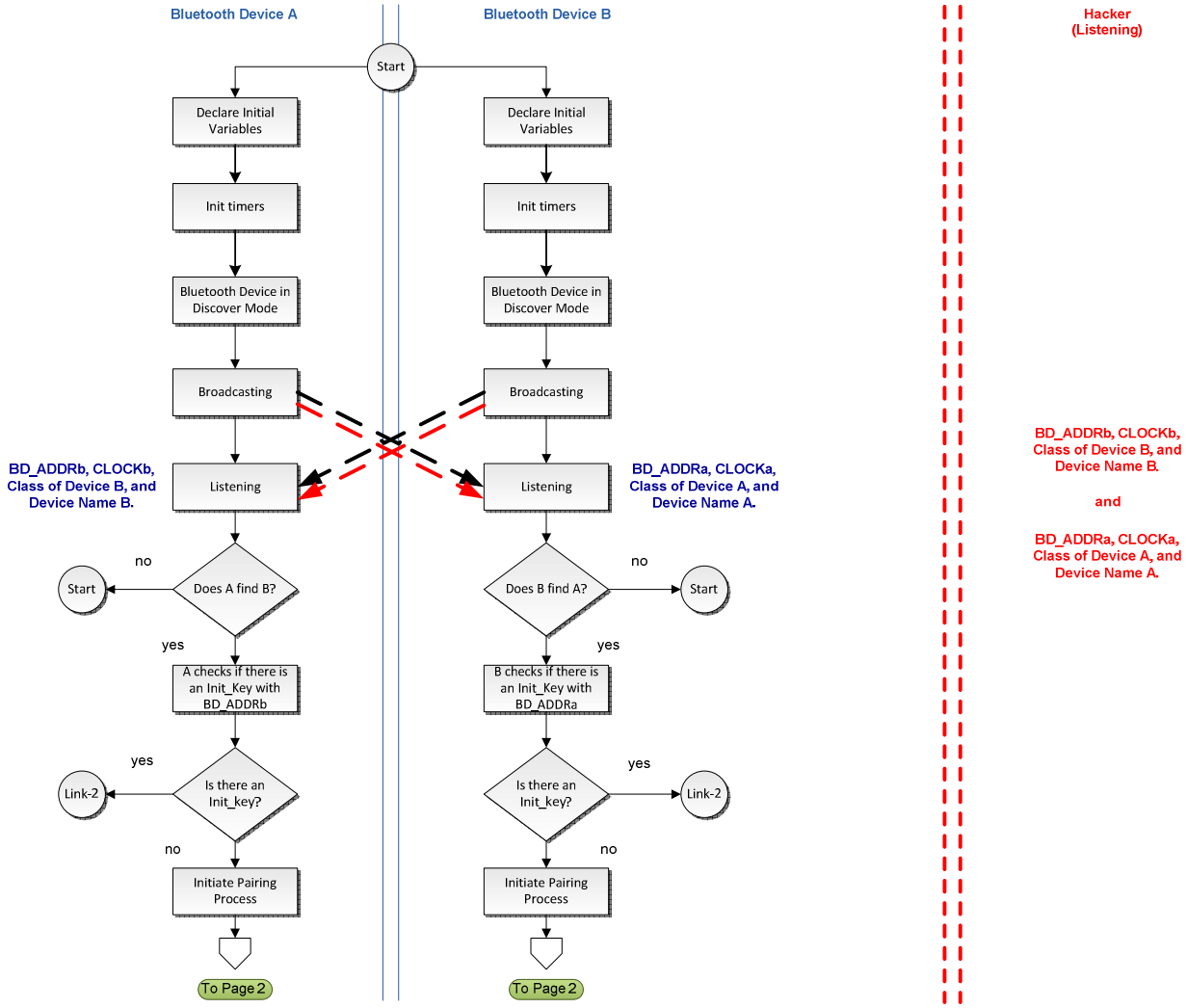
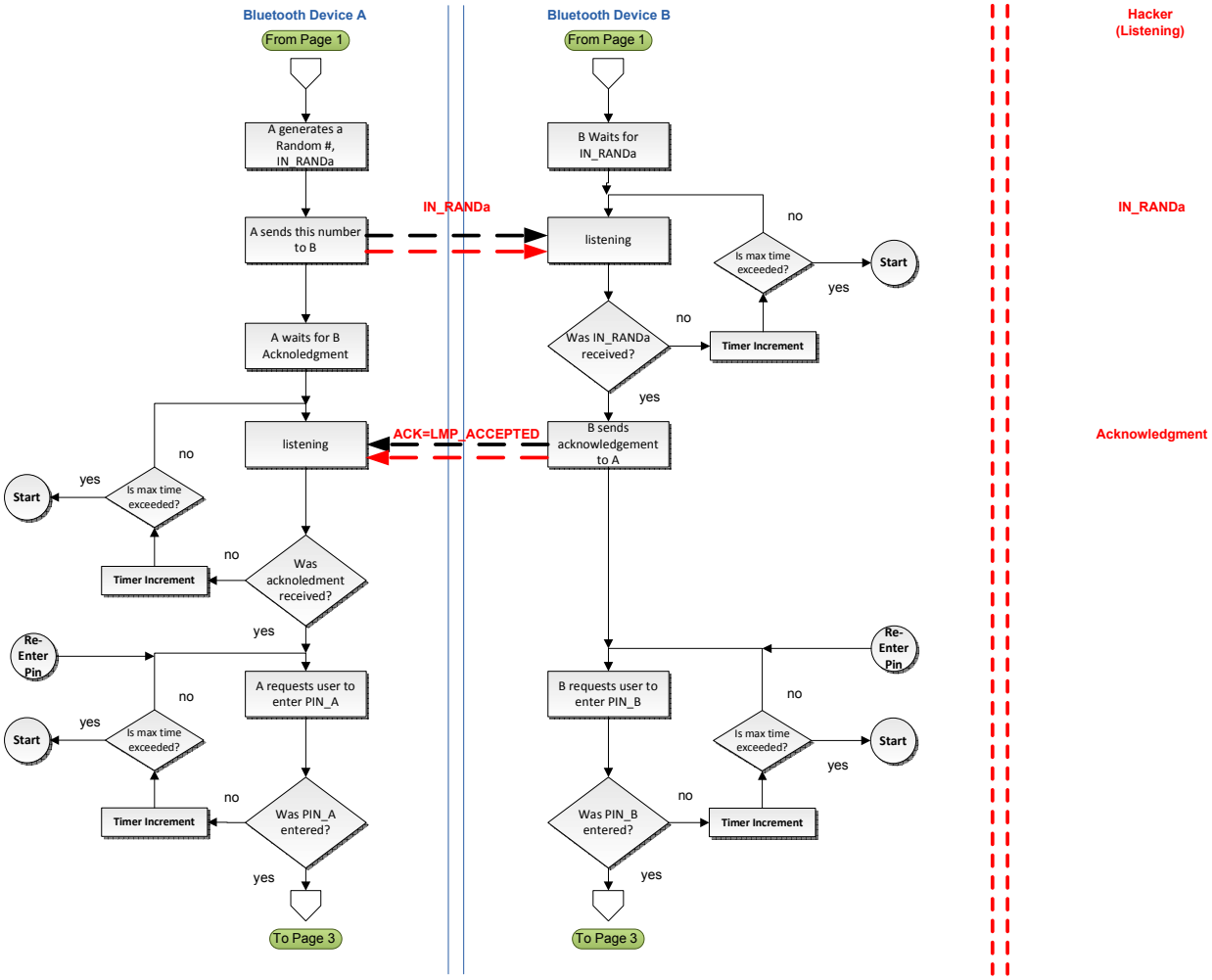


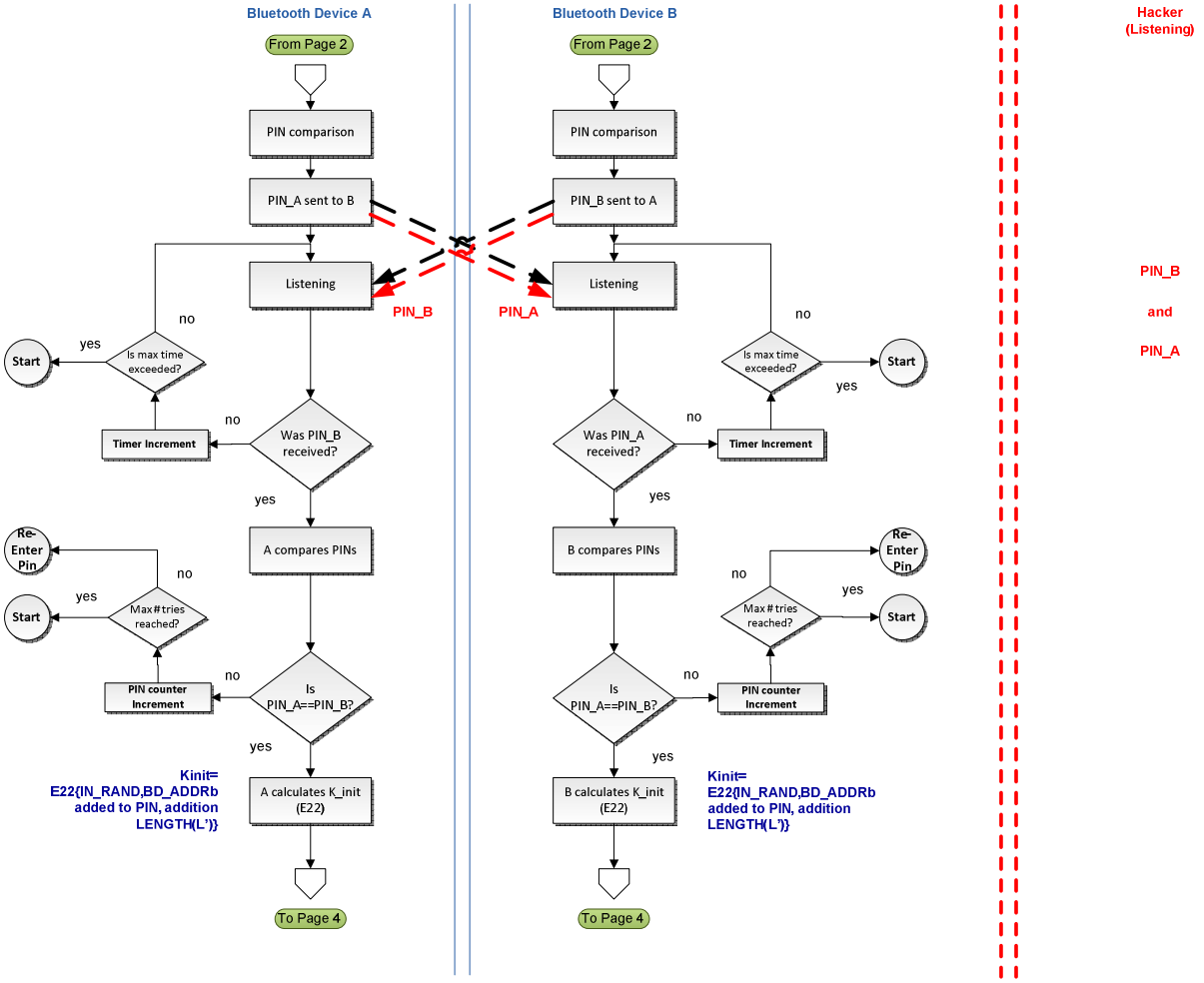
Figure A-3: Inner Design of E_1

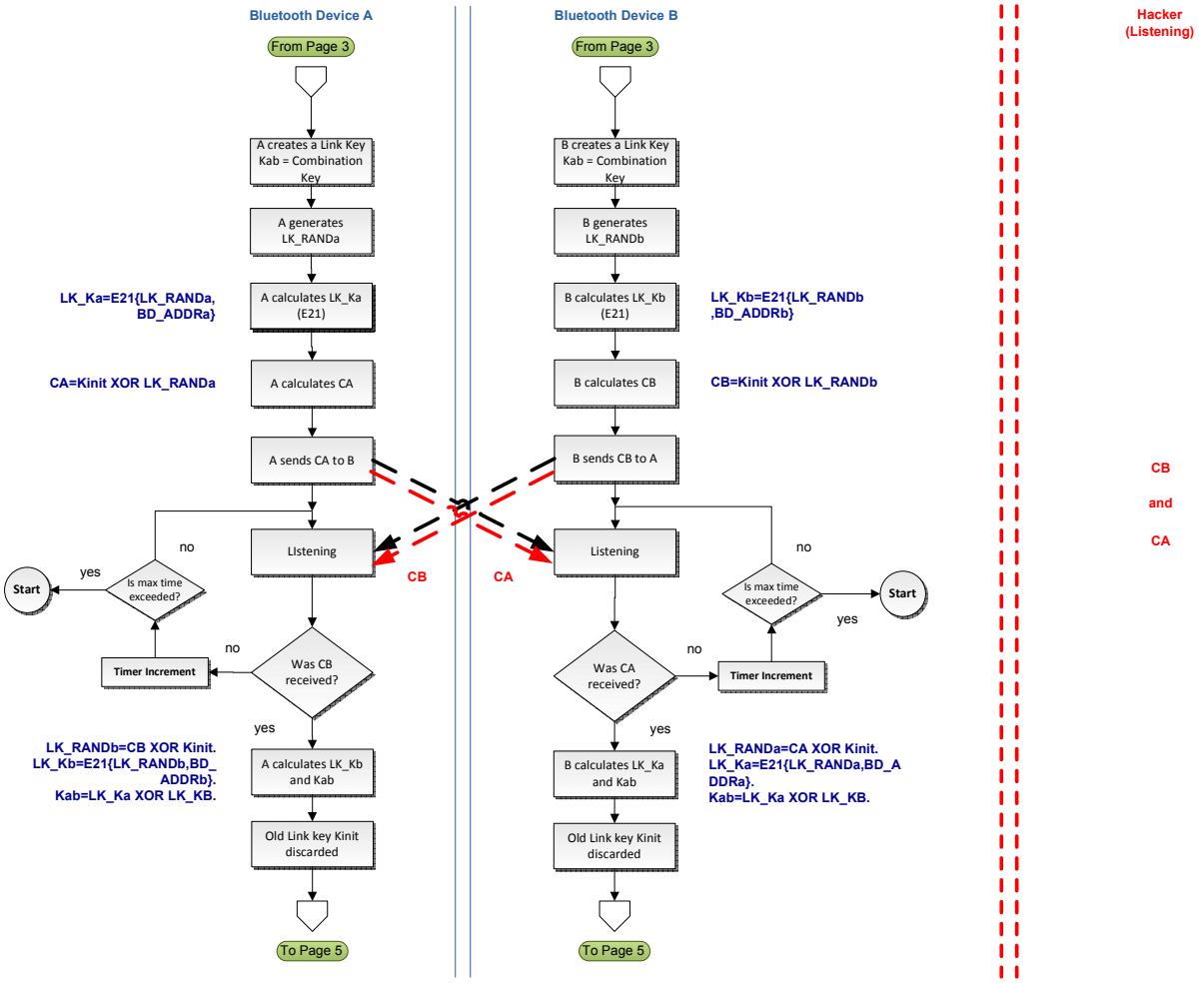
Appendix B: Flow Diagrams

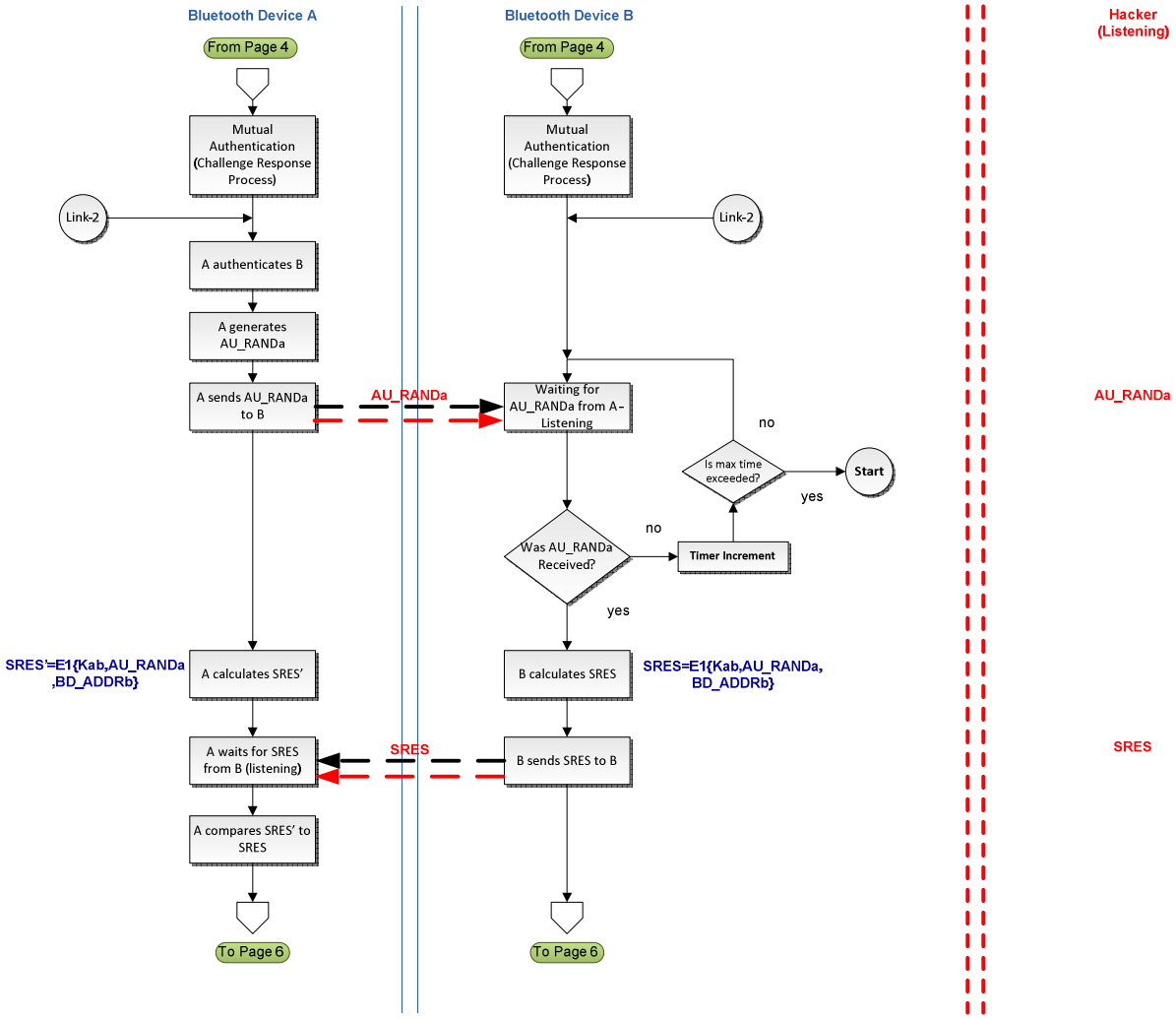
ORIGINAL-METHOD FLOW DIAGRAM

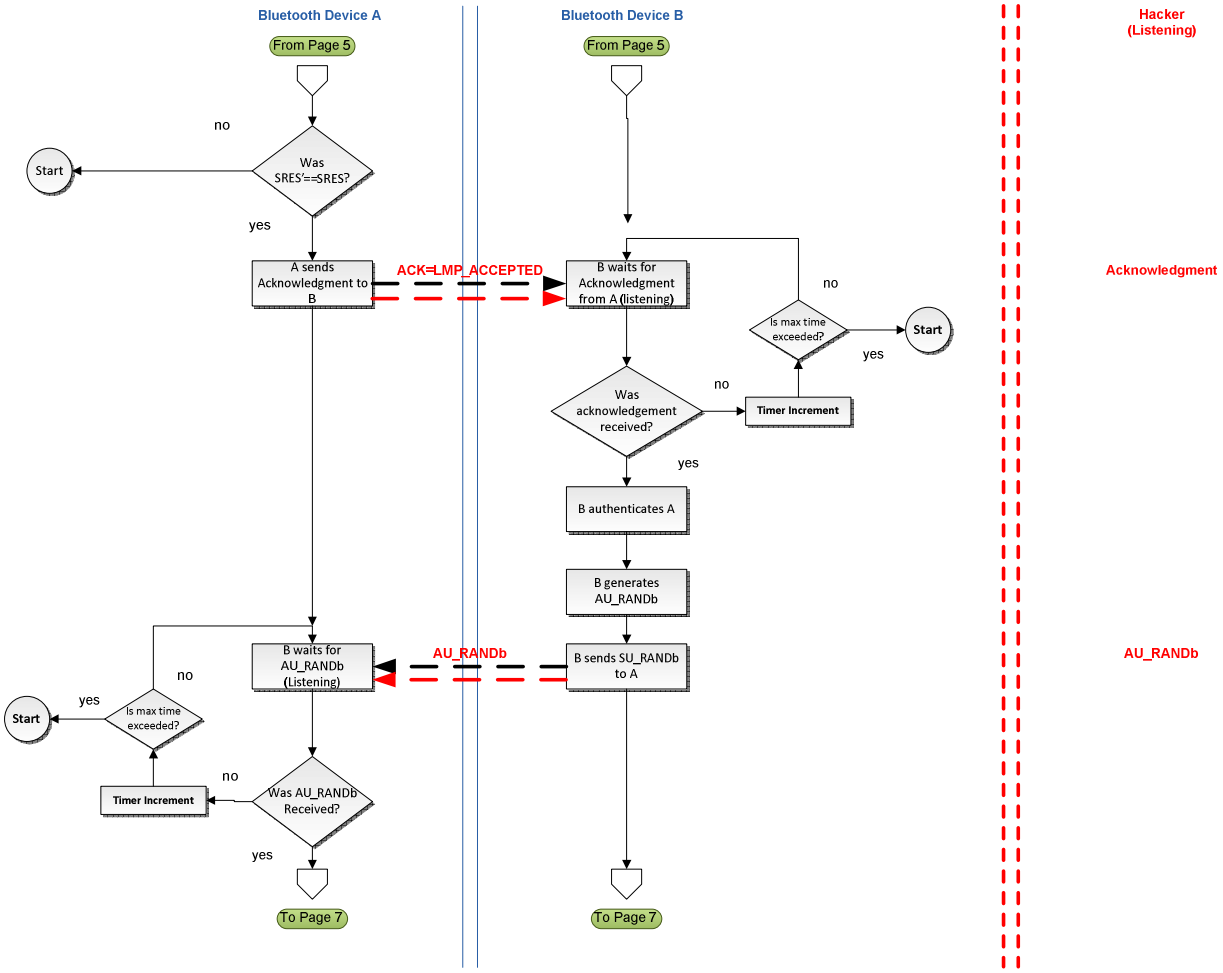


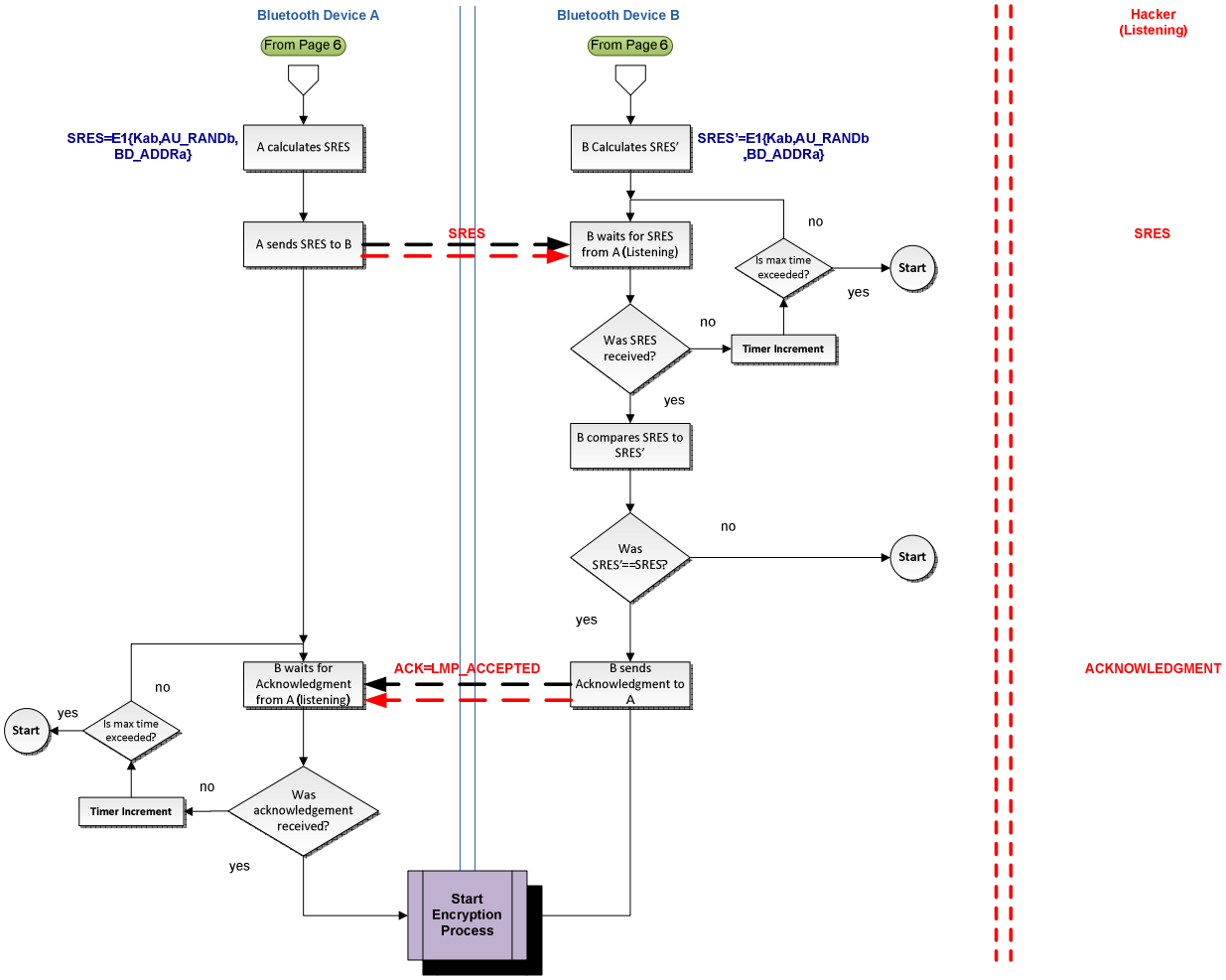

Bluetooth Device A
Bluetooth Device B
Hacker (Listening)
From Page 1
From Page 1
A generates a Random #, IN_RANDa
B Waits for IN_RANDa
IN_RANDa
IN_RANDa
A sends this number to B
listening
no
Is max time exceeded?
Start
yes
A waits for B Acknoledgment
Was IN_RANDa received?
no
Timer Increment
yes
listening
ACK=LMP_ACCEPTED
B sends acknowledgement to A
Acknowledgment
no
Start
yes
Is max time exceeded?
Timer Increment
no
Was acknoledment received?
yes
Re-Enter Pin
A requests user to enter PIN_A
B requests user to enter PIN_B
Re-Enter Pin
no
Start
yes
Is max time exceeded?
Timer Increment
no
Was PIN_A entered?
yes
Was PIN_B entered?
no
Timer Increment
yes
To Page 3
To Page 3


Bluetooth Device A
Bluetooth Device B
Hacker (Listening)
From Page 2
From Page 2
PIN comparison
PIN comparison
PIN_A sent to B
PIN_B sent to A
Listening
Listening
PIN_B
PIN_A
PIN_B and PIN_A
no
yes
Is max time exceeded?
Start
Timer Increment
Was PIN_B received?
Was PIN_A received?
A compares PINs
B compares PINs
Re-Enter Pin
Max # tries reached?
PIN counter Increment
Is PIN_A==PIN_B?
Kinit= E22{IN_RAND,BD_ADDRb added to PIN, addition LENGTH(L')}
A calculates K_init (E22)
B calculates K_init (E22)
To Page 4
To Page 4

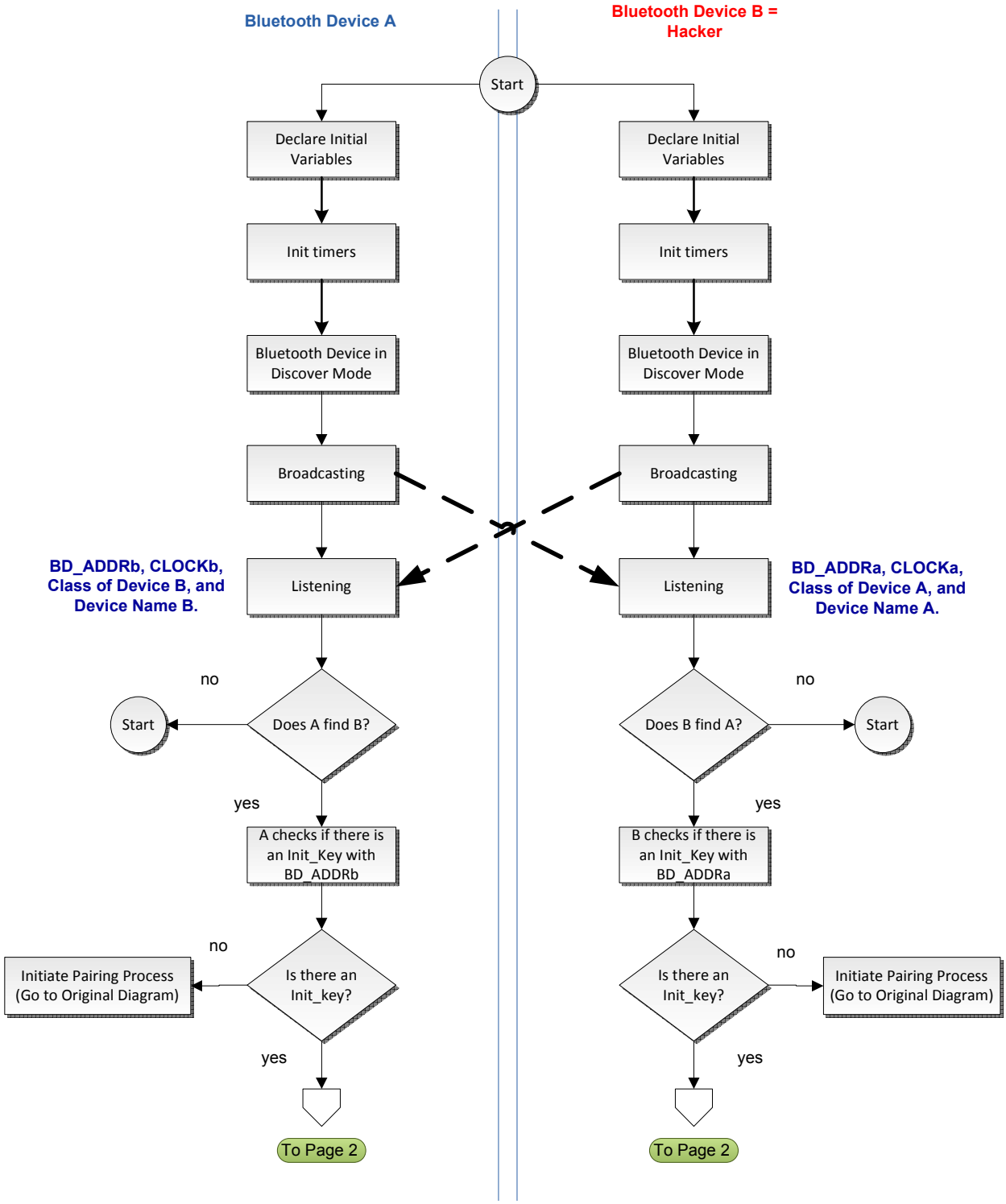

Bluetooth Device A
Bluetooth Device B
Hacker (Listening)
From Page 3
From Page 3
A creates a Link Key Kab = Combination Key
B creates a Link Key Kab = Combination Key
A generates LK_RANDa
B generates LK_RANDb
LK_Ka=E21{LK_RANDa, BD_ADDRa}
A calculates LK_Ka (E21)
B calculates LK_Kb (E21)
LK_Kb=E21{LK_RANDb ,BD_ADDRb}
CA=Kinit XOR LK_RANDa
A calculates CA
B calculates CB
CB=Kinit XOR LK_RANDb
A sends CA to B
B sends CB to A
CB
and
CA
Listening
Listening
no
yes
Start
Is max time exceeded?
CB
CA
Is max time exceeded?
no
yes
Start
Timer Increment
no
Was CB received?
Was CA received?
no
Timer Increment
yes
yes
LK_RANDb=CB XOR Kinit.
LK_Kb=E21{LK_RANDb,BD_ ADDRb}.
Kab=LK_Ka XOR LK_KB.
A calculates LK_Kb and Kab
B calculates LK_Ka and Kab
LK_RANDa=CA XOR Kinit.
LK_Ka=E21{LK_RANDa,BD_A DDRa}.
Kab=LK_Ka XOR LK_KB.
Old Link key Kinit discarded
Old Link key Kinit discarded
To Page 5
To Page 5

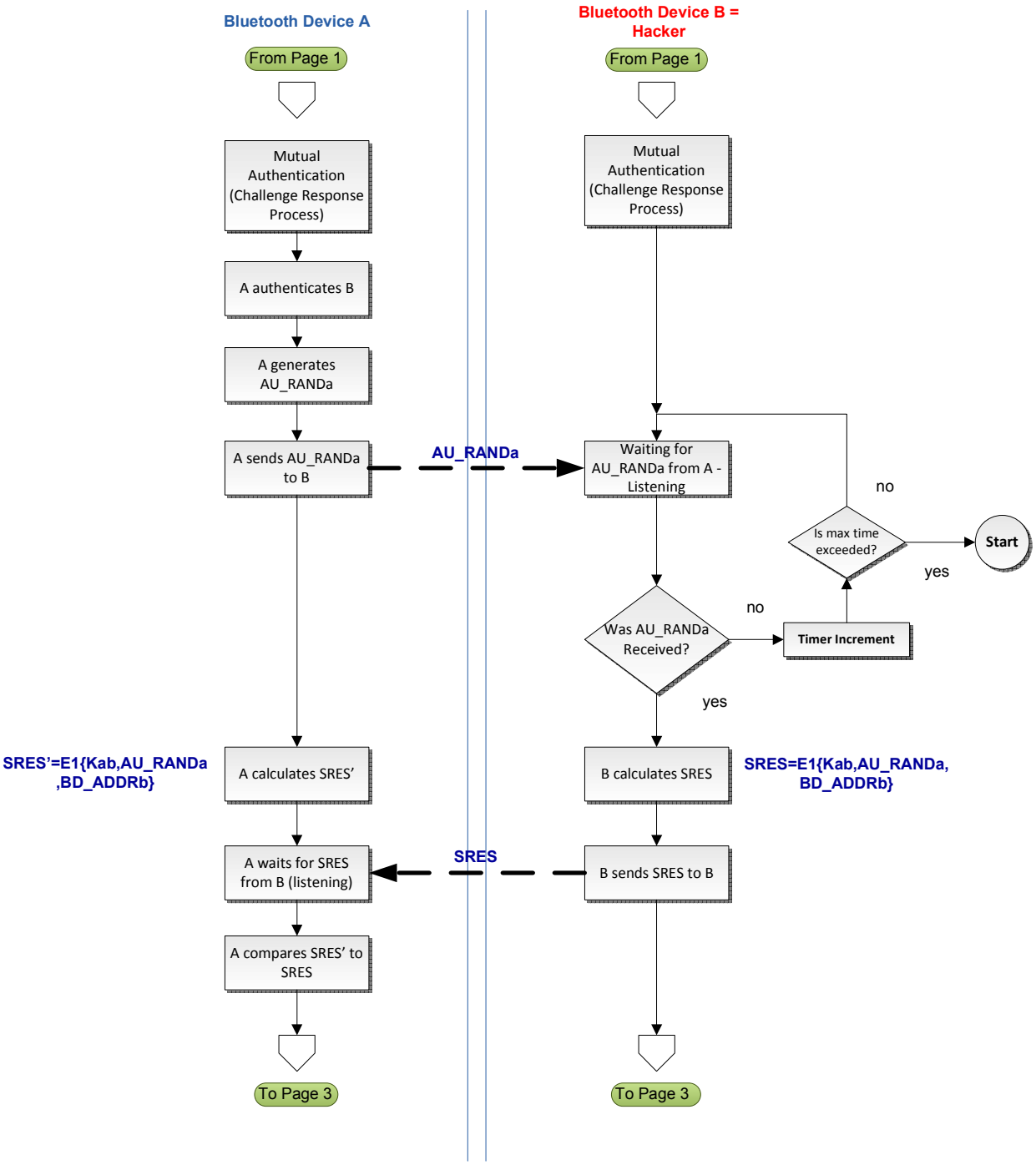

Bluetooth Device A
Bluetooth Device B
Hacker (Listening)
From Page 4
From Page 4
Mutual Authentication (Challenge Response Process)
Mutual Authentication (Challenge Response Process)
Link-2
Link-2
A authenticates B
A generates AU_RANDa
A sends AU_RANDa to B
AU_RANDa
Waiting for AU_RANDa from A - Listening
AU_RANDa
no
Is max time exceeded?
Start
yes
Was AU_RANDa Received?
no
Timer Increment
yes
SRES'=E1{Kab,AU_RANDa ,BD_ADDRb}
A calculates SRES'
B calculates SRES
SRES=E1{Kab,AU_RANDa, BD_ADDRb}
A waits for SRES from B (listening)
SRES
B sends SRES to B
SRES
A compares SRES' to SRES
To Page 6
To Page 6

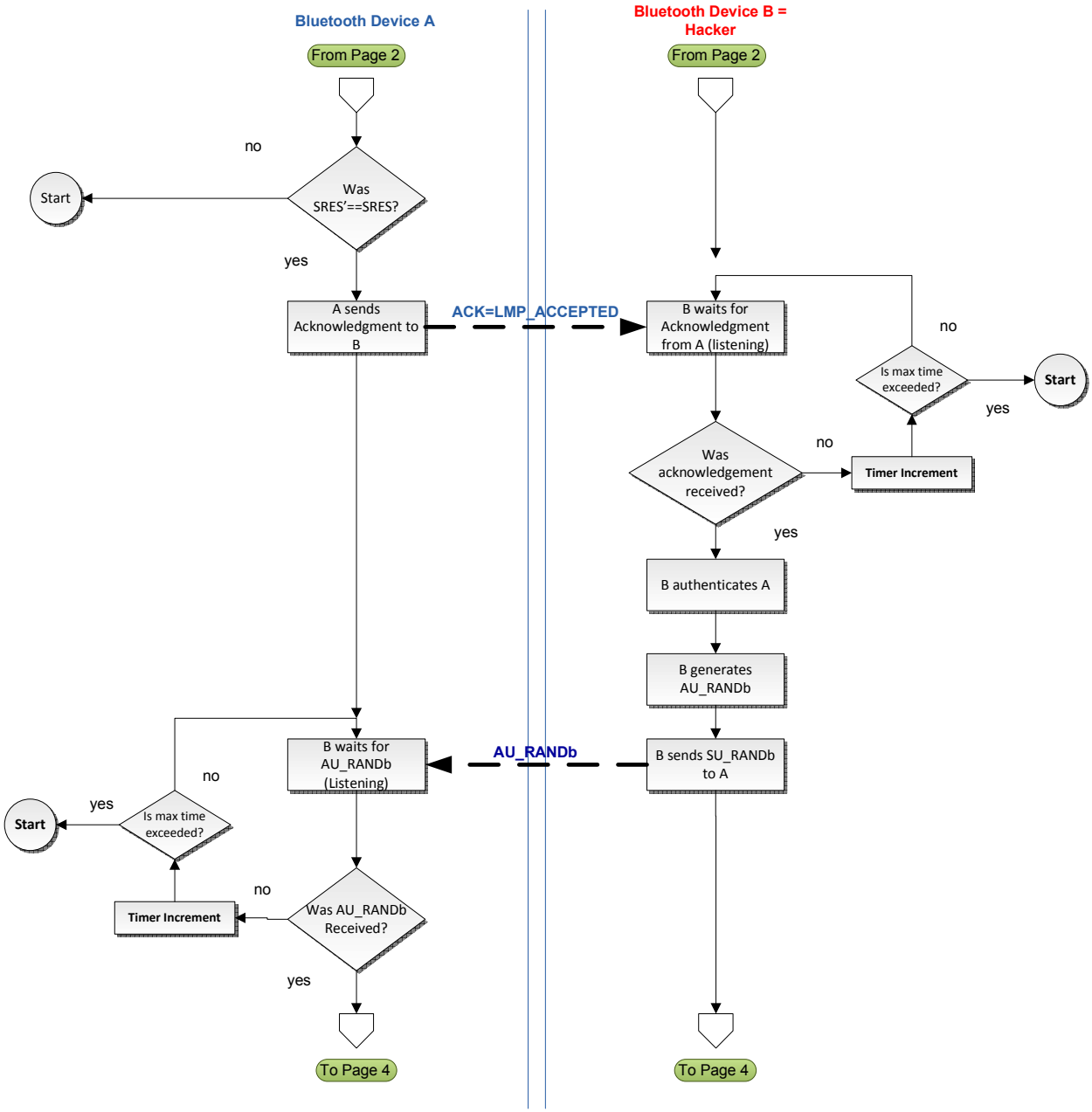

Bluetooth Device A
Bluetooth Device B
Hacker (Listening)
From Page 5
From Page 5
no
Was SRES'==SRES?
Start
yes
A sends Acknowledgment to B
ACK=LMP_ACCEPTED
B waits for Acknowledgment from A (listening)
Acknowledgment
no
Is max time exceeded?
Start
yes
Was acknowledgement received?
no
Timer Increment
yes
B authenticates A
B generates AU_RANDb
B waits for AU_RANDb (Listening)
AU_RANDb
B sends SU_RANDb to A
AU_RANDb
no
yes
Is max time exceeded?
Start
no
Timer Increment
Was AU_RANDb Received?
yes
To Page 7
To Page 7

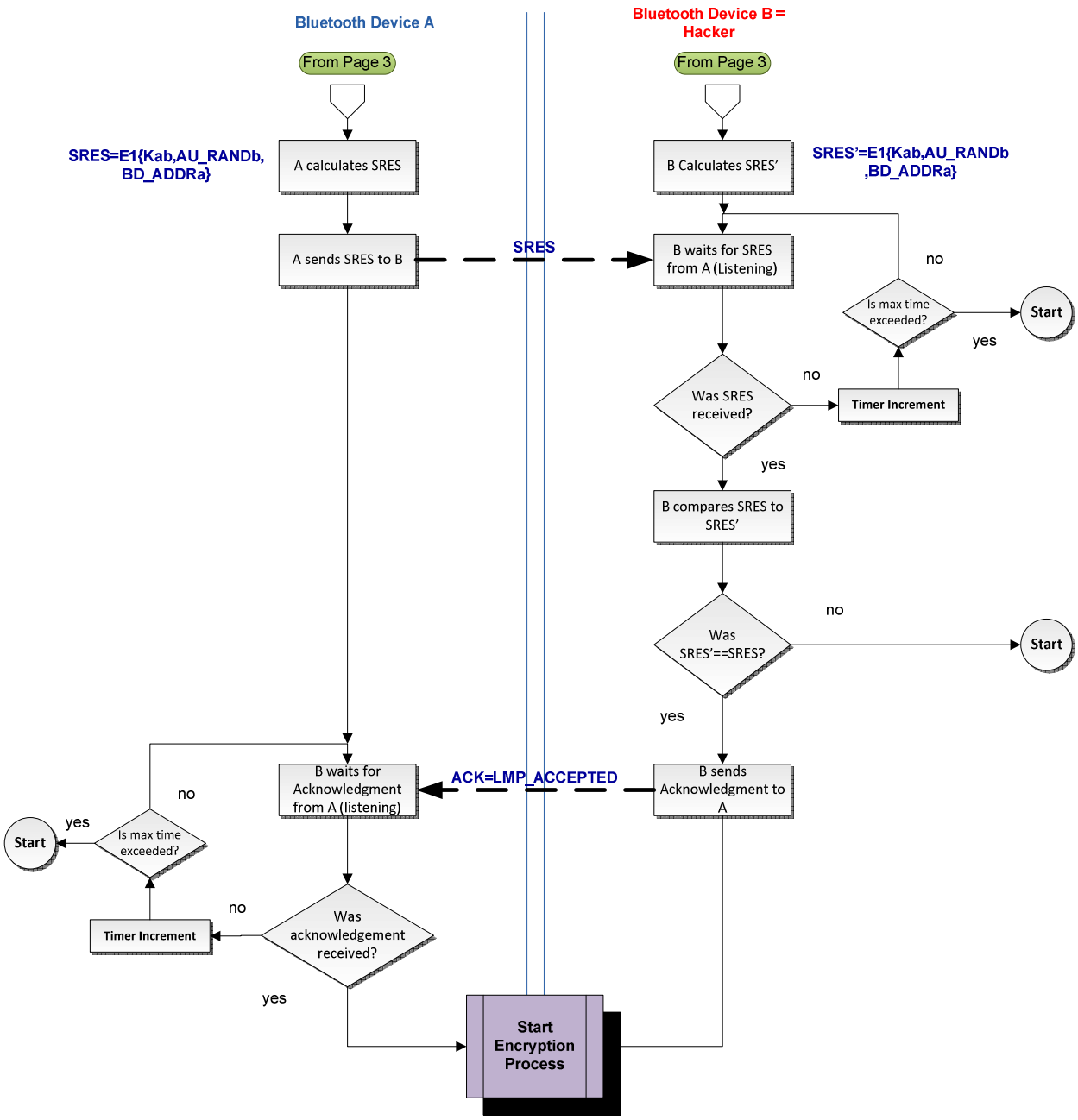

Bluetooth Device A
Bluetooth Device B
Hacker (Listening)
From Page 6
From Page 6
SRES=E1{Kab,AU_RANDb, BD_ADDRa}
A calculates SRES
B Calculates SRES'
SRES'=E1{Kab,AU_RANDb ,BD_ADDRa}
A sends SRES to B
SRES
B waits for SRES from A (Listening)
SRES
no
Is max time exceeded?
Start
yes
Was SRES received?
no
Timer Increment
yes
B compares SRES to SRES'
Was SRES'==SRES?
no
Start
yes
B waits for Acknowledgment from A (listening)
ACK=LMP_ACCEPTED
B sends Acknowledgment to A
ACKNOWLEDGMENT
no
yes
Start
Is max time exceeded?
no
Timer Increment
Was acknowledgement received?
yes
Start Encryption Process

ORIGINAL-HACKED-METHOD FLOW DIAGRAM

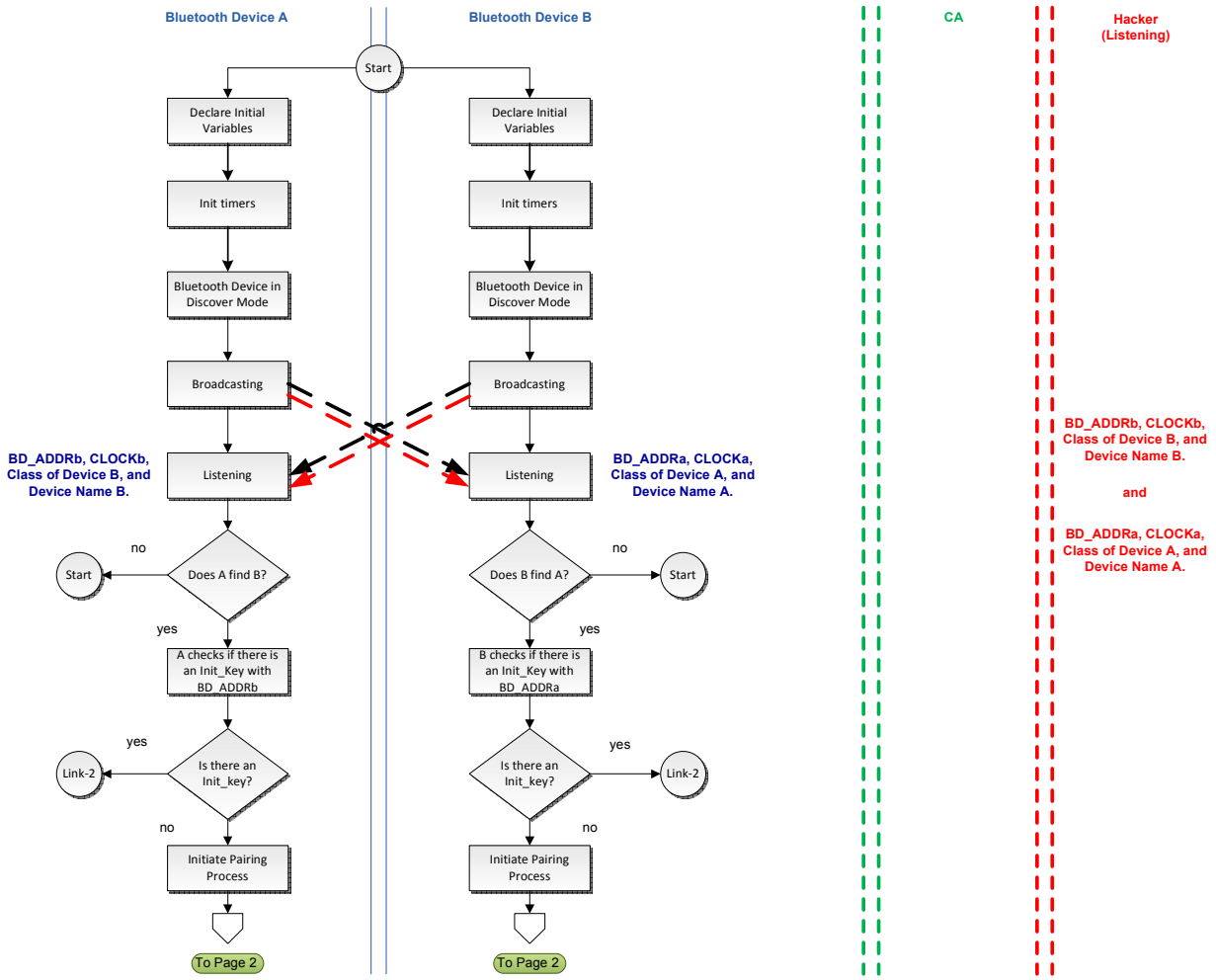


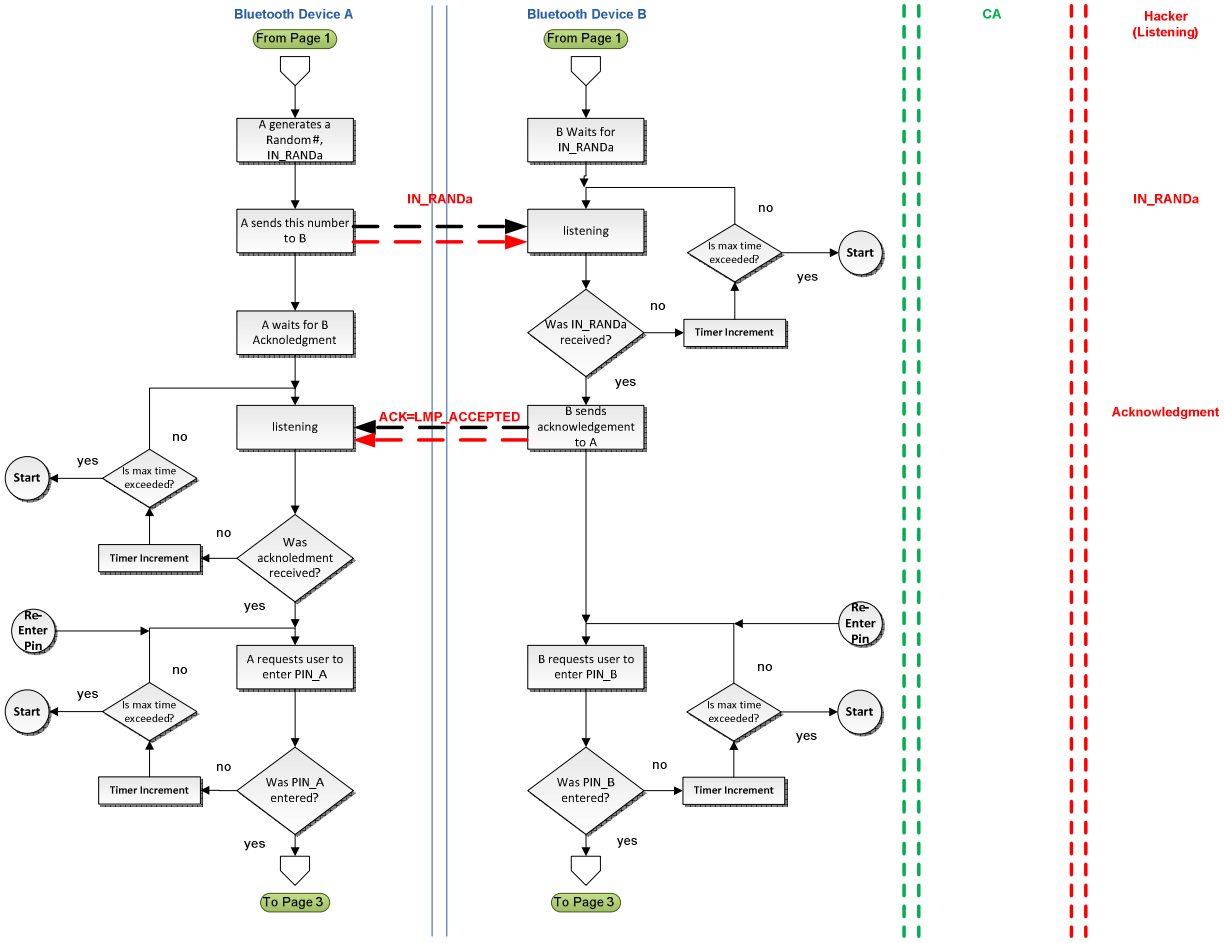

Bluetooth Device A
Bluetooth Device B = Hacker
From Page 1
From Page 1
Mutual Authentication (Challenge Response Process)
Mutual Authentication (Challenge Response Process)
A authenticates B
A generates AU_RANDa
A sends AU_RANDa to B
AU_RANDa
Waiting for AU_RANDa from A - Listening
no
Is max time exceeded?
Start
yes
Was AU_RANDa Received?
no
Timer Increment
yes
SRES'=E1{Kab,AU_RANDa ,BD_ADDRb}
A calculates SRES'
B calculates SRES
SRES=E1{Kab,AU_RANDa, BD_ADDRb}
SRES
A waits for SRES from B (listening)
B sends SRES to B
A compares SRES' to SRES
To Page 3
To Page 3

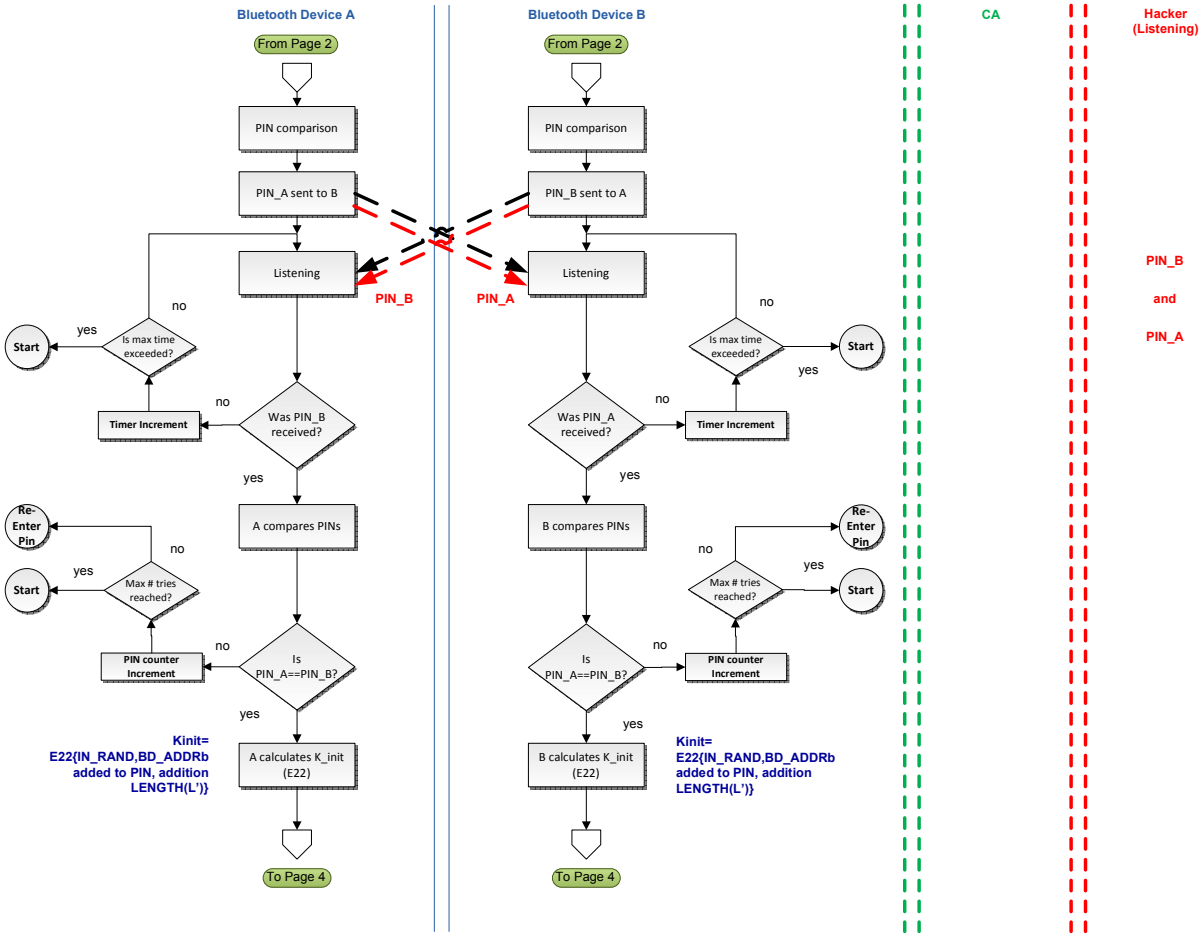

Bluetooth Device A
Bluetooth Device B = Hacker
From Page 2
From Page 2
no
Was SRES'==SRES?
Start
yes
A sends Acknowledgment to B
ACK=LMP_ACCEPTED
B waits for Acknowledgment from A (listening)
no
Is max time exceeded?
Start
yes
Was acknowledgement received?
no
Timer Increment
yes
B authenticates A
B generates AU_RANDb
B sends SU_RANDb to A
AU_RANDb
B waits for AU_RANDb (Listening)
no
Is max time exceeded?
yes
Start
no
Was AU_RANDb Received?
Timer Increment
yes
To Page 4
To Page 4

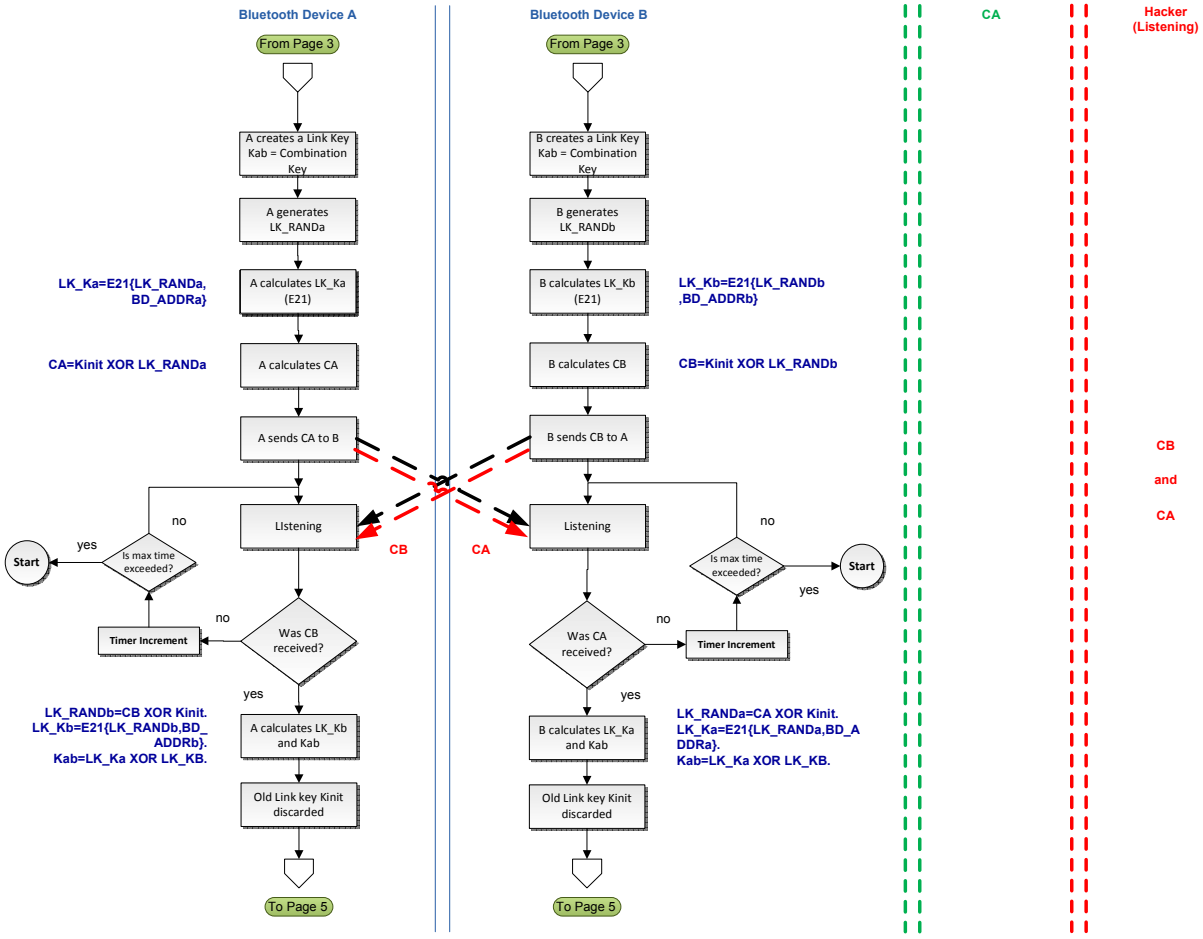

Bluetooth Device A
Bluetooth Device B = Hacker
From Page 3
From Page 3
SRES=E1{Kab,AU_RANDb, BD_ADDRa}
A calculates SRES
B Calculates SRES'
SRES'=E1{Kab,AU_RANDb ,BD_ADDRa}
A sends SRES to B
SRES
B waits for SRES from A (Listening)
no
Is max time exceeded?
Start
yes
Was SRES received?
no
Timer Increment
yes
B compares SRES to SRES'
Was SRES'==SRES?
no
Start
yes
B waits for Acknowledgment from A (listening)
ACK=LMP_ACCEPTED
B sends Acknowledgment to A
no
yes
Start
Is max time exceeded?
no
Timer Increment
Was acknowledgement received?
yes
Start Encryption Process

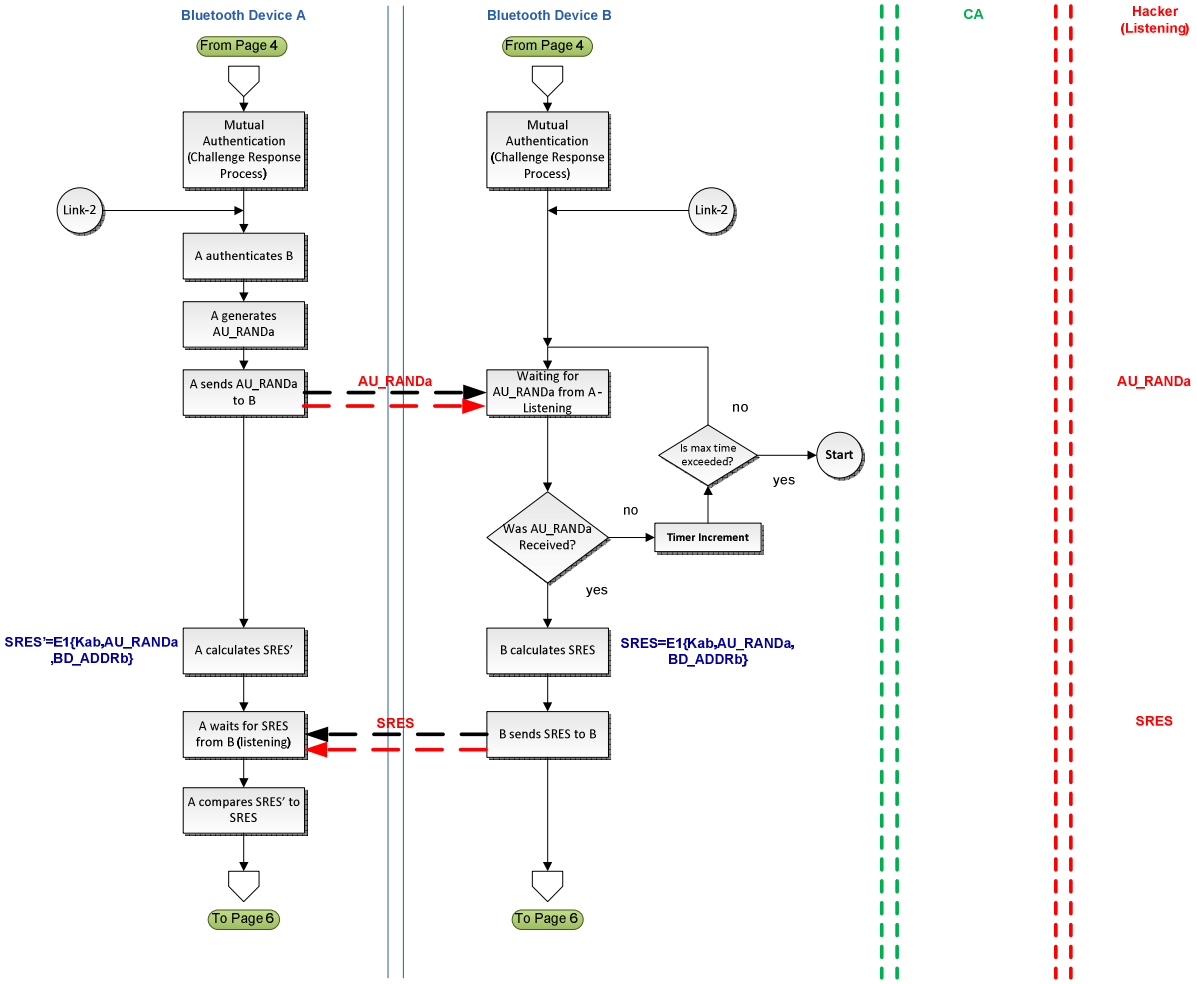
ENHANCED-METHOD FLOW DIAGRAM

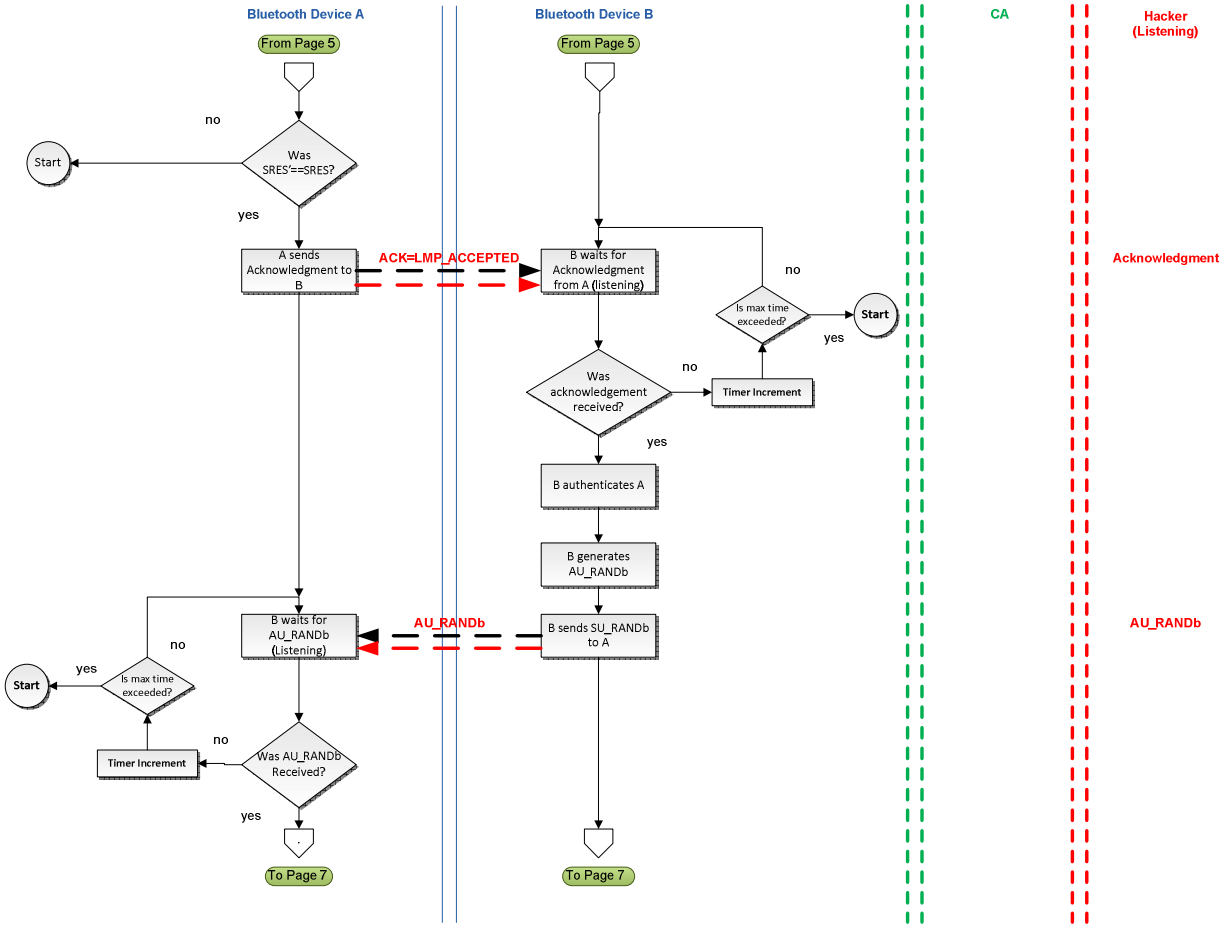


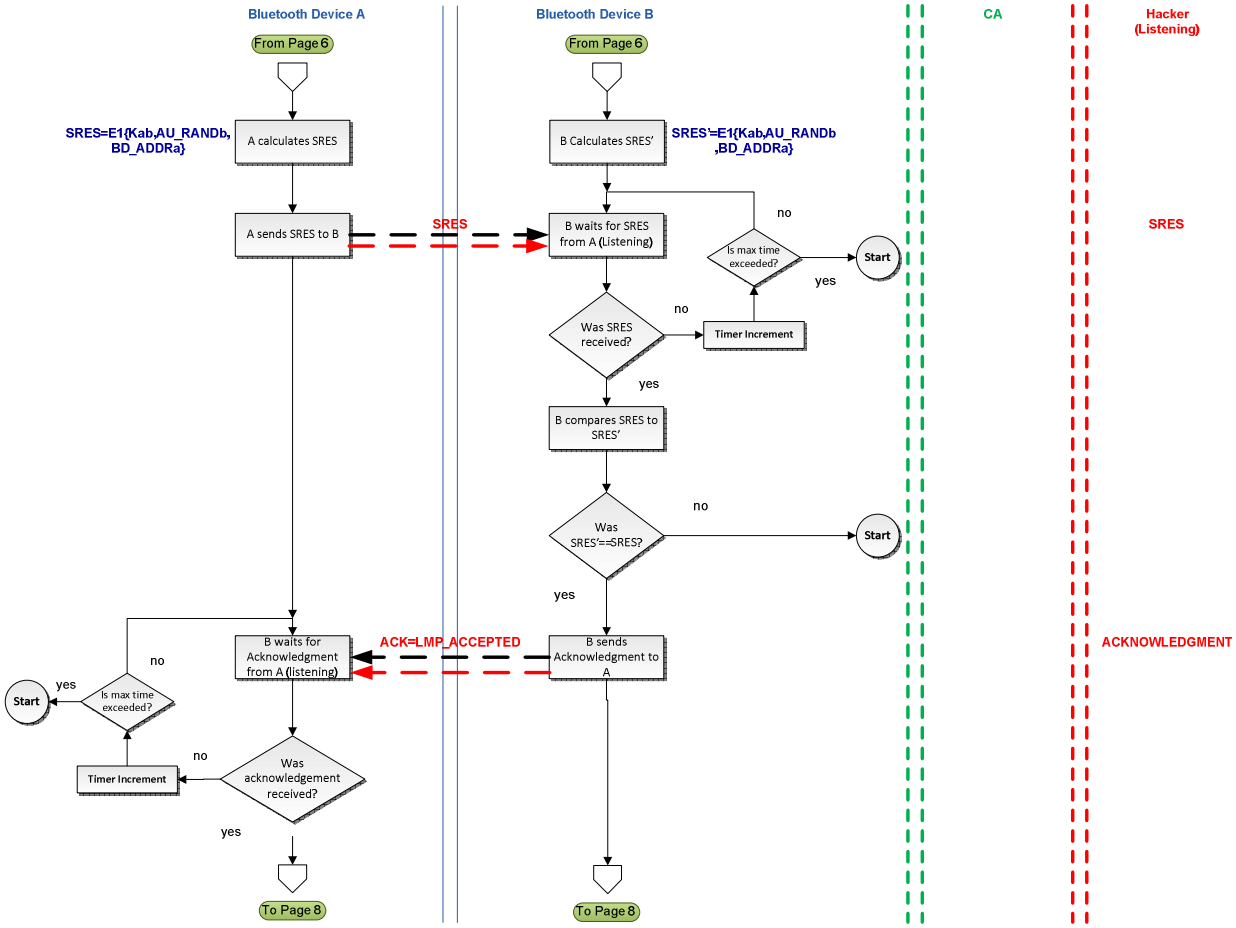

Bluetooth Device A
Bluetooth Device B
CA
Hacker (Listening)
From Page 1
From Page 1
A generates a Random#, IN_RANDa
B Waits for IN_RANDa
IN_RANDa
IN_RANDa
A sends this number to B
listening
no
Is max time exceeded?
Start
yes
A waits for B Acknoledgment
Was IN_RANDa received?
no
Timer Increment
yes
ACK=LMP_ACCEPTED
Acknowledgment
listening
B sends acknowledgement to A
no
Start
yes
Is max time exceeded?
Timer Increment
no
Was acknoledment received?
yes
Re-Enter Pin
A requests user to enter PIN_A
B requests user to enter PIN_B
Re-Enter Pin
no
Start
yes
Is max time exceeded?
Timer Increment
no
Was PIN_A entered?
Was PIN_B entered?
Timer Increment
yes
To Page 3
To Page 3

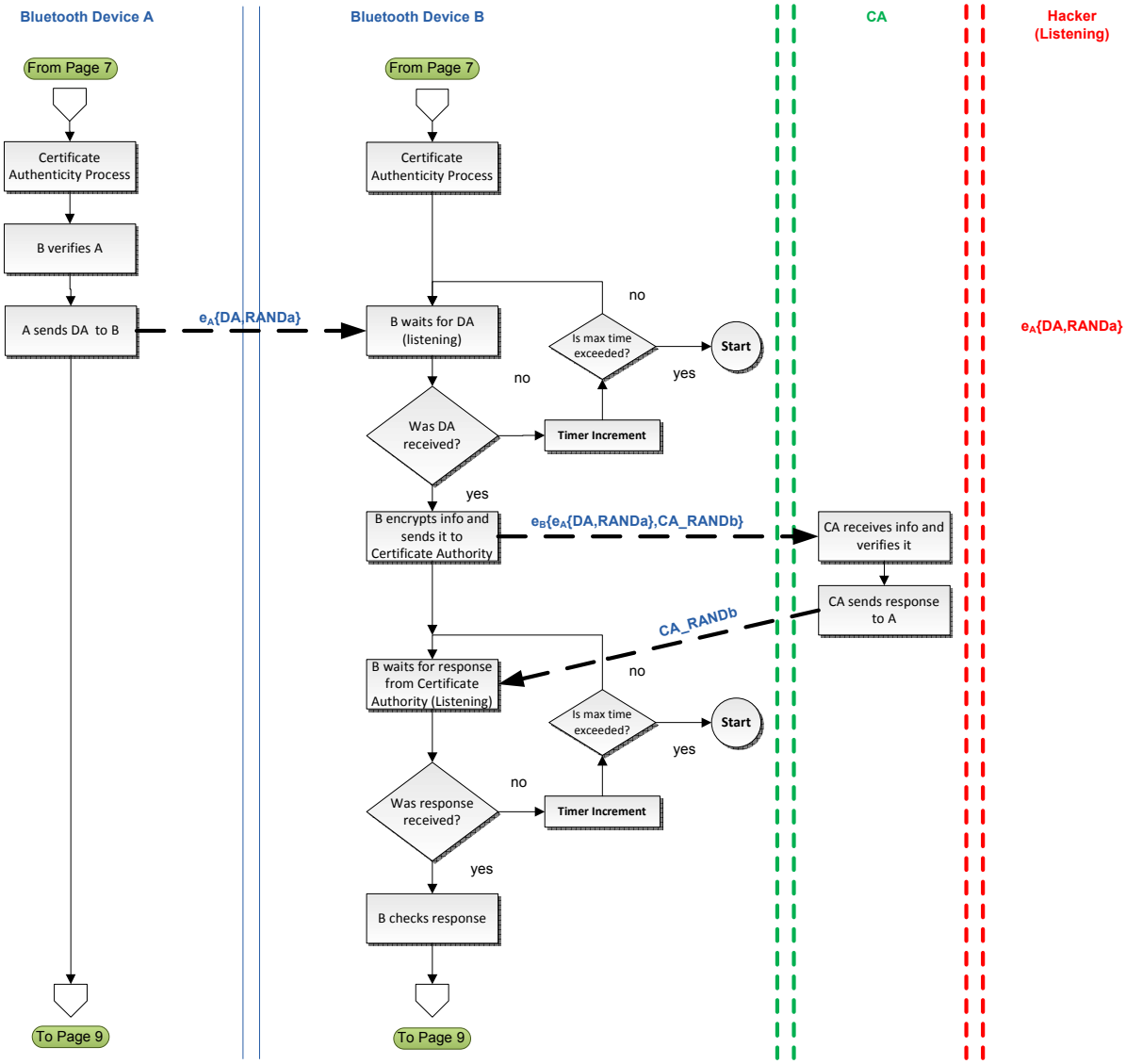

Bluetooth Device A
Bluetooth Device B
CA
Hacker (Listening)
From Page 2
From Page 2
PIN comparison
PIN comparison
PIN_A sent to B
PIN_B sent to A
Listening
Listening
PIN_B
PIN_A
PIN_B and PIN_A
no
yes
Is max time exceeded?
Start
Timer Increment
Was PIN_B received?
Was PIN_A received?
A compares PINs
B compares PINs
Re-Enter Pin
Max # tries reached?
PIN counter Increment
Is PIN_A==PIN_B?
A calculates K_init (E22)
B calculates K_init (E22)
Kinit= E22{IN_RAND,BD_ADDRb added to PIN, addition LENGTH(L')}
To Page 4
To Page 4

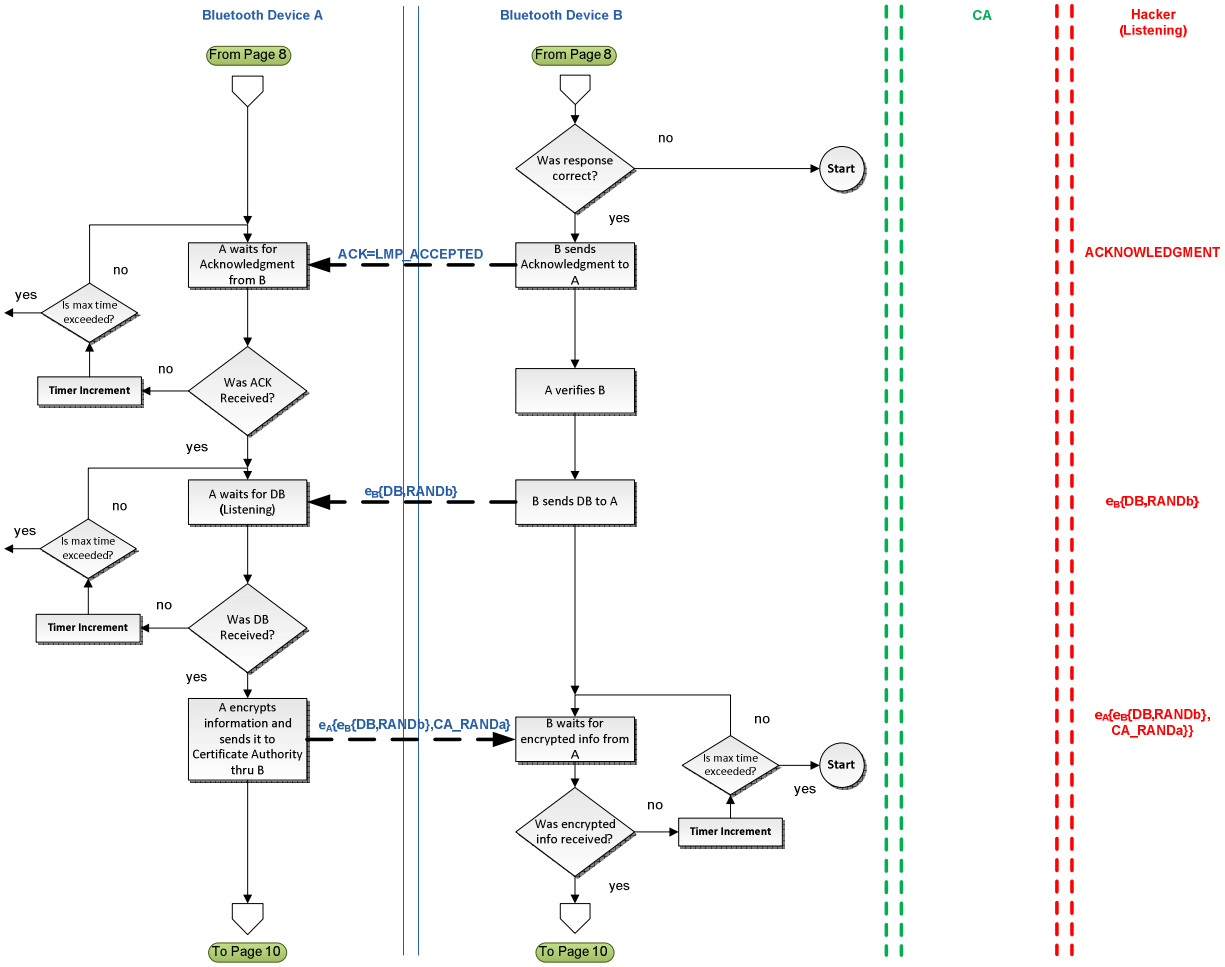

Bluetooth Device A
Bluetooth Device B
CA
Hacker (Listening)
From Page 3
From Page 3
A creates a Link Key Kab = Combination Key
B creates a Link Key Kab = Combination Key
A generates LK_RANDa
B generates LK_RANDb
LK_Ka=E21{LK_RANDa, BD_ADDRa}
A calculates LK_Ka (E21)
B calculates LK_Kb (E21)
LK_Kb=E21{LK_RANDb ,BD_ADDRb}
CA=Kinit XOR LK_RANDa
A calculates CA
B calculates CB
CB=Kinit XOR LK_RANDb
A sends CA to B
B sends CB to A
CB
and
CA
LIstening
Listening
no
yes
Start
Is max time exceeded?
Timer Increment
no
Was CB received?
Was CA received?
yes
LK_RANDb=CB XOR Kinit. LK_Kb=E21{LK_RANDb,BD_ADDRb}. Kab=LK_Ka XOR LK_KB.
A calculates LK_Kb and Kab
B calculates LK_Ka and Kab
LK_RANDa=CA XOR Kinit. LK_Ka=E21{LK_RANDa,BD_ADDRa}. Kab=LK_Ka XOR LK_KB.
Old Link key Kinit discarded
To Page 5
To Page 5

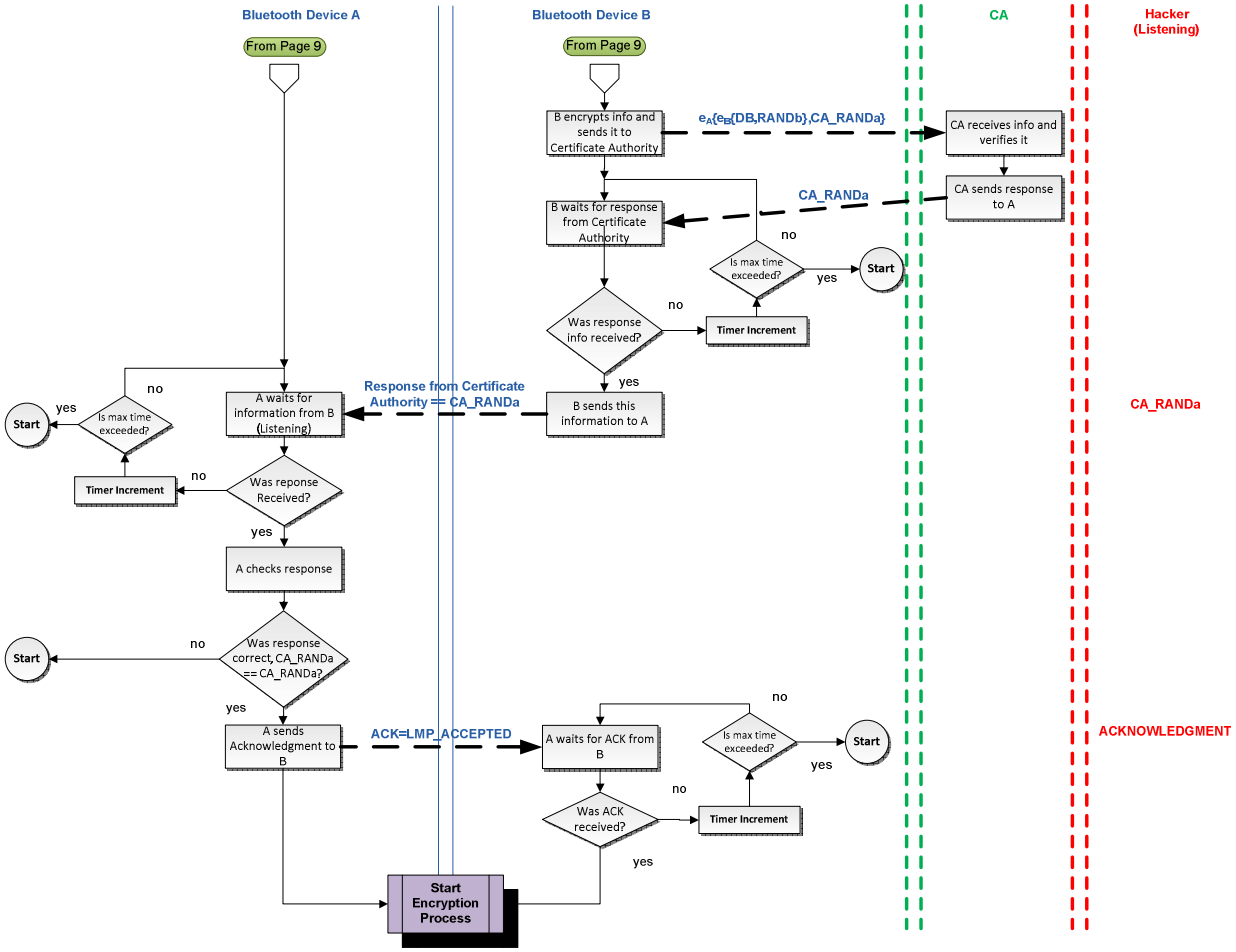

Bluetooth Device A
Bluetooth Device B
CA
Hacker (Listening)
From Page 4
From Page 4
Mutual Authentication (Challenge Response Process)
Mutual Authentication (Challenge Response Process)
Link-2
Link-2
A authenticates B
A generates AU_RANDa
A sends AU_RANDa to B
AU_RANDa
Waiting for AU_RANDa from A - Listening
AU_RANDa
no
Is max time exceeded?
Start
yes
no
Was AU_RANDa Received?
Timer Increment
yes
SRES'=E1{Kab,AU_RANDa ,BD_ADDRb}
A calculates SRES'
B calculates SRES
SRES=E1{Kab,AU_RANDa, BD_ADDRb}
A waits for SRES from B (listening)
SRES
B sends SRES to B
SRES
A compares SRES' to SRES
To Page 6
To Page 6


Bluetooth Device A
Bluetooth Device B
CA
Hacker (Listening)
From Page 5
From Page 5
no
Start
Was SRES'==SRES?
yes
A sends Acknowledgment to B
ACK=LMP_ACCEPTED
B waits for Acknowledgment from A (listening)
Acknowledgment
no
Is max time exceeded?
Start
yes
Was acknowledgement received?
no
Timer Increment
yes
B authenticates A
B generates AU_RANDb
B waits for AU_RANDb (Listening)
AU_RANDb
B sends SU_RANDb to A
AU_RANDb
no
yes
Start
Is max time exceeded?
Timer Increment
no
Was AU_RANDb Received?
yes
To Page 7
To Page 7

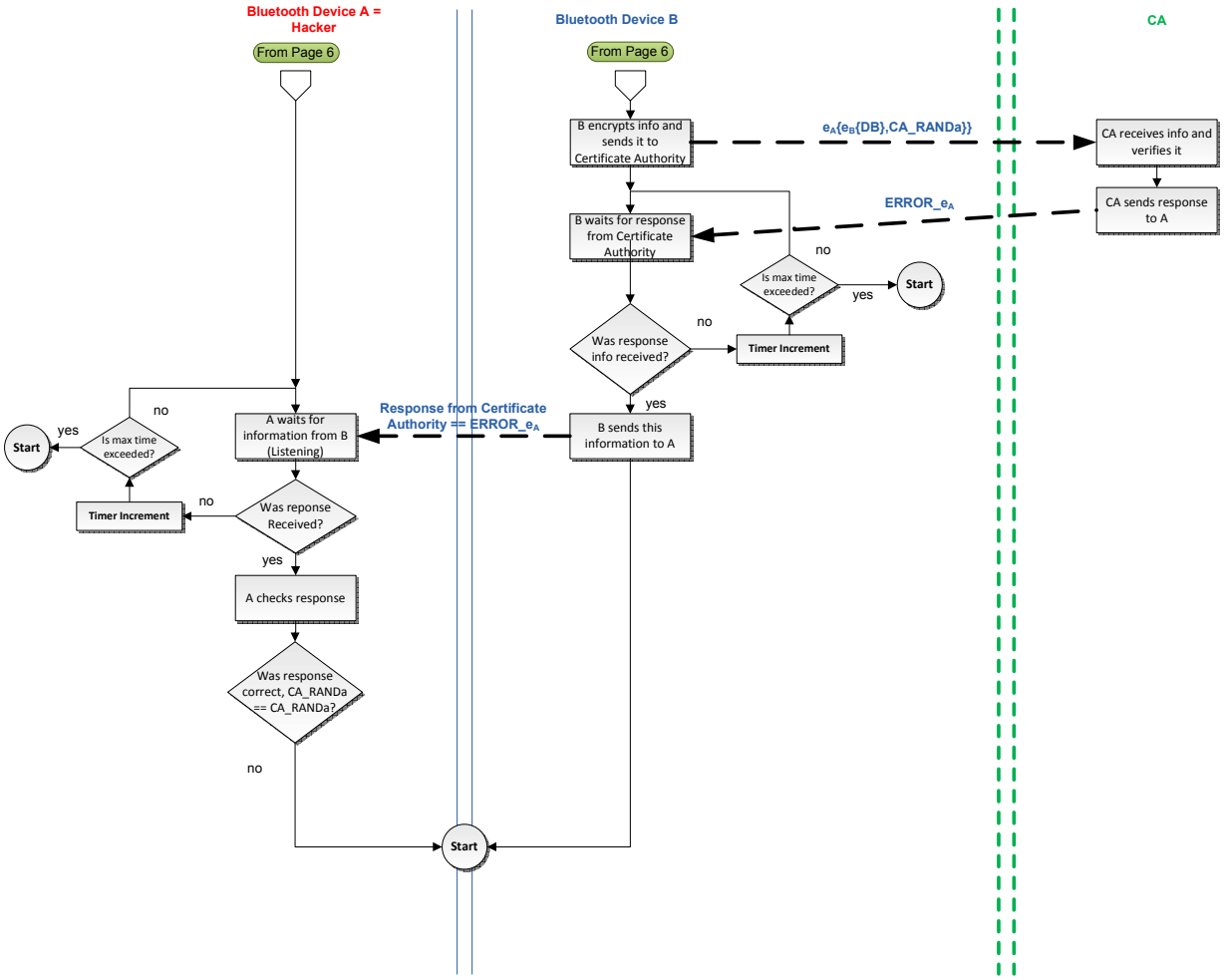

Bluetooth Device A
Bluetooth Device B
CA
Hacker (Listening)
From Page 6
From Page 6
SRES=E1(Kab,AU_RANDb, BD_ADDRa)
A calculates SRES
B Calculates SRES'
SRES'=E1(Kab,AU_RANDb ,BD_ADDRa)
A sends SRES to B
SRES
B waits for SRES from A (Listening)
SRES
no
Is max time exceeded?
Start
yes
Was SRES received?
no
Timer Increment
yes
B compares SRES to SRES'
Was SRES'==SRES?
no
Start
yes
B waits for Acknowledgment from A (listening)
ACK=LMP_ACCEPTED
B sends Acknowledgment to A
ACKNOWLEDGMENT
no
Start
yes
Is max time exceeded?
Timer Increment
no
Was acknowledgement received?
yes
To Page 8
To Page 8

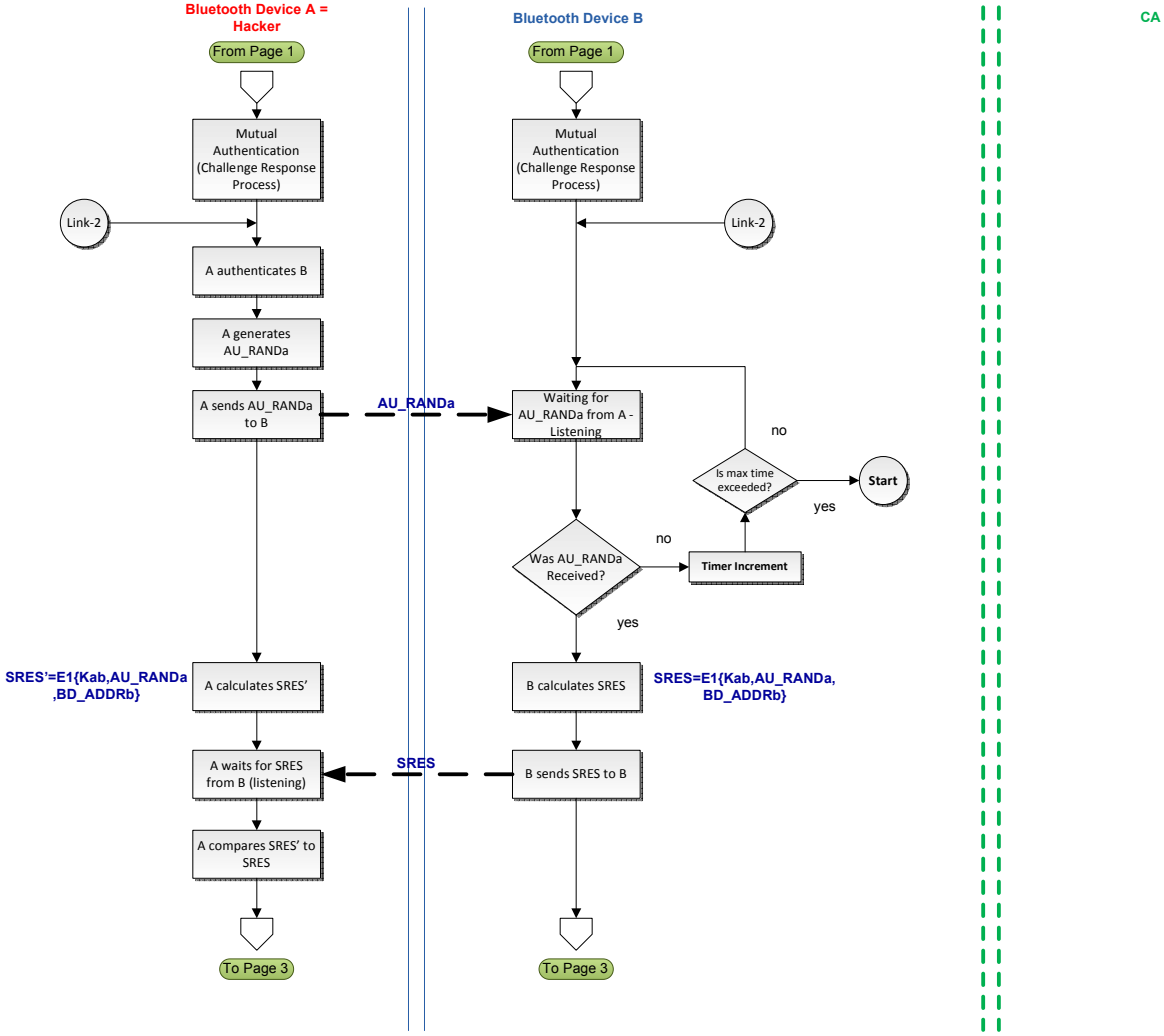

Bluetooth Device A
Bluetooth Device B
CA
Hacker (Listening)
From Page 7
From Page 7
Certificate Authenticity Process
Certificate Authenticity Process
B verifies A
A sends DA to B
eA{DA,RANDa}
B waits for DA (listening)
no
Is max time exceeded?
Start
yes
no
Was DA received?
Timer Increment
yes
eA{DA,RANDa}
B encrypts info and sends it to Certificate Authority
eB{eA{DA,RANDa},CA_RANDb}
CA receives info and verifies it
CA sends response to A
CA_RANDb
B waits for response from Certificate Authority (Listening)
no
Is max time exceeded?
Start
yes
no
Was response received?
Timer Increment
yes
B checks response
To Page 9
To Page 9

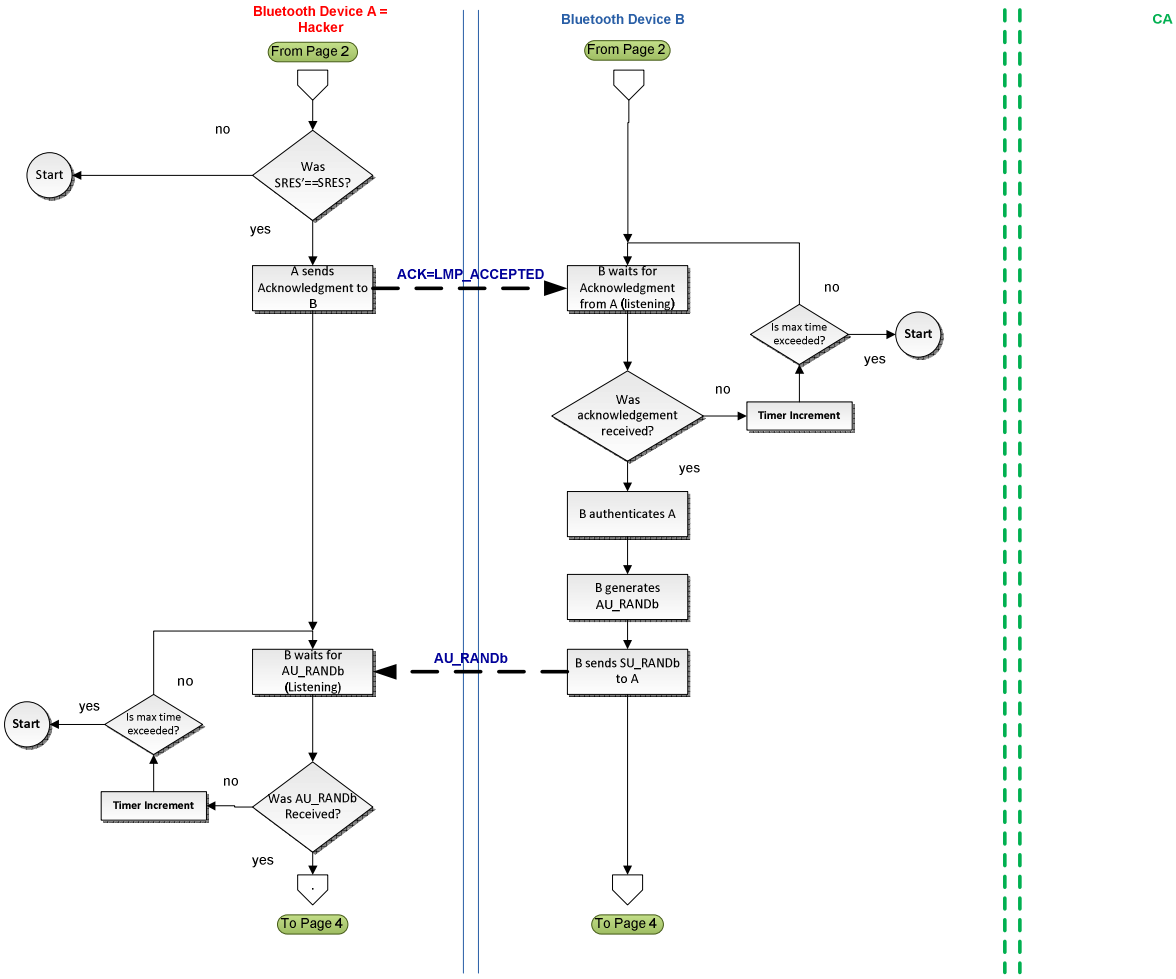

Bluetooth Device A
Bluetooth Device B
CA
Hacker (Listening)
From Page 8
From Page 8
Was response correct?
no
Start
yes
A waits for Acknowledgment from B
ACK=LMP_ACCEPTED
B sends Acknowledgment to A
ACKNOWLEDGMENT
no
yes
Is max time exceeded?
Timer Increment
no
Was ACK Received?
A verifies B
yes
A waits for DB (Listening)
eB{DB,RANDb}
B sends DB to A
eB{DB,RANDb}
no
yes
Is max time exceeded?
Timer Increment
no
Was DB Received?
yes
A encrypts information and sends it to Certificate Authority thru B
eA{eB{DB,RANDb},CA_RANDa}
B waits for encrypted info from A
eA{eB{DB,RANDb}, CA_RANDa}}
no
Is max time exceeded?
Start
yes
no
Was encrypted info received?
Timer Increment
yes
To Page 10
To Page 10

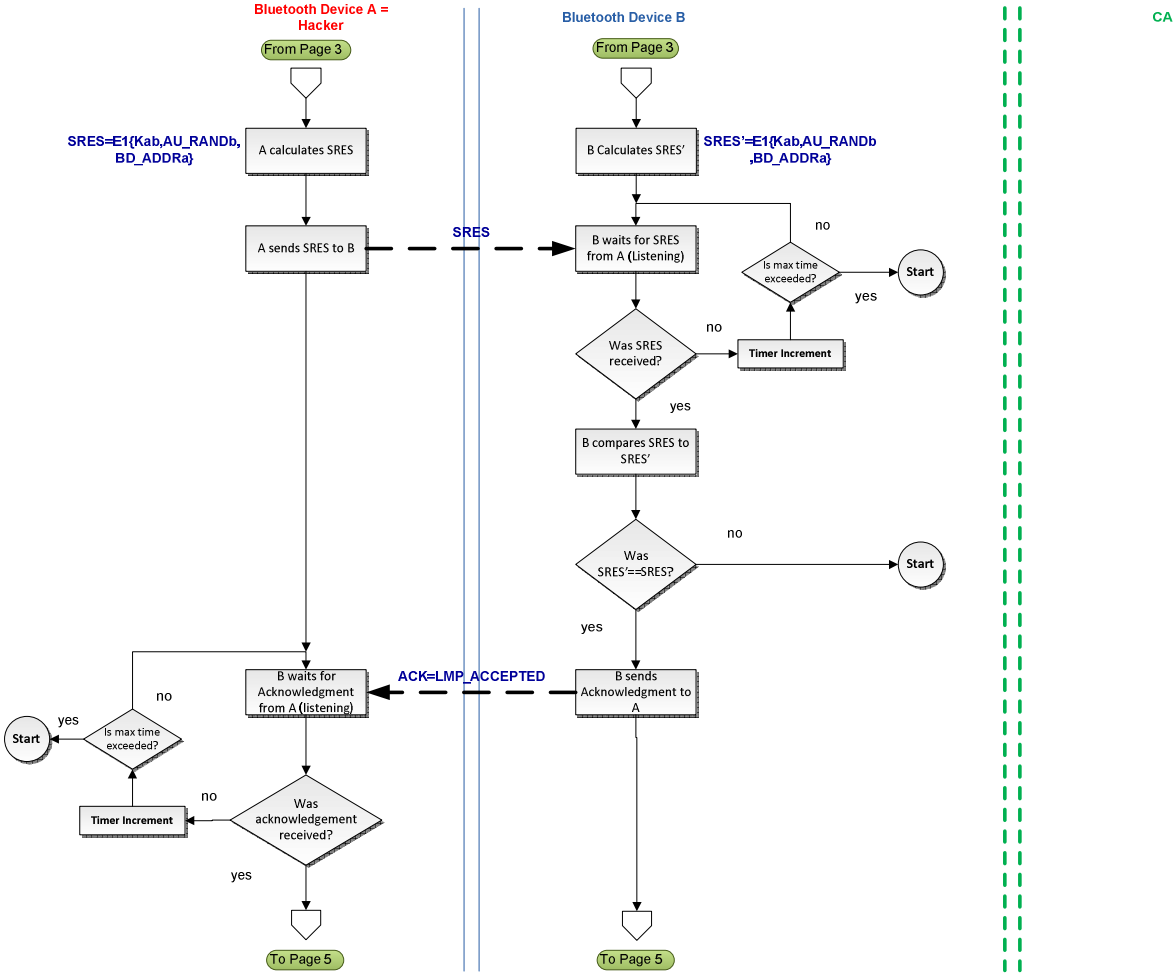

Bluetooth Device A
Bluetooth Device B
CA
Hacker (Listening)
From Page 9
From Page 9
B encrypts info and sends it to Certificate Authority
$e_A\{e_{BB}\{DB,RANDb\},CA\_RANDa\}$
CA receives info and verifies it
CA sends response to A
CA_RANDa
B waits for response from Certificate Authority
no
Is max time exceeded?
yes
Start
Was response info received?
no
Timer Increment
yes
Response from Certificate Authority == CA_RANDa
B sends this information to A
CA_RANDa
A waits for information from B (Listening)
no
Is max time exceeded?
yes
Start
Timer Increment
no
Was reponse Received?
yes
A checks response
Was response correct, CA_RANDa == CA_RANDa?
no
Start
yes
A sends Acknowledgment to B
ACK=LMP_ACCEPTED
A waits for ACK from B
no
Is max time exceeded?
yes
Start
Was ACK received?
no
Timer Increment
yes
ACKNOWLEDGMENT
Start Encryption Process

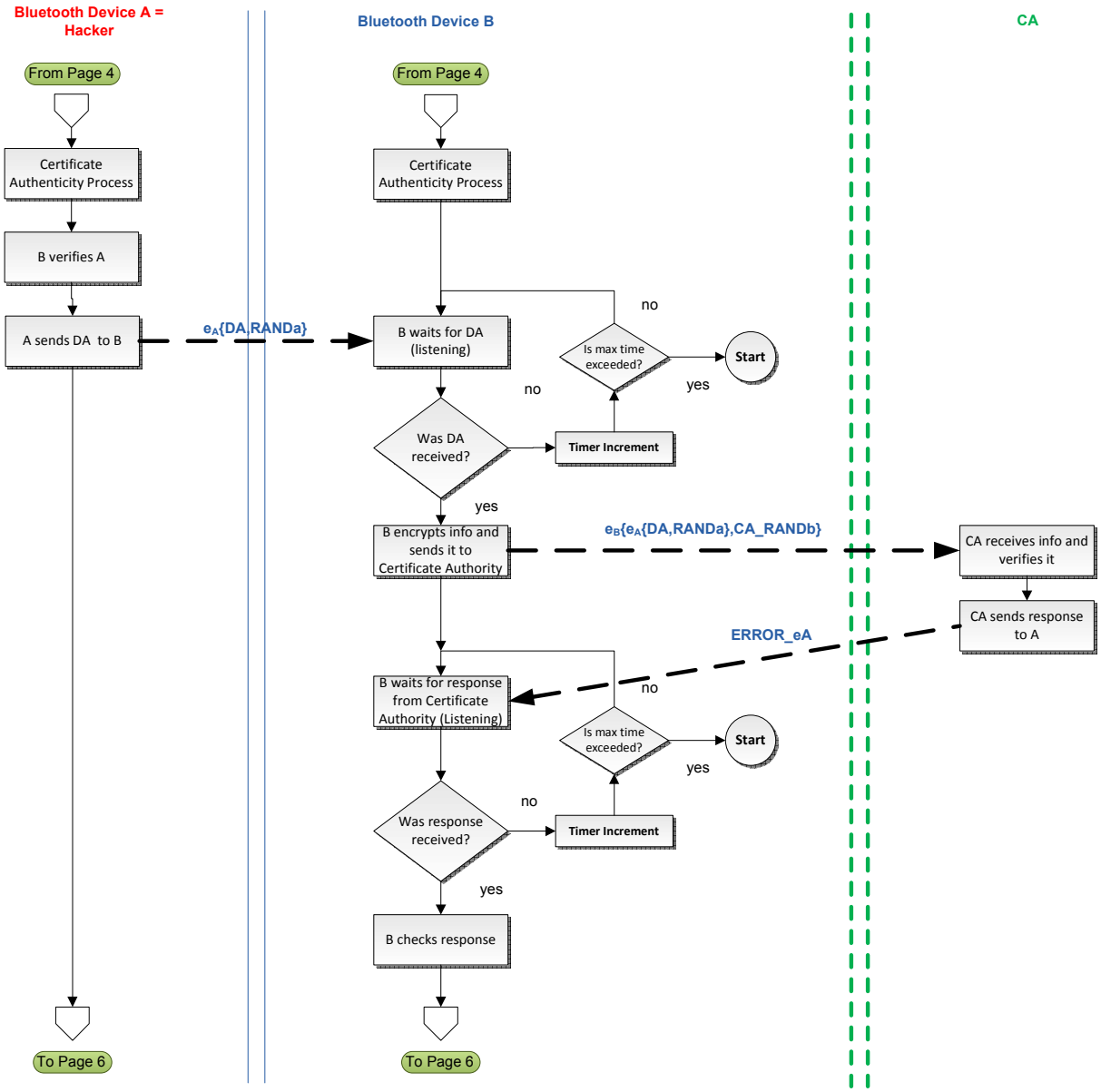
ENHANCED-HACKED-METHOD-WHEN-A-IS-A-HACKER FLOW DIAGRAM

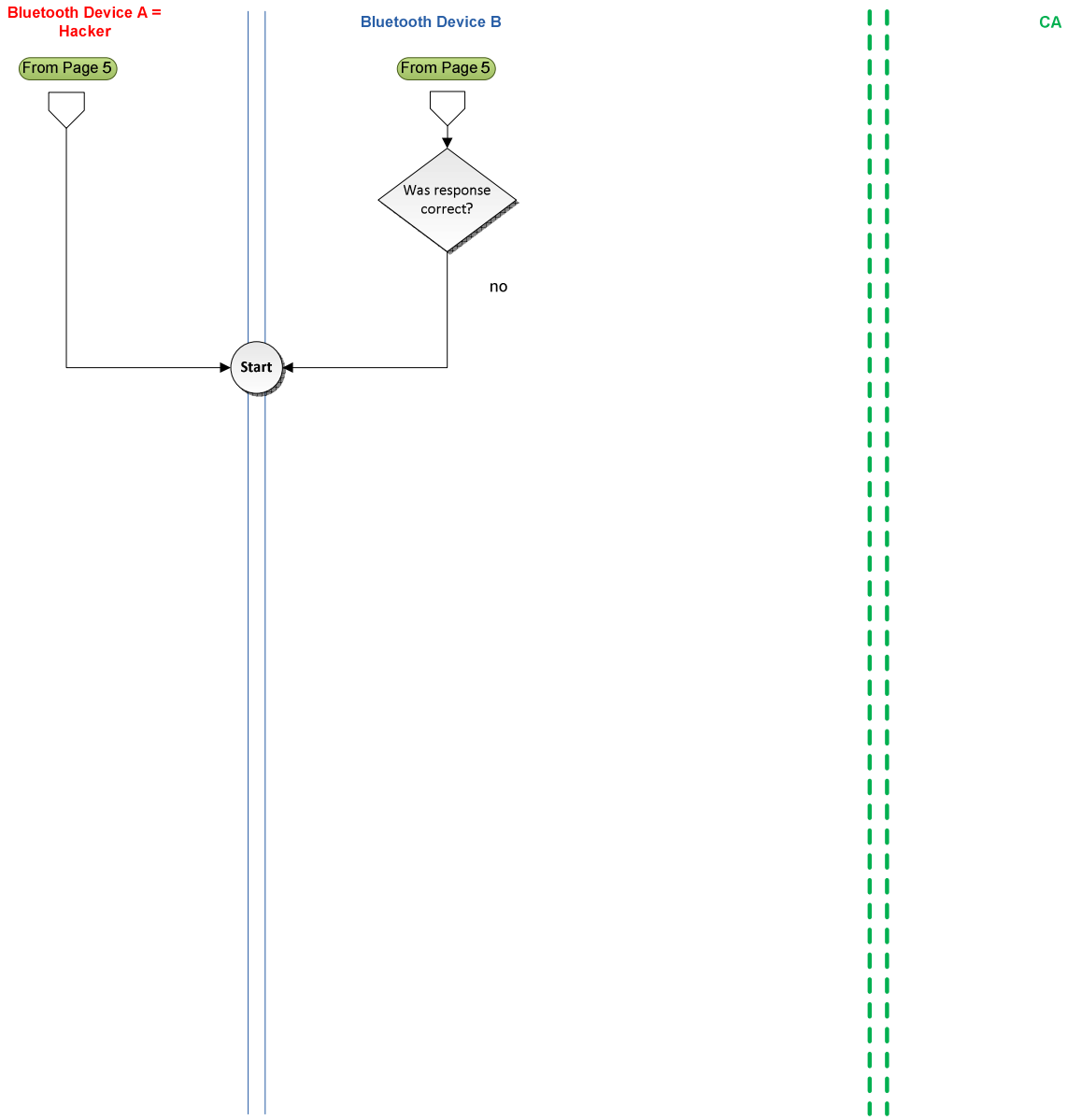



Bluetooth Device A = Hacker
Bluetooth Device B
CA
From Page 1
From Page 1
Mutual Authentication (Challenge Response Process)
Mutual Authentication (Challenge Response Process)
Link-2
Link-2
A authenticates B
A generates AU_RANDa
A sends AU_RANDa to B
AU_RANDa
Waiting for AU_RANDa from A - Listening
no
Is max time exceeded?
Start
yes
Was AU_RANDa Received?
no
Timer Increment
yes
SRES'=E1{Kab,AU_RANDa ,BD_ADDRb}
A calculates SRES'
B calculates SRES
SRES=E1{Kab,AU_RANDa, BD_ADDRb}
A waits for SRES from B (listening)
SRES
B sends SRES to B
A compares SRES' to SRES
To Page 3
To Page 3

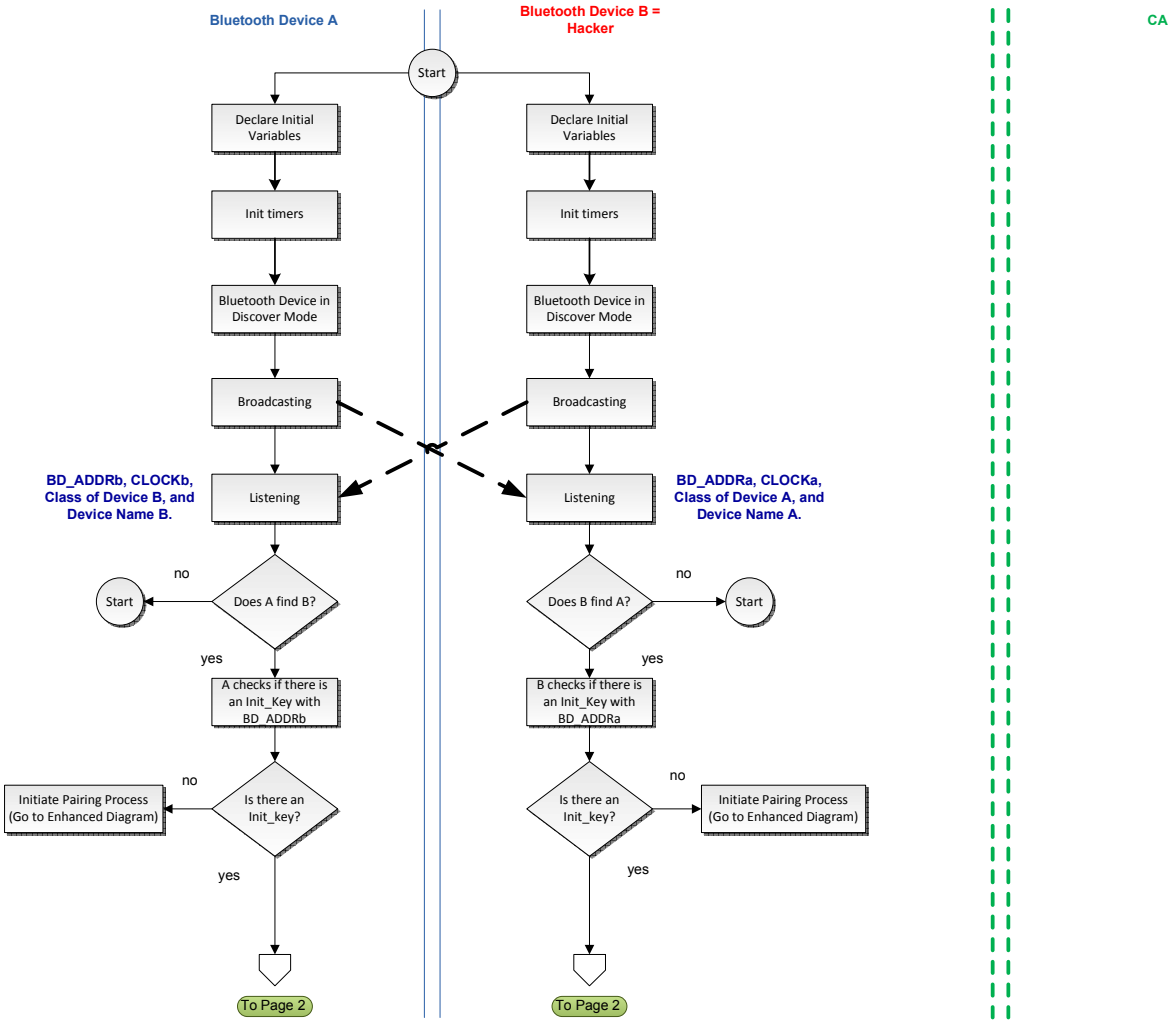

Bluetooth Device A = Hacker
Bluetooth Device B
CA
From Page 2
From Page 2
no
Start
Was SRES'==SRES?
yes
A sends Acknowledgment to B
ACK=LMP_ACCEPTED
B waits for Acknowledgment from A (listening)
no
Is max time exceeded?
Start
yes
Was acknowledgement received?
no
Timer Increment
yes
B authenticates A
B generates AU_RANDb
B waits for AU_RANDb (Listening)
AU_RANDb
B sends SU_RANDb to A
no
yes
Start
Is max time exceeded?
no
Timer Increment
Was AU_RANDb Received?
yes
To Page 4
To Page 4

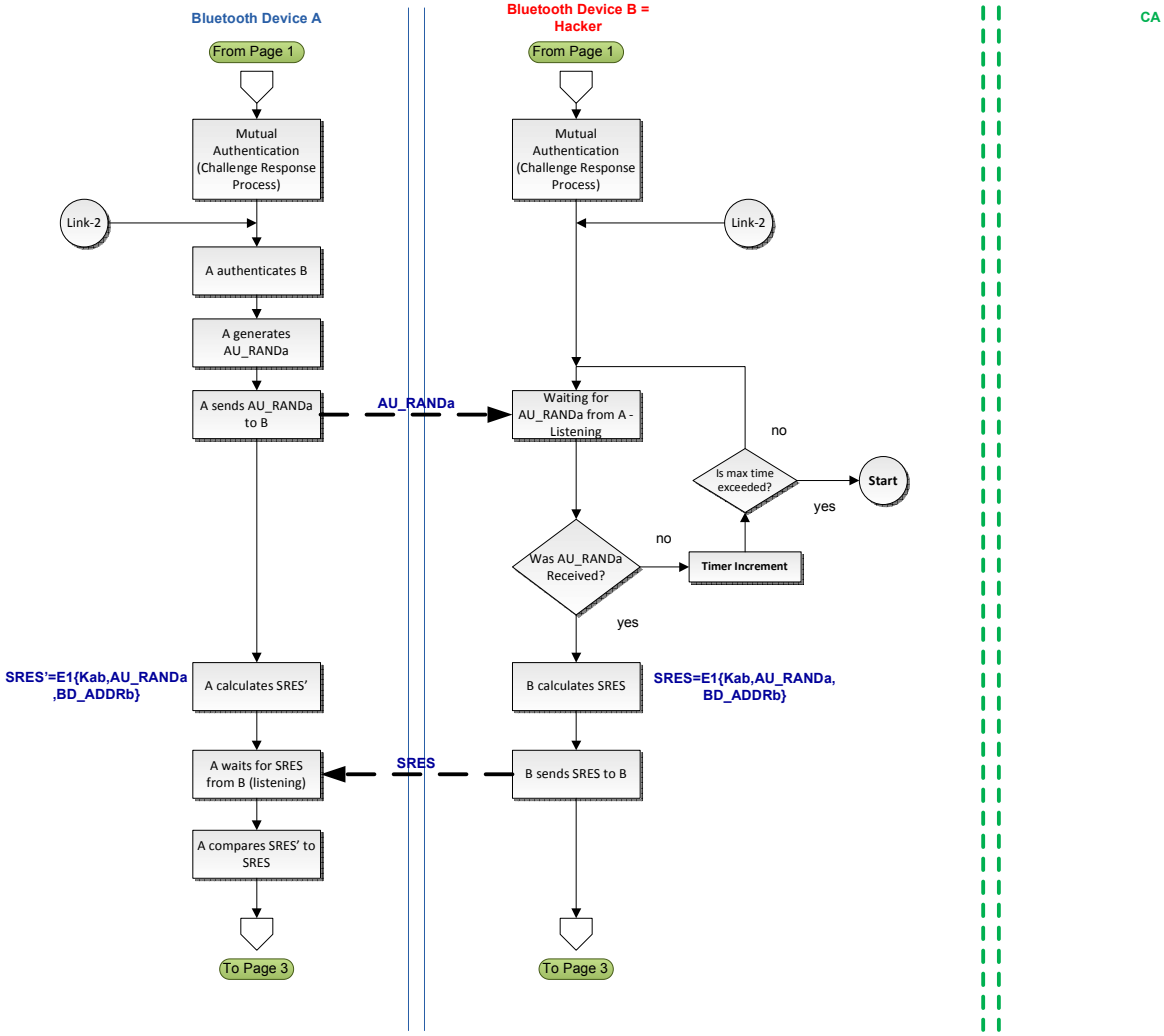

Bluetooth Device A = Hacker
Bluetooth Device B
CA
From Page 3
From Page 3
SRES=E1{Kab,AU_RANDb, BD_ADDRa}
A calculates SRES
B Calculates SRES'
SRES'=E1{Kab,AU_RANDb ,BD_ADDRa}
A sends SRES to B
SRES
B waits for SRES from A (Listening)
no
Is max time exceeded?
Start
yes
Was SRES received?
no
Timer Increment
yes
B compares SRES to SRES'
Was SRES'==SRES?
no
Start
yes
B waits for Acknowledgment from A (listening)
ACK=LMP_ACCEPTED
B sends Acknowledgment to A
no
yes
Start
Is max time exceeded?
Timer Increment
no
Was acknowledgement received?
yes
To Page 5
To Page 5

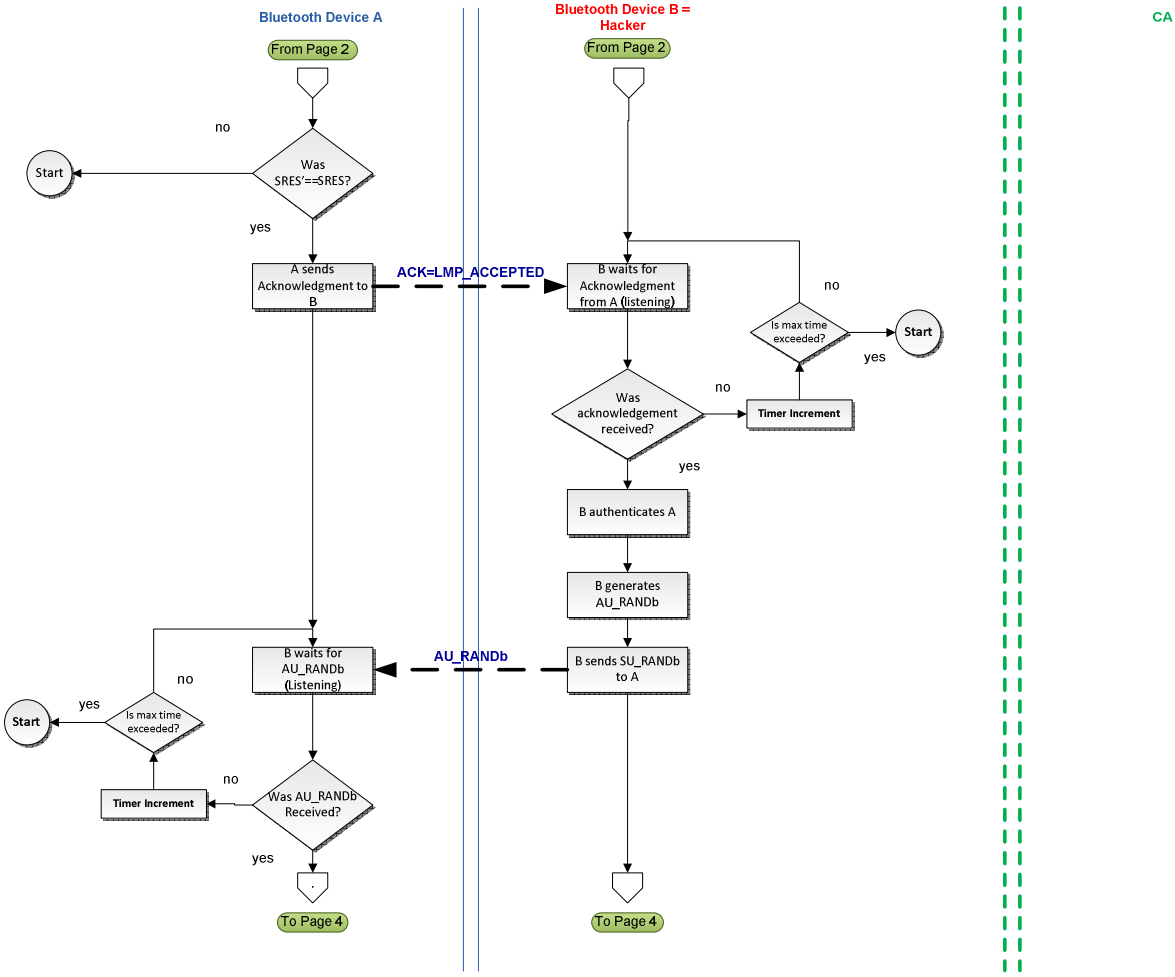

Bluetooth Device A = Hacker
Bluetooth Device B
CA
From Page 4
From Page 4
Certificate Authenticity Process
Certificate Authenticity Process
B verifies A
A sends DA to B
e_A{DA,RANDa}
B waits for DA (listening)
no
Is max time exceeded?
Start
yes
no
Was DA received?
Timer Increment
yes
B encrypts info and sends it to Certificate Authority
$e_B{e_A{DA,RANDa},CA_RANDb}$
CA receives info and verifies it
CA sends response to A
ERROR_eA
B waits for response from Certificate Authority (Listening)
no
Is max time exceeded?
Start
yes
no
Was response received?
Timer Increment
yes
B checks response
To Page 6
To Page 6

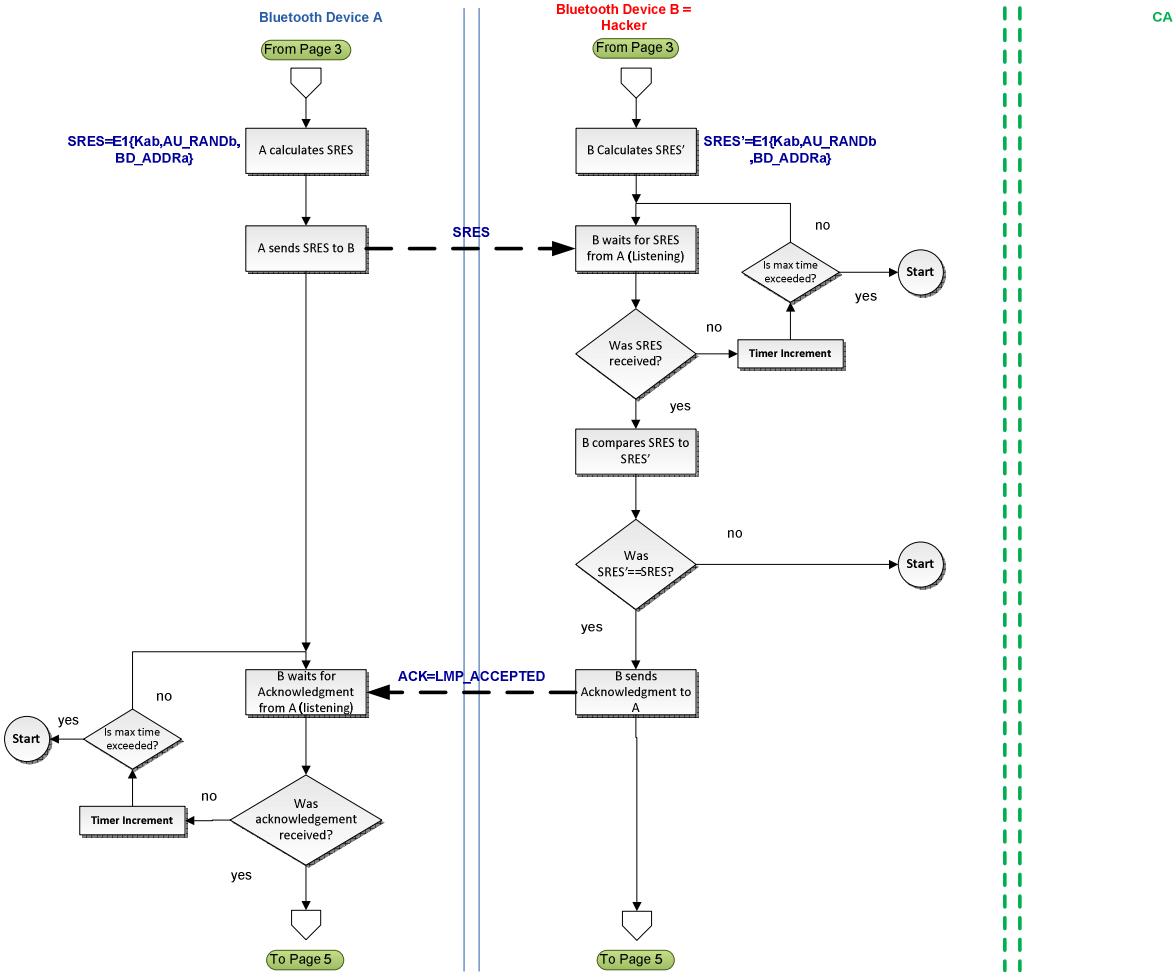

Bluetooth Device A = Hacker
Bluetooth Device B
CA
From Page 5
From Page 5
Was response correct?
no
Start

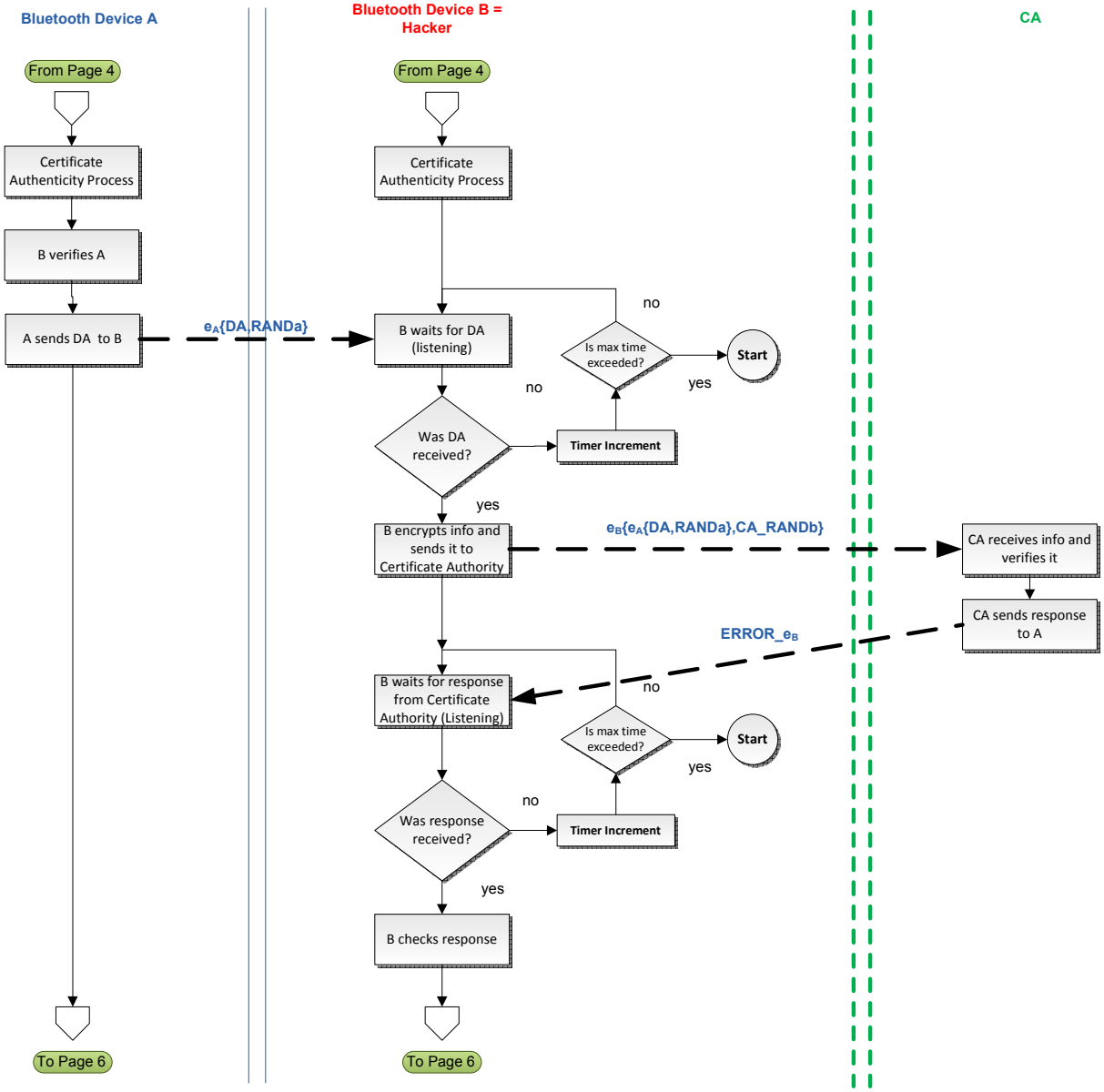
ENHANCED-HACKED-METHOD-WHEN-B-IS-A-HACKER FLOW DIAGRAM

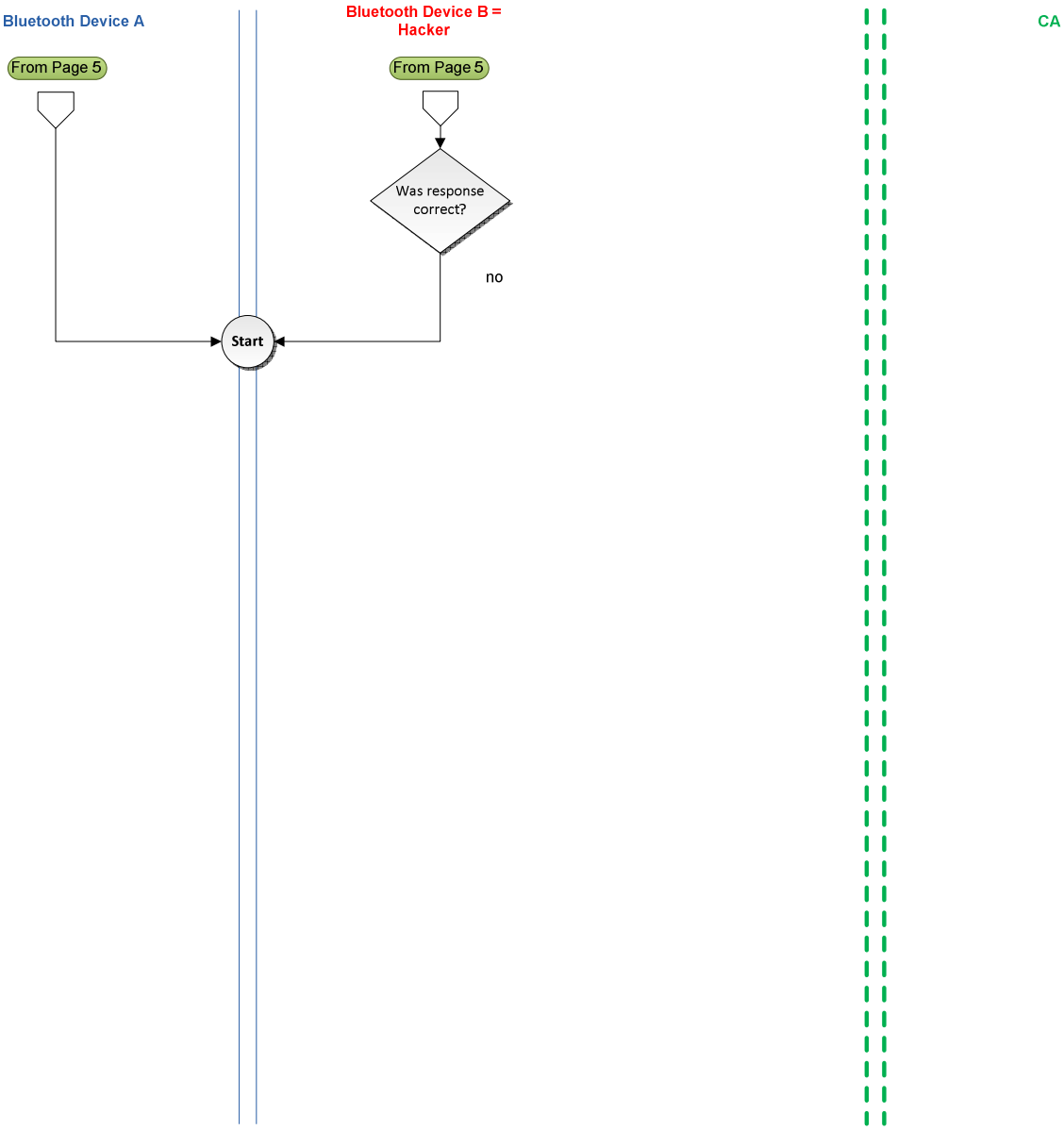



Bluetooth Device A
Bluetooth Device B = Hacker
CA
From Page 1
From Page 1
Mutual Authentication (Challenge Response Process)
Mutual Authentication (Challenge Response Process)
Link-2
Link-2
A authenticates B
A generates AU_RANDa
A sends AU_RANDa to B
AU_RANDa
Waiting for AU_RANDa from A - Listening
no
Is max time exceeded?
Start
yes
Was AU_RANDa Received?
no
Timer Increment
yes
SRES'=E1{Kab,AU_RANDa ,BD_ADDRb}
A calculates SRES'
B calculates SRES
SRES=E1{Kab,AU_RANDa, BD_ADDRb}
A waits for SRES from B (listening)
SRES
B sends SRES to B
A compares SRES' to SRES
To Page 3
To Page 3


Bluetooth Device A
Bluetooth Device B = Hacker
CA
From Page 2
From Page 2
no
Was SRES'==SRES?
Start
yes
A sends Acknowledgment to B
ACK=LMP_ACCEPTED
B waits for Acknowledgment from A (listening)
no
Is max time exceeded?
Start
yes
Was acknowledgement received?
no
Timer Increment
yes
B authenticates A
B generates AU_RANDb
B waits for AU_RANDb (Listening)
AU_RANDb
B sends SU_RANDb to A
no
Is max time exceeded?
Start
yes
no
Timer Increment
Was AU_RANDb Received?
yes
To Page 4
To Page 4


Bluetooth Device A
Bluetooth Device B = Hacker
CA
From Page 3
From Page 3
SRES=E1{Kab,AU_RANDb, BD_ADDRa}
A calculates SRES
B Calculates SRES'
SRES'=E1{Kab,AU_RANDb ,BD_ADDRa}
A sends SRES to B
SRES
B waits for SRES from A (Listening)
no
Is max time exceeded?
Start
yes
Was SRES received?
no
Timer Increment
yes
B compares SRES to SRES'
Was SRES'==SRES?
no
Start
yes
B waits for Acknowledgment from A (listening)
ACK=LMP_ACCEPTED
B sends Acknowledgment to A
no
yes
Start
Is max time exceeded?
Timer Increment
no
Was acknowledgement received?
yes
To Page 5
To Page 5


Bluetooth Device A
Bluetooth Device B = Hacker
CA
From Page 4
From Page 4
Certificate Authenticity Process
Certificate Authenticity Process
B verifies A
A sends DA to B
e_A{DA,RANDa}
B waits for DA (listening)
no
Is max time exceeded?
Start
yes
no
Was DA received?
Timer Increment
yes
B encrypts info and sends it to Certificate Authority
$e_B{e_A{DA,RANDa},CA_RANDb}$
CA receives info and verifies it
CA sends response to A
ERROR_e_B
B waits for response from Certificate Authority (Listening)
no
Is max time exceeded?
Start
yes
no
Was response received?
Timer Increment
yes
B checks response
To Page 6
To Page 6


Bluetooth Device A
Bluetooth Device B = Hacker
CA
From Page 5
From Page 5
Was response correct?
no
Start

Appendix G: C# Code for Visual Simulations

BLUETOOTHTHESIS.CSPROJ

……

File name: Form1.cs

```
//Project Name: BluetoothThesis.csproj

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace BluetoothThesis
{
	/// <summary>
	/// Summary description for Form1.
	/// </summary>
	public class Form1 : System.Windows.Forms.Form
	{
		private System.Windows.Forms.TabControl tabControl1;
		private System.Windows.Forms.TabPage OriginalTab;
		private System.Windows.Forms.TabPage OriginalHackedTab;
		private System.Windows.Forms.TabPage EnhancedTab;
		private System.Windows.Forms.MainMenu MainMenu;
		private System.Windows.Forms.MenuItem FileMenu;
		private System.Windows.Forms.MenuItem ExitMenu;
		private System.Windows.Forms.PictureBox pictureBox1;
		private System.Windows.Forms.PictureBox pictureBox2;
		private System.Windows.Forms.PictureBox pictureBox3;
		private System.Windows.Forms.PictureBox pictureBox4;
		private System.Windows.Forms.Label OriginalLabelA_B;
		private System.Windows.Forms.Label OriginalLabelB_A;
		private System.Windows.Forms.Label label3;
		private System.Windows.Forms.Label label4;
		private System.Windows.Forms.Button OriginalNextStep;
		private System.Windows.Forms.PictureBox pictureBox5;
		private System.Windows.Forms.Label label1;
		private System.Windows.Forms.TextBox OriginalAData;
		private System.Windows.Forms.TextBox OriginalBData;
		private System.Windows.Forms.TextBox OriginalHData;
		private System.Windows.Forms.Label label6;
		private System.Windows.Forms.Label label7;
		private System.Windows.Forms.Label OriginalStatusA;
		private System.Windows.Forms.Label OriginalStatusB;
		private System.Windows.Forms.Button OriginalReset;


		//_____________________________________________
		private System.Collections.ArrayList OriginalStateA;
```

```
    private System.Collections.ArrayList OriginalStateB;
    private System.Collections.ArrayList HackedStateA;
    private System.Collections.ArrayList HackedStateB;
    private System.Collections.ArrayList EnhancedStateA;
    private System.Collections.ArrayList EnhancedStateB;
    private System.Collections.ArrayList EnhancedStateCA;
    private System.Collections.ArrayList EHStateA;
    private System.Collections.ArrayList EHStateB;
    private System.Collections.ArrayList EHStateCA;
private System.Collections.ArrayList EHBStateA;
private System.Collections.ArrayList EHBStateB;
private System.Collections.ArrayList EHBStateCA;
    private int OriginalState=-1;
    private int HackedState=-1;
    private int EnhancedState=-1;
    private int EHState=-1;
private int EHBState = -1;
    private System.Windows.Forms.Label label5;
    private System.Windows.Forms.PictureBox pictureBox6;
    private System.Windows.Forms.PictureBox pictureBox8;
    private System.Windows.Forms.PictureBox pictureBox9;
    private System.Windows.Forms.PictureBox pictureBox10;
    private System.Windows.Forms.Label label12;
    private System.Windows.Forms.Label label13;
    private System.Windows.Forms.Label label14;
    private System.Windows.Forms.Label HackedStatusA;
    private System.Windows.Forms.TextBox HackedAData;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.TextBox HackedHData;
    private System.Windows.Forms.Button HackedReset;
    private System.Windows.Forms.Button HackedNextStep;
    private System.Windows.Forms.Label HackedLabelB_A;
    private System.Windows.Forms.Label HackedLabelA_B;
    private System.Windows.Forms.Label HackedStatusB;
    private System.Windows.Forms.TextBox HackedBData;
    private System.Windows.Forms.PictureBox pictureBox7;
    private System.Windows.Forms.PictureBox pictureBox11;
    private System.Windows.Forms.PictureBox pictureBox12;
    private System.Windows.Forms.PictureBox pictureBox13;
    private System.Windows.Forms.PictureBox pictureBox14;
    private System.Windows.Forms.PictureBox pictureBox15;
    private System.Windows.Forms.PictureBox pictureBox16;
    private System.Windows.Forms.Label label16;
    private System.Windows.Forms.Label label17;
    private System.Windows.Forms.Label label18;
    private System.Windows.Forms.PictureBox pictureBox17;
    private System.Windows.Forms.Label label20;
    private System.Windows.Forms.Label label22;
    private System.Windows.Forms.Label EnhancedStatusA;
    private System.Windows.Forms.Label EnhancedStatusB;
    private System.Windows.Forms.TextBox EnhancedAData;
    private System.Windows.Forms.TextBox EnhancedHData;
    private System.Windows.Forms.TextBox EnhancedBData;
    private System.Windows.Forms.Label EnhancedLabelA_B;
    private System.Windows.Forms.Label EnhancedLabelB_A;
    private System.Windows.Forms.Label EnhancedLabelB_CA;
    private System.Windows.Forms.Label EnhancedLabelCA_B;
```

```
        private System.Windows.Forms.Button EnhancedNextStep;
        private System.Windows.Forms.Button EnhancedReset;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Label EnhancedStatusCA;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.PictureBox pictureBox18;
        private System.Windows.Forms.Label label15;
        private System.Windows.Forms.PictureBox pictureBox19;
        private System.Windows.Forms.PictureBox pictureBox20;
        private System.Windows.Forms.PictureBox pictureBox21;
        private System.Windows.Forms.PictureBox pictureBox22;
        private System.Windows.Forms.PictureBox pictureBox23;
        private System.Windows.Forms.PictureBox pictureBox24;
        private System.Windows.Forms.Label label27;
        private System.Windows.Forms.Label label28;
        private System.Windows.Forms.Label label29;
        private System.Windows.Forms.Label label30;
        private System.Windows.Forms.Label label31;
        private System.Windows.Forms.Button EHNextStep;
        private System.Windows.Forms.Button EHReset;
        private System.Windows.Forms.Label EHStatusA;
        private System.Windows.Forms.Label EHStatusCA;
        private System.Windows.Forms.Label EHLabelB_CA;
        private System.Windows.Forms.Label EHLabelCA_B;
        private System.Windows.Forms.Label EHLabelB_A;
        private System.Windows.Forms.Label EHLabelA_B;
        private System.Windows.Forms.Label EHStatusB;
        private System.Windows.Forms.TextBox EHHData;
        private System.Windows.Forms.TextBox EHBData;
        private System.Windows.Forms.TextBox EHAData;
    private System.Windows.Forms.TabPage EH_Tab;
    private Label label8;
    private Label label19;
    private TabPage EHB_Tab;
    private TextBox EHBAData;
    private Label label21;
    private Label EHBStatusA;
    private Label label24;
    private PictureBox pictureBox25;
    private Button EHBReset;
    private Button EHBNextStep;
    private Label EHBStatusB;
    private TextBox EHBBData;
    private Label label38;
    private PictureBox pictureBox31;
    private Label EHBStatusCA;
    private Label label25;
    private Label label26;
    private Label label32;
    private Label EHBLabelB_CA;
    private Label EHBLabelCA_B;
    private PictureBox pictureBox26;
    private PictureBox pictureBox27;
    private PictureBox pictureBox28;
    private Label EHBLabelB_A;
    private Label EHBLabelA_B;
```

```
        private PictureBox pictureBox29;
        private PictureBox pictureBox30;
        private Label label23;
        private TextBox EHBHData;
        private IContainer components;

            public Form1()
            {
                  //
                  // Required for Windows Form Designer support
                  //
                  InitializeComponent();

      //_____________________________________________________________________
                  #region Initialize Original
                  OriginalState=-1;
                  OriginalStateA = new ArrayList();
                  OriginalStateA.Add(new State("Discoverable mode",""));
                  OriginalStateA.Add(new State("Broadcast &
listening","BD_ADDRa,CLKa"));
                  OriginalStateA.Add(new State("Check if BD_ADDRb
exists",""));
                  OriginalStateA.Add(new State("Generate a random number
IN_RANDa",""));
                  OriginalStateA.Add(new State("Send IN_RANDa number to
B","IN_RANDa"));
                  OriginalStateA.Add(new State("Receive acknowledgment of
recipt",""));
                  OriginalStateA.Add(new State("Ask user for PIN",""));
                  OriginalStateA.Add(new State("Send PIN to B","PIN"));
                  OriginalStateA.Add(new State("Wait for B's PIN",""));
                  OriginalStateA.Add(new State("Compare PIN Numbers",""));
                  OriginalStateA.Add(new State("Calculate Kini key",""));
                  OriginalStateA.Add(new State("Generate Random number
LK_RANDa",""));
                  OriginalStateA.Add(new State("Calculate LK_Ka",""));
                  OriginalStateA.Add(new State("Calculate Ca",""));
                  OriginalStateA.Add(new State("Send Ca to B","Ca"));
                  OriginalStateA.Add(new State("Wait for B's Cb ",""));
                  OriginalStateA.Add(new State("Calculate LK_RANDb",""));
                  OriginalStateA.Add(new State("Calculate LK_Kb",""));
                  OriginalStateA.Add(new State("Calculate Kab",""));
                  OriginalStateA.Add(new State("Discard Kinit",""));
                  OriginalStateA.Add(new State("Generate a random number
AU_RANDa",""));
                  OriginalStateA.Add(new State("Send AU_RANDa to
B","AU_RANDa"));
                  OriginalStateA.Add(new State("Calculates SRES'",""));
                  OriginalStateA.Add(new State("Wait for SRES",""));
                  OriginalStateA.Add(new State("Compares SRES and
SRES'",""));
                  OriginalStateA.Add(new State("Send Acknowledgment to
B","Acknowledgment"));
                  OriginalStateA.Add(new State("Wait for B's AU_RANDb",""));
                  OriginalStateA.Add(new State("Receive B's AU_RANDb",""));
                  OriginalStateA.Add(new State("Calculates SRES",""));
                  OriginalStateA.Add(new State("Send SRES to B","SRES"));
```

```
                    OriginalStateA.Add(new State("Wait acknowledgment from
B",""));
                    OriginalStateA.Add(new State("Start Encryption
process",""));

                    OriginalStateB = new ArrayList();
                    OriginalStateB.Add(new State("Discoverable mode",""));
                    OriginalStateB.Add(new State("Broadcast &
listening","BD_ADDRb, CLKb"));
                    OriginalStateB.Add(new State("Check if BD_ADDRa
exists",""));
                    OriginalStateB.Add(new State("Wait for A's IN_RANDa",""));
                    OriginalStateB.Add(new State("Receive for A's
IN_RANDa",""));
                    OriginalStateB.Add(new State("Send Acknowledgment to
A","Acknowledgment"));
                    OriginalStateB.Add(new State("Ask user for PIN",""));
                    OriginalStateB.Add(new State("Send PIN to A","PIN"));
                    OriginalStateB.Add(new State("Wait for A's PIN",""));
                    OriginalStateB.Add(new State("Compare PIN Numbers",""));
                    OriginalStateB.Add(new State("Calculate Kinit key",""));
                    OriginalStateB.Add(new State("Generate Random number
LK_RANDb",""));
                    OriginalStateB.Add(new State("Calculate LK_Kb",""));
                    OriginalStateB.Add(new State("Calculate Cb",""));
                    OriginalStateB.Add(new State("Send Cb to A","Cb"));
                    OriginalStateB.Add(new State("Wait for A's Ca",""));
                    OriginalStateB.Add(new State("Calculate LK_RANDa",""));
                    OriginalStateB.Add(new State("Calculate LK_Ka",""));
                    OriginalStateB.Add(new State("Calculate Kab",""));
                    OriginalStateB.Add(new State("Discard Kinit",""));
                    OriginalStateB.Add(new State("Wait for A's AU_RANDa",""));
                    OriginalStateB.Add(new State("Receive  A's AU_RANDa",""));
                    OriginalStateB.Add(new State("Calculates SRES",""));
                    OriginalStateB.Add(new State("Send SRES to A","SRES"));
                    OriginalStateB.Add(new State("Wait for Acknowledment from
A",""));
                    OriginalStateB.Add(new State("Receive Acknowledment from
A",""));
                    OriginalStateB.Add(new State("Generate a random number
AU_RANDb",""));
                    OriginalStateB.Add(new State("Send AU_RANDb to
A'","AU_RANDb"));
                    OriginalStateB.Add(new State("Calculates SRES'",""));
                    OriginalStateB.Add(new State("Wait for A's SRES",""));
                    OriginalStateB.Add(new State("Send Acknowledgment to
A","Acknowledgment"));
                    OriginalStateB.Add(new State("Start Encryption
process",""));
                    #endregion
                    #region Initialize Original-Hacked
                    HackedHData.Text = "From A: BD_ADDRa,CLKa";
                    HackedHData.Text += System.Environment.NewLine+"From B:
BD_ADDRb, CLKb";
                    HackedHData.Text += System.Environment.NewLine+"From A:
IN_RANDa";
```

```
                    HackedHData.Text += System.Environment.NewLine+"From B:
Acknowledgment";
                    HackedHData.Text += System.Environment.NewLine+"From A:
PIN";
                    HackedHData.Text += System.Environment.NewLine+"From B:
PIN";
                    HackedHData.Text += System.Environment.NewLine+"-----------
--- CAN NOW CALCULATE Kab ------------------------";
                    HackedHData.Text += System.Environment.NewLine+"Kab ->From
IN_RANDa, BD_ADDRb, and PIN information";
                    HackedHData.Text += System.Environment.NewLine+"-----------
-----------------------------------------------------------------------";
                    HackedHData.Text += System.Environment.NewLine+"From A:
Ca";
                    HackedHData.Text += System.Environment.NewLine+"From B:
Cb";
                    HackedHData.Text += System.Environment.NewLine+"From A:
AU_RANDa";
                    HackedHData.Text += System.Environment.NewLine+"From B:
SRES";
                    HackedHData.Text += System.Environment.NewLine+"From A:
Acknowledgment";
                    HackedHData.Text += System.Environment.NewLine+"From B:
AU_RANDb";
                    HackedHData.Text += System.Environment.NewLine+"From A:
SRES";
                    HackedHData.Text += System.Environment.NewLine+"From B:
Acknowledgment";


                    HackedState=-1;
                    HackedStateA = new ArrayList();
                    HackedStateA.Add(new State("Discoverable mode",""));
                    HackedStateA.Add(new State("Broadcast &
listening","BD_ADDRa,CLKa"));
                    HackedStateA.Add(new State("Check if BD_ADDRb exists",""));

                    HackedStateA.Add(new State("Generate a random number
AU_RANDa",""));
                    HackedStateA.Add(new State("Send AU_RANDa to
B","AU_RANDa"));
                    HackedStateA.Add(new State("Calculates SRES'",""));
                    HackedStateA.Add(new State("Wait for SRES",""));
                    HackedStateA.Add(new State("Compares SRES and SRES'",""));
                    HackedStateA.Add(new State("Send Acknowledgment to
B","Acknowledgment"));
                    HackedStateA.Add(new State("Wait for B's AU_RANDb",""));
                    HackedStateA.Add(new State("Receive B's AU_RANDb",""));
                    HackedStateA.Add(new State("Calculates SRES",""));
                    HackedStateA.Add(new State("Send SRES to B","SRES"));
                    HackedStateA.Add(new State("Wait acknowledgment from
B",""));
                    HackedStateA.Add(new State("Start Encryption process",""));

                    HackedStateB = new ArrayList();
                    HackedStateB.Add(new State("Discoverable mode",""));
```

```
                    HackedStateB.Add(new State("Broadcast &
listening","BD_ADDRb, CLKb"));
                    HackedStateB.Add(new State("Check if BD_ADDRa exists",""));

                    HackedStateB.Add(new State("Wait for A's AU_RANDa",""));
                    HackedStateB.Add(new State("Receive  A's AU_RANDa",""));
                    HackedStateB.Add(new State("Calculates SRES",""));
                    HackedStateB.Add(new State("Send SRES to A","SRES"));
                    HackedStateB.Add(new State("Wait for Acknowledment from
A",""));
                    HackedStateB.Add(new State("Receive Acknowledment from
A",""));
                    HackedStateB.Add(new State("Generate a random number
AU_RANDb",""));
                    HackedStateB.Add(new State("Send AU_RANDb to
A'","AU_RANDb"));
                    HackedStateB.Add(new State("Calculates SRES'",""));
                    HackedStateB.Add(new State("Wait for A's SRES",""));
                    HackedStateB.Add(new State("Send Acknowledgment to
A","Acknowledgment"));
                    HackedStateB.Add(new State("Start Encryption process",""));
                    #endregion
                    #region Initialize Enhanced
                    OriginalState=-1;
                    EnhancedStateA = new ArrayList();
                    EnhancedStateA.Add(new State("Discoverable mode",""));
                    EnhancedStateA.Add(new State("Broadcast &
listening","BD_ADDRa,CLKa"));
                    EnhancedStateA.Add(new State("Check if BD_ADDRb
exists",""));
                    EnhancedStateA.Add(new State("Generate a random number
IN_RANDa",""));
                    EnhancedStateA.Add(new State("Send IN_RANDa number to
B","IN_RANDa"));
                    EnhancedStateA.Add(new State("Receive acknowledgment of
recipt",""));
                    EnhancedStateA.Add(new State("Ask user for PIN",""));
                    EnhancedStateA.Add(new State("Send PIN to B","PIN"));
                    EnhancedStateA.Add(new State("Wait for B's PIN",""));
                    EnhancedStateA.Add(new State("Compare PIN Numbers",""));
                    EnhancedStateA.Add(new State("Calculate Kini key",""));
                    EnhancedStateA.Add(new State("Generate Random number
LK_RANDa",""));
                    EnhancedStateA.Add(new State("Calculate LK_Ka",""));
                    EnhancedStateA.Add(new State("Calculate Ca",""));
                    EnhancedStateA.Add(new State("Send Ca to B","Ca"));
                    EnhancedStateA.Add(new State("Wait for B's Cb ",""));
                    EnhancedStateA.Add(new State("Calculate LK_RANDb",""));
                    EnhancedStateA.Add(new State("Calculate LK_Kb",""));
                    EnhancedStateA.Add(new State("Calculate Kab",""));
                    EnhancedStateA.Add(new State("Discard Kinit",""));
                    EnhancedStateA.Add(new State("Generate a random number
AU_RANDa",""));
                    EnhancedStateA.Add(new State("Send AU_RANDa to
B","AU_RANDa"));
                    EnhancedStateA.Add(new State("Calculates SRES'",""));
                    EnhancedStateA.Add(new State("Wait for SRES",""));
```

```
                    EnhancedStateA.Add(new State("Compares SRES and
SRES'",""));
                    EnhancedStateA.Add(new State("Send Acknowledgment to
B","Acknowledgment"));
                    EnhancedStateA.Add(new State("Wait for B's AU_RANDb",""));
                    EnhancedStateA.Add(new State("Receive B's AU_RANDb",""));
                    EnhancedStateA.Add(new State("Calculates SRES",""));
                    EnhancedStateA.Add(new State("Send SRES to B","SRES"));
                    EnhancedStateA.Add(new State("Wait acknowledgment from
B",""));
                    EnhancedStateA.Add(new State("Send DA to
B","eA{DA,RANDa}"));
                    EnhancedStateA.Add(new State("Wait for acknowledgment from
B",""));
                    EnhancedStateA.Add(new State("Wait for acknowledgment from
B",""));
                    EnhancedStateA.Add(new State("Receive acknowledgment from
B",""));
                    EnhancedStateA.Add(new State("Receive DB from B",""));
                    EnhancedStateA.Add(new State("Encrypt DB and send to
B","eA{eB{DB,RANDb},CA_RANDa}"));
                    EnhancedStateA.Add(new State("Wait for acknowledgment from
B",""));
                    EnhancedStateA.Add(new State("Wait for acknowledgment from
B",""));
                    EnhancedStateA.Add(new State("Receive
acknowledgment=CA_RANDa from B",""));
                    EnhancedStateA.Add(new State("Check CA_RANDa and send
acknowledgment to B","acknowledgment"));
                    EnhancedStateA.Add(new State("Start Encryption
process",""));

                    EnhancedStateB = new ArrayList();
                    EnhancedStateB.Add(new State("Discoverable mode",""));
                    EnhancedStateB.Add(new State("Broadcast &
listening","BD_ADDRb, CLKb"));
                    EnhancedStateB.Add(new State("Check if BD_ADDRa
exists",""));
                    EnhancedStateB.Add(new State("Wait for A's IN_RANDa",""));
                    EnhancedStateB.Add(new State("Receive for A's
IN_RANDa",""));
                    EnhancedStateB.Add(new State("Send Acknowledgment to
A","Acknowledgment"));
                    EnhancedStateB.Add(new State("Ask user for PIN",""));
                    EnhancedStateB.Add(new State("Send PIN to A","PIN"));
                    EnhancedStateB.Add(new State("Wait for A's PIN",""));
                    EnhancedStateB.Add(new State("Compare PIN Numbers",""));
                    EnhancedStateB.Add(new State("Calculate Kinit key",""));
                    EnhancedStateB.Add(new State("Generate Random number
LK_RANDb",""));
                    EnhancedStateB.Add(new State("Calculate LK_Kb",""));
                    EnhancedStateB.Add(new State("Calculate Cb",""));
                    EnhancedStateB.Add(new State("Send Cb to A","Cb"));
                    EnhancedStateB.Add(new State("Wait for A's Ca",""));
                    EnhancedStateB.Add(new State("Calculate LK_RANDa",""));
                    EnhancedStateB.Add(new State("Calculate LK_Ka",""));
                    EnhancedStateB.Add(new State("Calculate Kab",""));
```

```
                    EnhancedStateB.Add(new State("Discard Kinit",""));
                    EnhancedStateB.Add(new State("Wait for A's AU_RANDa",""));
                    EnhancedStateB.Add(new State("Receive  A's AU_RANDa",""));
                    EnhancedStateB.Add(new State("Calculates SRES",""));
                    EnhancedStateB.Add(new State("Send SRES to A","SRES"));
                    EnhancedStateB.Add(new State("Wait for Acknowledment from
A",""));
                    EnhancedStateB.Add(new State("Receive Acknowledment from
A",""));
                    EnhancedStateB.Add(new State("Generate a random number
AU_RANDb",""));
                    EnhancedStateB.Add(new State("Send AU_RANDb to
A'","AU_RANDb"));
                    EnhancedStateB.Add(new State("Calculates SRES'",""));
                    EnhancedStateB.Add(new State("Wait for A's SRES",""));
                    EnhancedStateB.Add(new State("Send Acknowledgment to
A","Acknowledgment"));
                    EnhancedStateB.Add(new State("Receive DA from A",""));

                    EnhancedStateB.Add(new State("Encrypt DA and sents it to
CA","","eB{eA{DA,RANDa},CA_RANDb}"));
                    EnhancedStateB.Add(new State("Receive Acknowledgment from
CA","",""));
                    EnhancedStateB.Add(new State("Send Acknowledgment to
A","Acknowledgment"));
                    EnhancedStateB.Add(new State("Send DB to
A","eB{DB,RANDb}"));
                    EnhancedStateB.Add(new State("Receive
eA{eB{DB,RANDb},CA_RANDa}",""));
                EnhancedStateB.Add(new State("Send eA{eB{DB,RANDb},CA_RANDa} to
CA", "", "eA{eB{DB,RANDb},CA_RANDa}"));
                    EnhancedStateB.Add(new State("Receive Acknowledgment from
CA(CA_RANDa)","",""));
                    EnhancedStateB.Add(new State("Send Acknowledgment=CA_RANDa
to A","CA_RANDa",""));
                    EnhancedStateB.Add(new State("Receive Acknowledgment from
A","",""));
                    EnhancedStateB.Add(new State("Start Encryption
process",""));

                    EnhancedStateCA = new ArrayList();
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
```

```
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));

                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                EnhancedStateCA.Add(new State("Receive
eB{eA{DA,RANDa},CA_RANDb}from B", ""));
                    EnhancedStateCA.Add(new State("Send Acknowledgment to
B","Acknowledgment"));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                EnhancedStateCA.Add(new State("Receive eA{eB{DB,RANDb},CA_RANDa}
from B", ""));
                    EnhancedStateCA.Add(new State("Send Acknowledgment to
B(RANDdca)","RANDdca"));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    EnhancedStateCA.Add(new State("",""));
                    #endregion
                    #region Initialize Enhanced - Hacked

                    EHHData.Text = "From A: BD_ADDRa,CLKa";
                    EHHData.Text += System.Environment.NewLine+"From B:
BD_ADDRb, CLKb";
                    EHHData.Text += System.Environment.NewLine+"From A:
IN_RANDa";
                    EHHData.Text += System.Environment.NewLine+"From B:
Acknowledgment";
                    EHHData.Text += System.Environment.NewLine+"From A: PIN";
                    EHHData.Text += System.Environment.NewLine+"From B: PIN";
                    EHHData.Text += System.Environment.NewLine+"From A: Ca";
                    EHHData.Text += System.Environment.NewLine+"From B: Cb";
                    EHHData.Text += System.Environment.NewLine+"From A:
AU_RANDa";
                    EHHData.Text += System.Environment.NewLine+"From B: SRES";
                    EHHData.Text += System.Environment.NewLine+"From A:
Acknowledgment";
                    EHHData.Text += System.Environment.NewLine+"From B:
AU_RANDb";
                    EHHData.Text += System.Environment.NewLine+"From A: SRES";
                    EHHData.Text += System.Environment.NewLine+"From B:
Acknowledgment";
                    EHHData.Text += System.Environment.NewLine+"From A:
eA{DA,RANDa}";
```

```
                    EHHData.Text += System.Environment.NewLine+"From B:
Acknowledgment";
                    EHHData.Text += System.Environment.NewLine+"From B:
eB{DB,RANDa}";
                    EHHData.Text += System.Environment.NewLine+"From A:
eA{eB{DB,RANDb},CA_RANDa}";
                    EHHData.Text += System.Environment.NewLine+"From B:
CA_RANDa";
                    EHHData.Text += System.Environment.NewLine+"From A:
acknowledgment ";

                    EHState=-1;
                    EHStateA = new ArrayList();
                    EHStateA.Add(new State("Discoverable mode",""));
                    EHStateA.Add(new State("Broadcast &
listening","BD_ADDRa,CLKa"));
                    EHStateA.Add(new State("Check if BD_ADDRb exists",""));

                    EHStateA.Add(new State("Generate a random number
AU_RANDa",""));
                    EHStateA.Add(new State("Send AU_RANDa to B","AU_RANDa"));
                    EHStateA.Add(new State("Calculates SRES'",""));
                    EHStateA.Add(new State("Wait for SRES",""));
                    EHStateA.Add(new State("Compares SRES and SRES'",""));
                    EHStateA.Add(new State("Send Acknowledgment to
B","Acknowledgment"));
                    EHStateA.Add(new State("Wait for B's AU_RANDb",""));
                    EHStateA.Add(new State("Receive B's AU_RANDb",""));
                    EHStateA.Add(new State("Calculates SRES",""));
                    EHStateA.Add(new State("Send SRES to B","SRES"));
                    EHStateA.Add(new State("Wait acknowledgment from B",""));
                    EHStateA.Add(new State("Send DA to B","eA{DA,RANDa}"));
                    EHStateA.Add(new State("Wait for acknowledgment from
B",""));
                    EHStateA.Add(new State("Wait for acknowledgment from
B",""));
                    EHStateA.Add(new State("Receive acknowledgment from
B",""));
                    EHStateA.Add(new State("Receive DB from B",""));
                    EHStateA.Add(new State("Encrypt DB and send to
B","eA{eB{CB,RANDb},CA_RANDa}"));
                    EHStateA.Add(new State("Wait for acknowledgment from
B",""));
                    EHStateA.Add(new State("Wait for acknowledgment from
B",""));
            EHStateA.Add(new State("Receive acknowledgment=ERROR_eA from B",
""));
                    EHStateA.Add(new State("Send an ERROR Acknowledgment to
B",""));
                    EHStateA.Add(new State("Communication Failed",""));

                    EHStateB = new ArrayList();
                    EHStateB.Add(new State("Discoverable mode",""));
                    EHStateB.Add(new State("Broadcast & listening","BD_ADDRb,
CLKb"));
                    EHStateB.Add(new State("Check if BD_ADDRa exists",""));
```

```
                    EHStateB.Add(new State("Wait for A's AU_RANDa",""));
                    EHStateB.Add(new State("Receive  A's AU_RANDa",""));
                    EHStateB.Add(new State("Calculates SRES",""));
                    EHStateB.Add(new State("Send SRES to A","SRES"));
                    EHStateB.Add(new State("Wait for Acknowledment from
A",""));
                    EHStateB.Add(new State("Receive Acknowledment from A",""));
                    EHStateB.Add(new State("Generate a random number
AU_RANDb",""));
                    EHStateB.Add(new State("Send AU_RANDb to A'","AU_RANDb"));
                    EHStateB.Add(new State("Calculates SRES'",""));
                    EHStateB.Add(new State("Wait for A's SRES",""));
                    EHStateB.Add(new State("Send Acknowledgment to
A","Acknowledgment"));
                    EHStateB.Add(new State("Receive DA from A",""));

            EHStateB.Add(new State("Send eB{eA{DA,RANDa},CA_RANDb} to CA",
"", "eB{eA{DA,RANDa},CA_RANDb}"));
                    EHStateB.Add(new State("Receive Acknowledgment from
CA","",""));
                    EHStateB.Add(new State("Send Acknowledgment to
A","Acknowledgment"));
                    EHStateB.Add(new State("Send DB to A","eB{DB,RANDb}"));
                    EHStateB.Add(new State("Receive
eA{eB{DB,RANDb},CA_RANDa}",""));
            EHStateB.Add(new State("Send eA{eB{DB,RANDb},CA_RANDa} to CA",
"", "eA{eB{DB,RANDb},CA_RANDa}"));
            EHStateB.Add(new State("Receive Acknowledgment from
CA(ERROR_eA)", "", ""));
            EHStateB.Add(new State("Send Acknowledgment=ERROR_eA to A",
"ERROR_eA", ""));
                    EHStateB.Add(new State("Receive Acknowledgment from
A","",""));
                    EHStateB.Add(new State("Communication Failed",""));

                    EHStateCA = new ArrayList();
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));

                    EHStateCA.Add(new State("",""));
                    EHStateCA.Add(new State("",""));
            EHStateCA.Add(new State("Receive eB{eA{DA,RANDa},CA_RANDb} from
B", ""));
                    EHStateCA.Add(new State("Send Acknowledgment to
B","Acknowledgment"));
                    EHStateCA.Add(new State("",""));
```

```
                EHStateCA.Add(new State("",""));
                EHStateCA.Add(new State("",""));
            EHStateCA.Add(new State("Receive eA{eB{DB,RANDb},CA_RANDa} From
B", ""));
            EHStateCA.Add(new State("Send Acknowledgment to B(ERROR_eA)",
"ERROR_eA"));
                EHStateCA.Add(new State("",""));
                EHStateCA.Add(new State("",""));
                EHStateCA.Add(new State("",""));
                #endregion
            #region Initialize Enhanced - HackedB

            EHBHData.Text = "From A: BD_ADDRa,CLKa";
            EHBHData.Text += System.Environment.NewLine + "From B: BD_ADDRb,
CLKb";
            EHBHData.Text += System.Environment.NewLine + "From A: IN_RANDa";
            EHBHData.Text += System.Environment.NewLine + "From B:
Acknowledgment";
            EHBHData.Text += System.Environment.NewLine + "From A: PIN";
            EHBHData.Text += System.Environment.NewLine + "From B: PIN";
            EHBHData.Text += System.Environment.NewLine + "From A: Ca";
            EHBHData.Text += System.Environment.NewLine + "From B: Cb";
            EHBHData.Text += System.Environment.NewLine + "From A: AU_RANDa";
            EHBHData.Text += System.Environment.NewLine + "From B: SRES";
            EHBHData.Text += System.Environment.NewLine + "From A:
Acknowledgment";
            EHBHData.Text += System.Environment.NewLine + "From B: AU_RANDb";
            EHBHData.Text += System.Environment.NewLine + "From A: SRES";
            EHBHData.Text += System.Environment.NewLine + "From B:
Acknowledgment";
            EHBHData.Text += System.Environment.NewLine + "From A:
eA{DA,RANDa}";
            EHBHData.Text += System.Environment.NewLine + "From B:
Acknowledgment";
            EHBHData.Text += System.Environment.NewLine + "From B:
eB{DB,RANDb}";
            EHBHData.Text += System.Environment.NewLine + "From A:
eA{eB{DB,RANDb},CA_RANDa}";
            EHBHData.Text += System.Environment.NewLine + "From B: CA_RANDa";
            EHBHData.Text += System.Environment.NewLine + "From A:
acknowledgment ";

            EHBState = -1;
            EHBStateA = new ArrayList();
            EHBStateA.Add(new State("Discoverable mode", ""));
            EHBStateA.Add(new State("Broadcast & listening",
"BD_ADDRa,CLKa"));
            EHBStateA.Add(new State("Check if BD_ADDRb exists", ""));



            EHBStateA.Add(new State("Generate a random number AU_RANDa",
""));
            EHBStateA.Add(new State("Send AU_RANDa to B", "AU_RANDa"));
            EHBStateA.Add(new State("Calculates SRES'", ""));
            EHBStateA.Add(new State("Wait for SRES", ""));
            EHBStateA.Add(new State("Compares SRES and SRES'", ""));
```

```
            EHBStateA.Add(new State("Send Acknowledgment to B",
"Acknowledgment"));
            EHBStateA.Add(new State("Wait for B's AU_RANDb", ""));
            EHBStateA.Add(new State("Receive B's AU_RANDb", ""));
            EHBStateA.Add(new State("Calculates SRES", ""));
            EHBStateA.Add(new State("Send SRES to B", "SRES"));
            EHBStateA.Add(new State("Wait acknowledgment from B", ""));
            EHBStateA.Add(new State("Send DA to B", "eA{DA,RANDa}"));
            EHBStateA.Add(new State("Wait for acknowledgment from B", ""));
            EHBStateA.Add(new State("Wait for acknowledgment from B", ""));
            EHBStateA.Add(new State("Receive acknowledgment from B", ""));
            EHBStateA.Add(new State("Communication Failed", ""));

            EHBStateB = new ArrayList();
            EHBStateB.Add(new State("Discoverable mode", ""));
            EHBStateB.Add(new State("Broadcast & listening", "BD_ADDRb,
CLKb"));
            EHBStateB.Add(new State("Check if BD_ADDRa exists", ""));


            EHBStateB.Add(new State("Wait for A's AU_RANDa", ""));
            EHBStateB.Add(new State("Receive  A's AU_RANDa", ""));
            EHBStateB.Add(new State("Calculates SRES", ""));
            EHBStateB.Add(new State("Send SRES to A", "SRES"));
            EHBStateB.Add(new State("Wait for Acknowledment from A", ""));
            EHBStateB.Add(new State("Receive Acknowledment from A", ""));
            EHBStateB.Add(new State("Generate a random number AU_RANDb",
""));
            EHBStateB.Add(new State("Send AU_RANDb to A'", "AU_RANDb"));
            EHBStateB.Add(new State("Calculates SRES'", ""));
            EHBStateB.Add(new State("Wait for A's SRES", ""));
            EHBStateB.Add(new State("Send Acknowledgment to A",
"Acknowledgment"));
            EHBStateB.Add(new State("Receive DA from A", ""));

            EHBStateB.Add(new State("Send eB{eA{DA,RANDa},CA_RANDb} to CA",
"", "eB{eA{DA,RANDa},CA_RANDb}"));
            EHBStateB.Add(new State("Receive Acknowledgment from CA", "",
""));
            EHBStateB.Add(new State("Send Acknowledgment to A",
"Acknowledgment = ERROR_eB"));
            EHBStateB.Add(new State("Communication Failed", ""));

            EHBStateCA = new ArrayList();

            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
```

```
            EHBStateCA.Add(new State("", ""));

            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("Receive eB{eA{DA,RANDa},CA_RANDb} from
B", ""));
            EHBStateCA.Add(new State("Send Acknowledgment to B",
"Acknowledgment = ERROR_eB"));
            EHBStateCA.Add(new State("", ""));
            EHBStateCA.Add(new State("", ""));
            #endregion

                  //
                  // TODO: Add any constructor code after InitializeComponent
call
                  //
            }

            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            protected override void Dispose( bool disposing )
            {
                  if( disposing )
                  {
                        if (components != null)
                        {
                              components.Dispose();
                        }
                  }
                  base.Dispose( disposing );
            }

            #region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
            this.components = new System.ComponentModel.Container();
            System.ComponentModel.ComponentResourceManager resources = new
System.ComponentModel.ComponentResourceManager(typeof(Form1));
            this.tabControl1 = new System.Windows.Forms.TabControl();
            this.OriginalTab = new System.Windows.Forms.TabPage();
            this.OriginalReset = new System.Windows.Forms.Button();
            this.OriginalStatusB = new System.Windows.Forms.Label();
            this.OriginalStatusA = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.OriginalHData = new System.Windows.Forms.TextBox();
            this.OriginalBData = new System.Windows.Forms.TextBox();
            this.OriginalAData = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.pictureBox5 = new System.Windows.Forms.PictureBox();
            this.OriginalNextStep = new System.Windows.Forms.Button();
            this.label4 = new System.Windows.Forms.Label();
```

```
this.label3 = new System.Windows.Forms.Label();
this.OriginalLabelB_A = new System.Windows.Forms.Label();
this.OriginalLabelA_B = new System.Windows.Forms.Label();
this.pictureBox4 = new System.Windows.Forms.PictureBox();
this.pictureBox3 = new System.Windows.Forms.PictureBox();
this.pictureBox2 = new System.Windows.Forms.PictureBox();
this.pictureBox1 = new System.Windows.Forms.PictureBox();
this.OriginalHackedTab = new System.Windows.Forms.TabPage();
this.label9 = new System.Windows.Forms.Label();
this.label2 = new System.Windows.Forms.Label();
this.label14 = new System.Windows.Forms.Label();
this.HackedBData = new System.Windows.Forms.TextBox();
this.label13 = new System.Windows.Forms.Label();
this.HackedStatusB = new System.Windows.Forms.Label();
this.label12 = new System.Windows.Forms.Label();
this.pictureBox10 = new System.Windows.Forms.PictureBox();
this.HackedHData = new System.Windows.Forms.TextBox();
this.HackedLabelB_A = new System.Windows.Forms.Label();
this.HackedLabelA_B = new System.Windows.Forms.Label();
this.pictureBox8 = new System.Windows.Forms.PictureBox();
this.pictureBox9 = new System.Windows.Forms.PictureBox();
this.HackedReset = new System.Windows.Forms.Button();
this.HackedNextStep = new System.Windows.Forms.Button();
this.HackedStatusA = new System.Windows.Forms.Label();
this.HackedAData = new System.Windows.Forms.TextBox();
this.label5 = new System.Windows.Forms.Label();
this.pictureBox6 = new System.Windows.Forms.PictureBox();
this.EnhancedTab = new System.Windows.Forms.TabPage();
this.EnhancedStatusB = new System.Windows.Forms.Label();
this.pictureBox11 = new System.Windows.Forms.PictureBox();
this.label8 = new System.Windows.Forms.Label();
this.label10 = new System.Windows.Forms.Label();
this.EnhancedStatusCA = new System.Windows.Forms.Label();
this.label22 = new System.Windows.Forms.Label();
this.EnhancedStatusA = new System.Windows.Forms.Label();
this.label20 = new System.Windows.Forms.Label();
this.pictureBox17 = new System.Windows.Forms.PictureBox();
this.EnhancedHData = new System.Windows.Forms.TextBox();
this.EnhancedBData = new System.Windows.Forms.TextBox();
this.EnhancedAData = new System.Windows.Forms.TextBox();
this.label16 = new System.Windows.Forms.Label();
this.label17 = new System.Windows.Forms.Label();
this.label18 = new System.Windows.Forms.Label();
this.EnhancedLabelB_CA = new System.Windows.Forms.Label();
this.EnhancedLabelCA_B = new System.Windows.Forms.Label();
this.pictureBox15 = new System.Windows.Forms.PictureBox();
this.pictureBox16 = new System.Windows.Forms.PictureBox();
this.pictureBox14 = new System.Windows.Forms.PictureBox();
this.pictureBox13 = new System.Windows.Forms.PictureBox();
this.EnhancedLabelB_A = new System.Windows.Forms.Label();
this.EnhancedLabelA_B = new System.Windows.Forms.Label();
this.pictureBox12 = new System.Windows.Forms.PictureBox();
this.EnhancedReset = new System.Windows.Forms.Button();
this.EnhancedNextStep = new System.Windows.Forms.Button();
this.pictureBox7 = new System.Windows.Forms.PictureBox();
this.EH_Tab = new System.Windows.Forms.TabPage();
this.EHStatusCA = new System.Windows.Forms.Label();
```

```
            this.label19 = new System.Windows.Forms.Label();
            this.label31 = new System.Windows.Forms.Label();
            this.label30 = new System.Windows.Forms.Label();
            this.EHStatusB = new System.Windows.Forms.Label();
            this.label27 = new System.Windows.Forms.Label();
            this.EHHData = new System.Windows.Forms.TextBox();
            this.EHBData = new System.Windows.Forms.TextBox();
            this.EHAData = new System.Windows.Forms.TextBox();
            this.label28 = new System.Windows.Forms.Label();
            this.label29 = new System.Windows.Forms.Label();
            this.label15 = new System.Windows.Forms.Label();
            this.EHLabelB_CA = new System.Windows.Forms.Label();
            this.EHLabelCA_B = new System.Windows.Forms.Label();
            this.pictureBox19 = new System.Windows.Forms.PictureBox();
            this.pictureBox20 = new System.Windows.Forms.PictureBox();
            this.pictureBox21 = new System.Windows.Forms.PictureBox();
            this.pictureBox22 = new System.Windows.Forms.PictureBox();
            this.EHLabelB_A = new System.Windows.Forms.Label();
            this.EHLabelA_B = new System.Windows.Forms.Label();
            this.pictureBox23 = new System.Windows.Forms.PictureBox();
            this.pictureBox24 = new System.Windows.Forms.PictureBox();
            this.pictureBox18 = new System.Windows.Forms.PictureBox();
            this.EHStatusA = new System.Windows.Forms.Label();
            this.label11 = new System.Windows.Forms.Label();
            this.EHReset = new System.Windows.Forms.Button();
            this.EHNextStep = new System.Windows.Forms.Button();
            this.MainMenu = new
System.Windows.Forms.MainMenu(this.components);
            this.FileMenu = new System.Windows.Forms.MenuItem();
            this.ExitMenu = new System.Windows.Forms.MenuItem();
            this.EHB_Tab = new System.Windows.Forms.TabPage();
            this.EHBNextStep = new System.Windows.Forms.Button();
            this.EHBReset = new System.Windows.Forms.Button();
            this.pictureBox25 = new System.Windows.Forms.PictureBox();
            this.label21 = new System.Windows.Forms.Label();
            this.EHBStatusA = new System.Windows.Forms.Label();
            this.label24 = new System.Windows.Forms.Label();
            this.EHBAData = new System.Windows.Forms.TextBox();
            this.EHBStatusCA = new System.Windows.Forms.Label();
            this.label25 = new System.Windows.Forms.Label();
            this.label26 = new System.Windows.Forms.Label();
            this.label32 = new System.Windows.Forms.Label();
            this.EHBLabelB_CA = new System.Windows.Forms.Label();
            this.EHBLabelCA_B = new System.Windows.Forms.Label();
            this.pictureBox26 = new System.Windows.Forms.PictureBox();
            this.pictureBox27 = new System.Windows.Forms.PictureBox();
            this.pictureBox28 = new System.Windows.Forms.PictureBox();
            this.EHBLabelB_A = new System.Windows.Forms.Label();
            this.EHBLabelA_B = new System.Windows.Forms.Label();
            this.pictureBox29 = new System.Windows.Forms.PictureBox();
            this.pictureBox30 = new System.Windows.Forms.PictureBox();
            this.EHBStatusB = new System.Windows.Forms.Label();
            this.EHBBData = new System.Windows.Forms.TextBox();
            this.label38 = new System.Windows.Forms.Label();
            this.pictureBox31 = new System.Windows.Forms.PictureBox();
            this.label23 = new System.Windows.Forms.Label();
            this.EHBHData = new System.Windows.Forms.TextBox();
```

```
            this.tabControl1.SuspendLayout();
            this.OriginalTab.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox5)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox4)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox3)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginInit();
            this.OriginalHackedTab.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox10)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox8)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox9)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox6)).BeginInit();
            this.EnhancedTab.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox11)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox17)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox15)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox16)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox14)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox13)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox12)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox7)).BeginInit();
            this.EH_Tab.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox19)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox20)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox21)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox22)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox23)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox24)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox18)).BeginInit();
            this.EHB_Tab.SuspendLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox25)).BeginInit();
```

```
((System.ComponentModel.ISupportInitialize)(this.pictureBox26)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox27)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox28)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox29)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox30)).BeginInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox31)).BeginInit();
            this.SuspendLayout();
            //
            // tabControl1
            //
            this.tabControl1.Controls.Add(this.OriginalTab);
            this.tabControl1.Controls.Add(this.OriginalHackedTab);
            this.tabControl1.Controls.Add(this.EnhancedTab);
            this.tabControl1.Controls.Add(this.EH_Tab);
            this.tabControl1.Controls.Add(this.EHB_Tab);
            this.tabControl1.Location = new System.Drawing.Point(0, 0);
            this.tabControl1.Name = "tabControl1";
            this.tabControl1.SelectedIndex = 0;
            this.tabControl1.Size = new System.Drawing.Size(1120, 808);
            this.tabControl1.TabIndex = 0;
            //
            // OriginalTab
            //
            this.OriginalTab.BackColor = System.Drawing.SystemColors.Window;
            this.OriginalTab.Controls.Add(this.OriginalReset);
            this.OriginalTab.Controls.Add(this.OriginalStatusB);
            this.OriginalTab.Controls.Add(this.OriginalStatusA);
            this.OriginalTab.Controls.Add(this.label7);
            this.OriginalTab.Controls.Add(this.label6);
            this.OriginalTab.Controls.Add(this.OriginalHData);
            this.OriginalTab.Controls.Add(this.OriginalBData);
            this.OriginalTab.Controls.Add(this.OriginalAData);
            this.OriginalTab.Controls.Add(this.label1);
            this.OriginalTab.Controls.Add(this.pictureBox5);
            this.OriginalTab.Controls.Add(this.OriginalNextStep);
            this.OriginalTab.Controls.Add(this.label4);
            this.OriginalTab.Controls.Add(this.label3);
            this.OriginalTab.Controls.Add(this.OriginalLabelB_A);
            this.OriginalTab.Controls.Add(this.OriginalLabelA_B);
            this.OriginalTab.Controls.Add(this.pictureBox4);
            this.OriginalTab.Controls.Add(this.pictureBox3);
            this.OriginalTab.Controls.Add(this.pictureBox2);
            this.OriginalTab.Controls.Add(this.pictureBox1);
            this.OriginalTab.Location = new System.Drawing.Point(4, 22);
            this.OriginalTab.Name = "OriginalTab";
            this.OriginalTab.Size = new System.Drawing.Size(1112, 782);
            this.OriginalTab.TabIndex = 0;
            this.OriginalTab.Text = "Original";
            //
            // OriginalReset
            //
```

```
            this.OriginalReset.BackColor =
System.Drawing.SystemColors.Control;
            this.OriginalReset.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.OriginalReset.Location = new System.Drawing.Point(344, 8);
            this.OriginalReset.Name = "OriginalReset";
            this.OriginalReset.Size = new System.Drawing.Size(288, 24);
            this.OriginalReset.TabIndex = 20;
            this.OriginalReset.Text = "Reset";
            this.OriginalReset.UseVisualStyleBackColor = false;
            this.OriginalReset.Click += new
System.EventHandler(this.OriginalReset_Click);
            //
            // OriginalStatusB
            //
            this.OriginalStatusB.Location = new System.Drawing.Point(685,
320);
            this.OriginalStatusB.Name = "OriginalStatusB";
            this.OriginalStatusB.Size = new System.Drawing.Size(296, 16);
            this.OriginalStatusB.TabIndex = 19;
            this.OriginalStatusB.Text = "_";
            //
            // OriginalStatusA
            //
            this.OriginalStatusA.Location = new System.Drawing.Point(57,
320);
            this.OriginalStatusA.Name = "OriginalStatusA";
            this.OriginalStatusA.Size = new System.Drawing.Size(328, 16);
            this.OriginalStatusA.TabIndex = 18;
            this.OriginalStatusA.Text = "_";
            //
            // label7
            //
            this.label7.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label7.Location = new System.Drawing.Point(636, 320);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(56, 16);
            this.label7.TabIndex = 17;
            this.label7.Text = "B Status:";
            //
            // label6
            //
            this.label6.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label6.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label6.Location = new System.Drawing.Point(8, 320);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(56, 16);
            this.label6.TabIndex = 16;
            this.label6.Text = "A Status:";
            //
            // OriginalHData
            //
```

```
            this.OriginalHData.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(255)))),
((int)(((byte)(192)))));
            this.OriginalHData.Location = new System.Drawing.Point(312, 452);
            this.OriginalHData.Multiline = true;
            this.OriginalHData.Name = "OriginalHData";
            this.OriginalHData.ScrollBars =
System.Windows.Forms.ScrollBars.Both;
            this.OriginalHData.Size = new System.Drawing.Size(337, 205);
            this.OriginalHData.TabIndex = 13;
            //
            // OriginalBData
            //
            this.OriginalBData.BackColor = System.Drawing.Color.SkyBlue;
            this.OriginalBData.Location = new System.Drawing.Point(688, 367);
            this.OriginalBData.Multiline = true;
            this.OriginalBData.Name = "OriginalBData";
            this.OriginalBData.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
            this.OriginalBData.Size = new System.Drawing.Size(280, 290);
            this.OriginalBData.TabIndex = 12;
            //
            // OriginalAData
            //
            this.OriginalAData.BackColor = System.Drawing.Color.SkyBlue;
            this.OriginalAData.Location = new System.Drawing.Point(3, 367);
            this.OriginalAData.Multiline = true;
            this.OriginalAData.Name = "OriginalAData";
            this.OriginalAData.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
            this.OriginalAData.Size = new System.Drawing.Size(269, 290);
            this.OriginalAData.TabIndex = 11;
            //
            // label1
            //
            this.label1.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label1.ForeColor = System.Drawing.Color.Red;
            this.label1.Location = new System.Drawing.Point(440, 427);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(72, 16);
            this.label1.TabIndex = 10;
            this.label1.Text = "Hacker";
            //
            // pictureBox5
            //
            this.pictureBox5.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox5.Image")));
            this.pictureBox5.Location = new System.Drawing.Point(361, 256);
            this.pictureBox5.Name = "pictureBox5";
            this.pictureBox5.Size = new System.Drawing.Size(232, 168);
            this.pictureBox5.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox5.TabIndex = 9;
            this.pictureBox5.TabStop = false;
            //
```

```
            // OriginalNextStep
            //
            this.OriginalNextStep.BackColor =
System.Drawing.SystemColors.Control;
            this.OriginalNextStep.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.OriginalNextStep.Location = new System.Drawing.Point(8, 8);
            this.OriginalNextStep.Name = "OriginalNextStep";
            this.OriginalNextStep.Size = new System.Drawing.Size(288, 24);
            this.OriginalNextStep.TabIndex = 8;
            this.OriginalNextStep.Text = "NextStep";
            this.OriginalNextStep.UseVisualStyleBackColor = false;
            this.OriginalNextStep.Click += new
System.EventHandler(this.OriginalNextStep_Click);
            //
            // label4
            //
            this.label4.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label4.ForeColor =
System.Drawing.SystemColors.ActiveCaption;
            this.label4.Location = new System.Drawing.Point(744, 344);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(172, 16);
            this.label4.TabIndex = 7;
            this.label4.Text = "Bluetooth Device B";
            //
            // label3
            //
            this.label3.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label3.ForeColor =
System.Drawing.SystemColors.ActiveCaption;
            this.label3.Location = new System.Drawing.Point(48, 344);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(179, 21);
            this.label3.TabIndex = 6;
            this.label3.Text = "Bluetooth Device A";
            //
            // OriginalLabelB_A
            //
            this.OriginalLabelB_A.Location = new System.Drawing.Point(283,
154);
            this.OriginalLabelB_A.Name = "OriginalLabelB_A";
            this.OriginalLabelB_A.Size = new System.Drawing.Size(320, 16);
            this.OriginalLabelB_A.TabIndex = 5;
            //
            // OriginalLabelA_B
            //
            this.OriginalLabelA_B.Location = new System.Drawing.Point(352,
68);
            this.OriginalLabelA_B.Name = "OriginalLabelA_B";
            this.OriginalLabelA_B.Size = new System.Drawing.Size(320, 16);
            this.OriginalLabelA_B.TabIndex = 4;
```

```
            //
            // pictureBox4
            //
            this.pictureBox4.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox4.Image")));
            this.pictureBox4.Location = new System.Drawing.Point(349, 176);
            this.pictureBox4.Name = "pictureBox4";
            this.pictureBox4.Size = new System.Drawing.Size(256, 24);
            this.pictureBox4.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox4.TabIndex = 3;
            this.pictureBox4.TabStop = false;
            //
            // pictureBox3
            //
            this.pictureBox3.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox3.Image")));
            this.pictureBox3.Location = new System.Drawing.Point(349, 88);
            this.pictureBox3.Name = "pictureBox3";
            this.pictureBox3.Size = new System.Drawing.Size(256, 24);
            this.pictureBox3.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox3.TabIndex = 2;
            this.pictureBox3.TabStop = false;
            //
            // pictureBox2
            //
            this.pictureBox2.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox2.Image")));
            this.pictureBox2.Location = new System.Drawing.Point(685, 53);
            this.pictureBox2.Name = "pictureBox2";
            this.pictureBox2.Size = new System.Drawing.Size(272, 256);
            this.pictureBox2.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox2.TabIndex = 1;
            this.pictureBox2.TabStop = false;
            //
            // pictureBox1
            //
            this.pictureBox1.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox1.Image")));
            this.pictureBox1.Location = new System.Drawing.Point(0, 40);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(272, 280);
            this.pictureBox1.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox1.TabIndex = 0;
            this.pictureBox1.TabStop = false;
            //
            // OriginalHackedTab
            //
            this.OriginalHackedTab.BackColor =
System.Drawing.SystemColors.Window;
            this.OriginalHackedTab.Controls.Add(this.label9);
            this.OriginalHackedTab.Controls.Add(this.label2);
            this.OriginalHackedTab.Controls.Add(this.label14);
            this.OriginalHackedTab.Controls.Add(this.HackedBData);
```

```
            this.OriginalHackedTab.Controls.Add(this.label13);
            this.OriginalHackedTab.Controls.Add(this.HackedStatusB);
            this.OriginalHackedTab.Controls.Add(this.label12);
            this.OriginalHackedTab.Controls.Add(this.pictureBox10);
            this.OriginalHackedTab.Controls.Add(this.HackedHData);
            this.OriginalHackedTab.Controls.Add(this.HackedLabelB_A);
            this.OriginalHackedTab.Controls.Add(this.HackedLabelA_B);
            this.OriginalHackedTab.Controls.Add(this.pictureBox8);
            this.OriginalHackedTab.Controls.Add(this.pictureBox9);
            this.OriginalHackedTab.Controls.Add(this.HackedReset);
            this.OriginalHackedTab.Controls.Add(this.HackedNextStep);
            this.OriginalHackedTab.Controls.Add(this.HackedStatusA);
            this.OriginalHackedTab.Controls.Add(this.HackedAData);
            this.OriginalHackedTab.Controls.Add(this.label5);
            this.OriginalHackedTab.Controls.Add(this.pictureBox6);
            this.OriginalHackedTab.Location = new System.Drawing.Point(4,
22);
            this.OriginalHackedTab.Name = "OriginalHackedTab";
            this.OriginalHackedTab.Size = new System.Drawing.Size(1112, 782);
            this.OriginalHackedTab.TabIndex = 1;
            this.OriginalHackedTab.Text = "Original - Hacked";
            //
            // label9
            //
            this.label9.BackColor = System.Drawing.SystemColors.Window;
            this.label9.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label9.ForeColor = System.Drawing.Color.Red;
            this.label9.Location = new System.Drawing.Point(992, 352);
            this.label9.Name = "label9";
            this.label9.Size = new System.Drawing.Size(72, 24);
            this.label9.TabIndex = 39;
            this.label9.Text = "(Hacker)";
            //
            // label2
            //
            this.label2.Font = new System.Drawing.Font("Microsoft Sans
Serif", 10F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label2.ForeColor = System.Drawing.Color.Firebrick;
            this.label2.Location = new System.Drawing.Point(362, 361);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(226, 24);
            this.label2.TabIndex = 38;
            this.label2.Text = "Known Information from Hack";
            //
            // label14
            //
            this.label14.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label14.Location = new System.Drawing.Point(0, 324);
            this.label14.Name = "label14";
            this.label14.Size = new System.Drawing.Size(56, 16);
            this.label14.TabIndex = 37;
            this.label14.Text = "A Status:";
            //
```

```
            // HackedBData
            //
            this.HackedBData.BackColor = System.Drawing.Color.SkyBlue;
            this.HackedBData.Location = new System.Drawing.Point(690, 367);
            this.HackedBData.Multiline = true;
            this.HackedBData.Name = "HackedBData";
            this.HackedBData.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
            this.HackedBData.Size = new System.Drawing.Size(272, 283);
            this.HackedBData.TabIndex = 36;
            //
            // label13
            //
            this.label13.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label13.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(128)))),
((int)(((byte)(0)))));
            this.label13.Location = new System.Drawing.Point(707, 345);
            this.label13.Name = "label13";
            this.label13.Size = new System.Drawing.Size(255, 25);
            this.label13.TabIndex = 35;
            this.label13.Text = "Bluetooth Device B = Hacker";
            //
            // HackedStatusB
            //
            this.HackedStatusB.Location = new System.Drawing.Point(687, 324);
            this.HackedStatusB.Name = "HackedStatusB";
            this.HackedStatusB.Size = new System.Drawing.Size(296, 16);
            this.HackedStatusB.TabIndex = 34;
            this.HackedStatusB.Text = "_";
            //
            // label12
            //
            this.label12.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label12.Location = new System.Drawing.Point(640, 324);
            this.label12.Name = "label12";
            this.label12.Size = new System.Drawing.Size(56, 16);
            this.label12.TabIndex = 33;
            this.label12.Text = "B Status:";
            //
            // pictureBox10
            //
            this.pictureBox10.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox10.Image")));
            this.pictureBox10.Location = new System.Drawing.Point(690, 40);
            this.pictureBox10.Name = "pictureBox10";
            this.pictureBox10.Size = new System.Drawing.Size(272, 280);
            this.pictureBox10.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox10.TabIndex = 32;
            this.pictureBox10.TabStop = false;
            //
            // HackedHData
            //
```

```
            this.HackedHData.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(255)))),
((int)(((byte)(192)))));
            this.HackedHData.Location = new System.Drawing.Point(329, 388);
            this.HackedHData.Multiline = true;
            this.HackedHData.Name = "HackedHData";
            this.HackedHData.ScrollBars =
System.Windows.Forms.ScrollBars.Both;
            this.HackedHData.Size = new System.Drawing.Size(304, 262);
            this.HackedHData.TabIndex = 31;
            //
            // HackedLabelB_A
            //
            this.HackedLabelB_A.Location = new System.Drawing.Point(277,
170);
            this.HackedLabelB_A.Name = "HackedLabelB_A";
            this.HackedLabelB_A.Size = new System.Drawing.Size(320, 16);
            this.HackedLabelB_A.TabIndex = 28;
            //
            // HackedLabelA_B
            //
            this.HackedLabelA_B.Location = new System.Drawing.Point(346, 82);
            this.HackedLabelA_B.Name = "HackedLabelA_B";
            this.HackedLabelA_B.Size = new System.Drawing.Size(320, 16);
            this.HackedLabelA_B.TabIndex = 27;
            //
            // pictureBox8
            //
            this.pictureBox8.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox8.Image")));
            this.pictureBox8.Location = new System.Drawing.Point(344, 190);
            this.pictureBox8.Name = "pictureBox8";
            this.pictureBox8.Size = new System.Drawing.Size(256, 24);
            this.pictureBox8.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox8.TabIndex = 26;
            this.pictureBox8.TabStop = false;
            //
            // pictureBox9
            //
            this.pictureBox9.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox9.Image")));
            this.pictureBox9.Location = new System.Drawing.Point(344, 101);
            this.pictureBox9.Name = "pictureBox9";
            this.pictureBox9.Size = new System.Drawing.Size(256, 24);
            this.pictureBox9.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox9.TabIndex = 25;
            this.pictureBox9.TabStop = false;
            //
            // HackedReset
            //
            this.HackedReset.BackColor = System.Drawing.SystemColors.Control;
            this.HackedReset.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.HackedReset.Location = new System.Drawing.Point(345, 8);
```

```
            this.HackedReset.Name = "HackedReset";
            this.HackedReset.Size = new System.Drawing.Size(288, 24);
            this.HackedReset.TabIndex = 24;
            this.HackedReset.Text = "Reset";
            this.HackedReset.UseVisualStyleBackColor = false;
            this.HackedReset.Click += new
System.EventHandler(this.HackedReset_Click);
            //
            // HackedNextStep
            //
            this.HackedNextStep.BackColor =
System.Drawing.SystemColors.Control;
            this.HackedNextStep.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.HackedNextStep.Location = new System.Drawing.Point(8, 8);
            this.HackedNextStep.Name = "HackedNextStep";
            this.HackedNextStep.Size = new System.Drawing.Size(288, 24);
            this.HackedNextStep.TabIndex = 23;
            this.HackedNextStep.Text = "NextStep";
            this.HackedNextStep.UseVisualStyleBackColor = false;
            this.HackedNextStep.Click += new
System.EventHandler(this.HackedNextStep_Click);
            //
            // HackedStatusA
            //
            this.HackedStatusA.Location = new System.Drawing.Point(49, 323);
            this.HackedStatusA.Name = "HackedStatusA";
            this.HackedStatusA.Size = new System.Drawing.Size(328, 16);
            this.HackedStatusA.TabIndex = 22;
            this.HackedStatusA.Text = "__";
            //
            // HackedAData
            //
            this.HackedAData.BackColor = System.Drawing.Color.SkyBlue;
            this.HackedAData.Location = new System.Drawing.Point(0, 367);
            this.HackedAData.Multiline = true;
            this.HackedAData.Name = "HackedAData";
            this.HackedAData.ScrollBars =
System.Windows.Forms.ScrollBars.Both;
            this.HackedAData.Size = new System.Drawing.Size(272, 283);
            this.HackedAData.TabIndex = 21;
            //
            // label5
            //
            this.label5.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label5.ForeColor =
System.Drawing.SystemColors.ActiveCaption;
            this.label5.Location = new System.Drawing.Point(48, 345);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(172, 16);
            this.label5.TabIndex = 20;
            this.label5.Text = "Bluetooth Device A";
            //
            // pictureBox6
```

```
            //
            this.pictureBox6.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox6.Image")));
            this.pictureBox6.Location = new System.Drawing.Point(0, 40);
            this.pictureBox6.Name = "pictureBox6";
            this.pictureBox6.Size = new System.Drawing.Size(272, 280);
            this.pictureBox6.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox6.TabIndex = 19;
            this.pictureBox6.TabStop = false;
            //
            // EnhancedTab
            //
            this.EnhancedTab.BackColor = System.Drawing.SystemColors.Window;
            this.EnhancedTab.Controls.Add(this.EnhancedStatusB);
            this.EnhancedTab.Controls.Add(this.pictureBox11);
            this.EnhancedTab.Controls.Add(this.label8);
            this.EnhancedTab.Controls.Add(this.label10);
            this.EnhancedTab.Controls.Add(this.EnhancedStatusCA);
            this.EnhancedTab.Controls.Add(this.label22);
            this.EnhancedTab.Controls.Add(this.EnhancedStatusA);
            this.EnhancedTab.Controls.Add(this.label20);
            this.EnhancedTab.Controls.Add(this.pictureBox17);
            this.EnhancedTab.Controls.Add(this.EnhancedHData);
            this.EnhancedTab.Controls.Add(this.EnhancedBData);
            this.EnhancedTab.Controls.Add(this.EnhancedAData);
            this.EnhancedTab.Controls.Add(this.label16);
            this.EnhancedTab.Controls.Add(this.label17);
            this.EnhancedTab.Controls.Add(this.label18);
            this.EnhancedTab.Controls.Add(this.EnhancedLabelB_CA);
            this.EnhancedTab.Controls.Add(this.EnhancedLabelCA_B);
            this.EnhancedTab.Controls.Add(this.pictureBox15);
            this.EnhancedTab.Controls.Add(this.pictureBox16);
            this.EnhancedTab.Controls.Add(this.pictureBox14);
            this.EnhancedTab.Controls.Add(this.pictureBox13);
            this.EnhancedTab.Controls.Add(this.EnhancedLabelB_A);
            this.EnhancedTab.Controls.Add(this.EnhancedLabelA_B);
            this.EnhancedTab.Controls.Add(this.pictureBox12);
            this.EnhancedTab.Controls.Add(this.EnhancedReset);
            this.EnhancedTab.Controls.Add(this.EnhancedNextStep);
            this.EnhancedTab.Controls.Add(this.pictureBox7);
            this.EnhancedTab.Location = new System.Drawing.Point(4, 22);
            this.EnhancedTab.Name = "EnhancedTab";
            this.EnhancedTab.Size = new System.Drawing.Size(1112, 782);
            this.EnhancedTab.TabIndex = 2;
            this.EnhancedTab.Text = "Enhanced";
            //
            // EnhancedStatusB
            //
            this.EnhancedStatusB.Font = new System.Drawing.Font("Microsoft
Sans Serif", 7.5F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EnhancedStatusB.Location = new System.Drawing.Point(743,
323);
            this.EnhancedStatusB.Name = "EnhancedStatusB";
            this.EnhancedStatusB.Size = new System.Drawing.Size(256, 16);
            this.EnhancedStatusB.TabIndex = 50;
```

```
            this.EnhancedStatusB.Text = "_";
            //
            // pictureBox11
            //
            this.pictureBox11.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox11.Image")));
            this.pictureBox11.Location = new System.Drawing.Point(304, 319);
            this.pictureBox11.Name = "pictureBox11";
            this.pictureBox11.Size = new System.Drawing.Size(384, 24);
            this.pictureBox11.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox11.TabIndex = 31;
            this.pictureBox11.TabStop = false;
            //
            // label8
            //
            this.label8.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label8.ForeColor = System.Drawing.Color.Red;
            this.label8.Location = new System.Drawing.Point(416, 95);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(35, 22);
            this.label8.TabIndex = 53;
            this.label8.Text = "CA";
            //
            // label10
            //
            this.label10.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label10.Location = new System.Drawing.Point(336, 53);
            this.label10.Name = "label10";
            this.label10.Size = new System.Drawing.Size(64, 16);
            this.label10.TabIndex = 52;
            this.label10.Text = "CA Status:";
            //
            // EnhancedStatusCA
            //
            this.EnhancedStatusCA.ForeColor = System.Drawing.Color.Red;
            this.EnhancedStatusCA.Location = new System.Drawing.Point(400,
53);
            this.EnhancedStatusCA.Name = "EnhancedStatusCA";
            this.EnhancedStatusCA.Size = new System.Drawing.Size(376, 16);
            this.EnhancedStatusCA.TabIndex = 51;
            this.EnhancedStatusCA.Text = "__";
            //
            // label22
            //
            this.label22.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label22.Location = new System.Drawing.Point(694, 323);
            this.label22.Name = "label22";
            this.label22.Size = new System.Drawing.Size(56, 16);
            this.label22.TabIndex = 49;
            this.label22.Text = "B Status:";
            //
            // EnhancedStatusA
```

```
            //
            this.EnhancedStatusA.Font = new System.Drawing.Font("Microsoft
Sans Serif", 7.5F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EnhancedStatusA.Location = new System.Drawing.Point(48,
323);
            this.EnhancedStatusA.Name = "EnhancedStatusA";
            this.EnhancedStatusA.Size = new System.Drawing.Size(260, 16);
            this.EnhancedStatusA.TabIndex = 48;
            this.EnhancedStatusA.Text = "_";
            //
            // label20
            //
            this.label20.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label20.Location = new System.Drawing.Point(0, 323);
            this.label20.Name = "label20";
            this.label20.Size = new System.Drawing.Size(56, 16);
            this.label20.TabIndex = 47;
            this.label20.Text = "A Status:";
            //
            // pictureBox17
            //
            this.pictureBox17.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox17.Image")));
            this.pictureBox17.Location = new System.Drawing.Point(420, 384);
            this.pictureBox17.Name = "pictureBox17";
            this.pictureBox17.Size = new System.Drawing.Size(128, 96);
            this.pictureBox17.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox17.TabIndex = 46;
            this.pictureBox17.TabStop = false;
            //
            // EnhancedHData
            //
            this.EnhancedHData.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(255)))),
((int)(((byte)(192)))));
            this.EnhancedHData.Location = new System.Drawing.Point(312, 523);
            this.EnhancedHData.Multiline = true;
            this.EnhancedHData.Name = "EnhancedHData";
            this.EnhancedHData.RightToLeft =
System.Windows.Forms.RightToLeft.No;
            this.EnhancedHData.ScrollBars =
System.Windows.Forms.ScrollBars.Both;
            this.EnhancedHData.Size = new System.Drawing.Size(376, 117);
            this.EnhancedHData.TabIndex = 45;
            //
            // EnhancedBData
            //
            this.EnhancedBData.BackColor = System.Drawing.Color.SkyBlue;
            this.EnhancedBData.Location = new System.Drawing.Point(729, 375);
            this.EnhancedBData.Multiline = true;
            this.EnhancedBData.Name = "EnhancedBData";
            this.EnhancedBData.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
            this.EnhancedBData.Size = new System.Drawing.Size(247, 265);
```

```
            this.EnhancedBData.TabIndex = 44;
            //
            // EnhancedAData
            //
            this.EnhancedAData.BackColor = System.Drawing.Color.SkyBlue;
            this.EnhancedAData.Location = new System.Drawing.Point(3, 375);
            this.EnhancedAData.Multiline = true;
            this.EnhancedAData.Name = "EnhancedAData";
            this.EnhancedAData.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical;
            this.EnhancedAData.Size = new System.Drawing.Size(261, 278);
            this.EnhancedAData.TabIndex = 43;
            //
            // label16
            //
            this.label16.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label16.ForeColor = System.Drawing.Color.Red;
            this.label16.Location = new System.Drawing.Point(451, 493);
            this.label16.Name = "label16";
            this.label16.Size = new System.Drawing.Size(72, 16);
            this.label16.TabIndex = 42;
            this.label16.Text = "Hacker";
            //
            // label17
            //
            this.label17.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label17.ForeColor =
System.Drawing.SystemColors.ActiveCaption;
            this.label17.Location = new System.Drawing.Point(747, 349);
            this.label17.Name = "label17";
            this.label17.Size = new System.Drawing.Size(173, 16);
            this.label17.TabIndex = 41;
            this.label17.Text = "Bluetooth Device B";
            //
            // label18
            //
            this.label18.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label18.ForeColor =
System.Drawing.SystemColors.ActiveCaption;
            this.label18.Location = new System.Drawing.Point(36, 349);
            this.label18.Name = "label18";
            this.label18.Size = new System.Drawing.Size(181, 16);
            this.label18.TabIndex = 40;
            this.label18.Text = "Bluetooth Device A";
            //
            // EnhancedLabelB_CA
            //
            this.EnhancedLabelB_CA.Location = new System.Drawing.Point(500,
163);
            this.EnhancedLabelB_CA.Name = "EnhancedLabelB_CA";
            this.EnhancedLabelB_CA.Size = new System.Drawing.Size(192, 16);
```

```
            this.EnhancedLabelB_CA.TabIndex = 39;
            //
            // EnhancedLabelCA_B
            //
            this.EnhancedLabelCA_B.Location = new System.Drawing.Point(512,
103);
            this.EnhancedLabelCA_B.Name = "EnhancedLabelCA_B";
            this.EnhancedLabelCA_B.Size = new System.Drawing.Size(192, 16);
            this.EnhancedLabelCA_B.TabIndex = 38;
            //
            // pictureBox15
            //
            this.pictureBox15.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox15.Image")));
            this.pictureBox15.Location = new System.Drawing.Point(508, 182);
            this.pictureBox15.Name = "pictureBox15";
            this.pictureBox15.Size = new System.Drawing.Size(184, 24);
            this.pictureBox15.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox15.TabIndex = 37;
            this.pictureBox15.TabStop = false;
            //
            // pictureBox16
            //
            this.pictureBox16.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox16.Image")));
            this.pictureBox16.Location = new System.Drawing.Point(508, 123);
            this.pictureBox16.Name = "pictureBox16";
            this.pictureBox16.Size = new System.Drawing.Size(184, 24);
            this.pictureBox16.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox16.TabIndex = 36;
            this.pictureBox16.TabStop = false;
            //
            // pictureBox14
            //
            this.pictureBox14.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox14.Image")));
            this.pictureBox14.Location = new System.Drawing.Point(347, 72);
            this.pictureBox14.Name = "pictureBox14";
            this.pictureBox14.Size = new System.Drawing.Size(144, 136);
            this.pictureBox14.TabIndex = 35;
            this.pictureBox14.TabStop = false;
            //
            // pictureBox13
            //
            this.pictureBox13.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox13.Image")));
            this.pictureBox13.Location = new System.Drawing.Point(729, 64);
            this.pictureBox13.Name = "pictureBox13";
            this.pictureBox13.Size = new System.Drawing.Size(247, 250);
            this.pictureBox13.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox13.TabIndex = 34;
            this.pictureBox13.TabStop = false;
            //
            // EnhancedLabelB_A
```

```
            //
            this.EnhancedLabelB_A.Location = new System.Drawing.Point(310,
300);
            this.EnhancedLabelB_A.Name = "EnhancedLabelB_A";
            this.EnhancedLabelB_A.Size = new System.Drawing.Size(378, 16);
            this.EnhancedLabelB_A.TabIndex = 33;
            //
            // EnhancedLabelA_B
            //
            this.EnhancedLabelA_B.Location = new System.Drawing.Point(307,
229);
            this.EnhancedLabelA_B.Name = "EnhancedLabelA_B";
            this.EnhancedLabelA_B.Size = new System.Drawing.Size(381, 16);
            this.EnhancedLabelA_B.TabIndex = 32;
            //
            // pictureBox12
            //
            this.pictureBox12.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox12.Image")));
            this.pictureBox12.Location = new System.Drawing.Point(304, 248);
            this.pictureBox12.Name = "pictureBox12";
            this.pictureBox12.Size = new System.Drawing.Size(384, 24);
            this.pictureBox12.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox12.TabIndex = 30;
            this.pictureBox12.TabStop = false;
            //
            // EnhancedReset
            //
            this.EnhancedReset.BackColor =
System.Drawing.SystemColors.Control;
            this.EnhancedReset.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EnhancedReset.Location = new System.Drawing.Point(347, 8);
            this.EnhancedReset.Name = "EnhancedReset";
            this.EnhancedReset.Size = new System.Drawing.Size(288, 24);
            this.EnhancedReset.TabIndex = 29;
            this.EnhancedReset.Text = "Reset";
            this.EnhancedReset.UseVisualStyleBackColor = false;
            this.EnhancedReset.Click += new
System.EventHandler(this.EnhancedReset_Click);
            //
            // EnhancedNextStep
            //
            this.EnhancedNextStep.BackColor =
System.Drawing.SystemColors.Control;
            this.EnhancedNextStep.Font = new System.Drawing.Font("Microsoft
Sans Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EnhancedNextStep.Location = new System.Drawing.Point(8, 8);
            this.EnhancedNextStep.Name = "EnhancedNextStep";
            this.EnhancedNextStep.Size = new System.Drawing.Size(288, 24);
            this.EnhancedNextStep.TabIndex = 24;
            this.EnhancedNextStep.Text = "NextStep";
            this.EnhancedNextStep.UseVisualStyleBackColor = false;
```

```
            this.EnhancedNextStep.Click += new
System.EventHandler(this.EnhancedNextStep_Click);
            //
            // pictureBox7
            //
            this.pictureBox7.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox7.Image")));
            this.pictureBox7.Location = new System.Drawing.Point(0, 64);
            this.pictureBox7.Name = "pictureBox7";
            this.pictureBox7.Size = new System.Drawing.Size(254, 256);
            this.pictureBox7.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox7.TabIndex = 20;
            this.pictureBox7.TabStop = false;
            //
            // EH_Tab
            //
            this.EH_Tab.BackColor = System.Drawing.SystemColors.Window;
            this.EH_Tab.Controls.Add(this.EHStatusCA);
            this.EH_Tab.Controls.Add(this.label19);
            this.EH_Tab.Controls.Add(this.label31);
            this.EH_Tab.Controls.Add(this.label30);
            this.EH_Tab.Controls.Add(this.EHStatusB);
            this.EH_Tab.Controls.Add(this.label27);
            this.EH_Tab.Controls.Add(this.EHHData);
            this.EH_Tab.Controls.Add(this.EHBData);
            this.EH_Tab.Controls.Add(this.EHAData);
            this.EH_Tab.Controls.Add(this.label28);
            this.EH_Tab.Controls.Add(this.label29);
            this.EH_Tab.Controls.Add(this.label15);
            this.EH_Tab.Controls.Add(this.EHLabelB_CA);
            this.EH_Tab.Controls.Add(this.EHLabelCA_B);
            this.EH_Tab.Controls.Add(this.pictureBox19);
            this.EH_Tab.Controls.Add(this.pictureBox20);
            this.EH_Tab.Controls.Add(this.pictureBox21);
            this.EH_Tab.Controls.Add(this.pictureBox22);
            this.EH_Tab.Controls.Add(this.EHLabelB_A);
            this.EH_Tab.Controls.Add(this.EHLabelA_B);
            this.EH_Tab.Controls.Add(this.pictureBox23);
            this.EH_Tab.Controls.Add(this.pictureBox24);
            this.EH_Tab.Controls.Add(this.pictureBox18);
            this.EH_Tab.Controls.Add(this.EHStatusA);
            this.EH_Tab.Controls.Add(this.label11);
            this.EH_Tab.Controls.Add(this.EHReset);
            this.EH_Tab.Controls.Add(this.EHNextStep);
            this.EH_Tab.Location = new System.Drawing.Point(4, 22);
            this.EH_Tab.Name = "EH_Tab";
            this.EH_Tab.Size = new System.Drawing.Size(1112, 782);
            this.EH_Tab.TabIndex = 3;
            this.EH_Tab.Text = "Enhanced - Hacked A";
            //
            // EHStatusCA
            //
            this.EHStatusCA.ForeColor = System.Drawing.Color.Red;
            this.EHStatusCA.Location = new System.Drawing.Point(393, 56);
            this.EHStatusCA.Name = "EHStatusCA";
            this.EHStatusCA.Size = new System.Drawing.Size(303, 16);
```

```
            this.EHStatusCA.TabIndex = 63;
            this.EHStatusCA.Text = "__";
            //
            // label19
            //
            this.label19.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label19.ForeColor = System.Drawing.Color.Red;
            this.label19.Location = new System.Drawing.Point(414, 96);
            this.label19.Name = "label19";
            this.label19.Size = new System.Drawing.Size(35, 22);
            this.label19.TabIndex = 74;
            this.label19.Text = "CA";
            //
            // label31
            //
            this.label31.BackColor = System.Drawing.SystemColors.Window;
            this.label31.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label31.ForeColor = System.Drawing.Color.Red;
            this.label31.Location = new System.Drawing.Point(1000, 344);
            this.label31.Name = "label31";
            this.label31.Size = new System.Drawing.Size(72, 24);
            this.label31.TabIndex = 73;
            this.label31.Text = "(Hacker)";
            //
            // label30
            //
            this.label30.Font = new System.Drawing.Font("Microsoft Sans
Serif", 10F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label30.ForeColor = System.Drawing.Color.Firebrick;
            this.label30.Location = new System.Drawing.Point(371, 393);
            this.label30.Name = "label30";
            this.label30.Size = new System.Drawing.Size(222, 24);
            this.label30.TabIndex = 72;
            this.label30.Text = "Known Information from Hack";
            //
            // EHStatusB
            //
            this.EHStatusB.Font = new System.Drawing.Font("Microsoft Sans
Serif", 7.5F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EHStatusB.Location = new System.Drawing.Point(726, 327);
            this.EHStatusB.Name = "EHStatusB";
            this.EHStatusB.Size = new System.Drawing.Size(256, 16);
            this.EHStatusB.TabIndex = 71;
            this.EHStatusB.Text = "_";
            //
            // label27
            //
            this.label27.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label27.Location = new System.Drawing.Point(679, 327);
            this.label27.Name = "label27";
```

```
            this.label27.Size = new System.Drawing.Size(56, 16);
            this.label27.TabIndex = 70;
            this.label27.Text = "B Status:";
            //
            // EHHData
            //
            this.EHHData.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(255)))),
((int)(((byte)(192)))));
            this.EHHData.Location = new System.Drawing.Point(301, 420);
            this.EHHData.Multiline = true;
            this.EHHData.Name = "EHHData";
            this.EHHData.RightToLeft = System.Windows.Forms.RightToLeft.No;
            this.EHHData.ScrollBars = System.Windows.Forms.ScrollBars.Both;
            this.EHHData.Size = new System.Drawing.Size(376, 230);
            this.EHHData.TabIndex = 69;
            //
            // EHBData
            //
            this.EHBData.BackColor = System.Drawing.Color.SkyBlue;
            this.EHBData.Location = new System.Drawing.Point(711, 367);
            this.EHBData.Multiline = true;
            this.EHBData.Name = "EHBData";
            this.EHBData.ScrollBars = System.Windows.Forms.ScrollBars.Both;
            this.EHBData.Size = new System.Drawing.Size(266, 283);
            this.EHBData.TabIndex = 68;
            //
            // EHAData
            //
            this.EHAData.BackColor = System.Drawing.Color.SkyBlue;
            this.EHAData.Location = new System.Drawing.Point(0, 367);
            this.EHAData.Multiline = true;
            this.EHAData.Name = "EHAData";
            this.EHAData.ScrollBars = System.Windows.Forms.ScrollBars.Both;
            this.EHAData.Size = new System.Drawing.Size(276, 283);
            this.EHAData.TabIndex = 67;
            //
            // label28
            //
            this.label28.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label28.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(128)))),
((int)(((byte)(0)))));
            this.label28.Location = new System.Drawing.Point(717, 347);
            this.label28.Name = "label28";
            this.label28.Size = new System.Drawing.Size(260, 16);
            this.label28.TabIndex = 66;
            this.label28.Text = "Bluetooth Device B = Hacker";
            //
            // label29
            //
            this.label29.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
```

```
            this.label29.ForeColor =
System.Drawing.SystemColors.ActiveCaption;
            this.label29.Location = new System.Drawing.Point(48, 344);
            this.label29.Name = "label29";
            this.label29.Size = new System.Drawing.Size(177, 16);
            this.label29.TabIndex = 65;
            this.label29.Text = "Bluetooth Device A";
            //
            // label15
            //
            this.label15.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label15.Location = new System.Drawing.Point(332, 56);
            this.label15.Name = "label15";
            this.label15.Size = new System.Drawing.Size(64, 16);
            this.label15.TabIndex = 64;
            this.label15.Text = "CA Status:";
            //
            // EHLabelB_CA
            //
            this.EHLabelB_CA.Location = new System.Drawing.Point(480, 166);
            this.EHLabelB_CA.Name = "EHLabelB_CA";
            this.EHLabelB_CA.Size = new System.Drawing.Size(192, 16);
            this.EHLabelB_CA.TabIndex = 62;
            //
            // EHLabelCA_B
            //
            this.EHLabelCA_B.Location = new System.Drawing.Point(493, 103);
            this.EHLabelCA_B.Name = "EHLabelCA_B";
            this.EHLabelCA_B.Size = new System.Drawing.Size(192, 16);
            this.EHLabelCA_B.TabIndex = 61;
            //
            // pictureBox19
            //
            this.pictureBox19.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox19.Image")));
            this.pictureBox19.Location = new System.Drawing.Point(490, 182);
            this.pictureBox19.Name = "pictureBox19";
            this.pictureBox19.Size = new System.Drawing.Size(184, 24);
            this.pictureBox19.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox19.TabIndex = 60;
            this.pictureBox19.TabStop = false;
            //
            // pictureBox20
            //
            this.pictureBox20.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox20.Image")));
            this.pictureBox20.Location = new System.Drawing.Point(490, 120);
            this.pictureBox20.Name = "pictureBox20";
            this.pictureBox20.Size = new System.Drawing.Size(184, 24);
            this.pictureBox20.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox20.TabIndex = 59;
            this.pictureBox20.TabStop = false;
            //
            // pictureBox21
```

```
            //
            this.pictureBox21.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox21.Image")));
            this.pictureBox21.Location = new System.Drawing.Point(337, 75);
            this.pictureBox21.Name = "pictureBox21";
            this.pictureBox21.Size = new System.Drawing.Size(144, 136);
            this.pictureBox21.TabIndex = 58;
            this.pictureBox21.TabStop = false;
            //
            // pictureBox22
            //
            this.pictureBox22.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox22.Image")));
            this.pictureBox22.Location = new System.Drawing.Point(711, 72);
            this.pictureBox22.Name = "pictureBox22";
            this.pictureBox22.Size = new System.Drawing.Size(266, 245);
            this.pictureBox22.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox22.TabIndex = 57;
            this.pictureBox22.TabStop = false;
            //
            // EHLabelB_A
            //
            this.EHLabelB_A.Location = new System.Drawing.Point(296, 301);
            this.EHLabelB_A.Name = "EHLabelB_A";
            this.EHLabelB_A.Size = new System.Drawing.Size(378, 16);
            this.EHLabelB_A.TabIndex = 56;
            //
            // EHLabelA_B
            //
            this.EHLabelA_B.Location = new System.Drawing.Point(307, 232);
            this.EHLabelA_B.Name = "EHLabelA_B";
            this.EHLabelA_B.Size = new System.Drawing.Size(370, 16);
            this.EHLabelA_B.TabIndex = 55;
            //
            // pictureBox23
            //
            this.pictureBox23.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox23.Image")));
            this.pictureBox23.Location = new System.Drawing.Point(304, 320);
            this.pictureBox23.Name = "pictureBox23";
            this.pictureBox23.Size = new System.Drawing.Size(370, 24);
            this.pictureBox23.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox23.TabIndex = 54;
            this.pictureBox23.TabStop = false;
            //
            // pictureBox24
            //
            this.pictureBox24.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox24.Image")));
            this.pictureBox24.Location = new System.Drawing.Point(304, 252);
            this.pictureBox24.Name = "pictureBox24";
            this.pictureBox24.Size = new System.Drawing.Size(370, 24);
            this.pictureBox24.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox24.TabIndex = 53;
```

```
            this.pictureBox24.TabStop = false;
            //
            // pictureBox18
            //
            this.pictureBox18.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox18.Image")));
            this.pictureBox18.Location = new System.Drawing.Point(0, 72);
            this.pictureBox18.Name = "pictureBox18";
            this.pictureBox18.Size = new System.Drawing.Size(257, 248);
            this.pictureBox18.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox18.TabIndex = 51;
            this.pictureBox18.TabStop = false;
            //
            // EHStatusA
            //
            this.EHStatusA.Font = new System.Drawing.Font("Microsoft Sans
Serif", 7.5F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EHStatusA.Location = new System.Drawing.Point(48, 328);
            this.EHStatusA.Name = "EHStatusA";
            this.EHStatusA.Size = new System.Drawing.Size(266, 16);
            this.EHStatusA.TabIndex = 50;
            this.EHStatusA.Text = "_";
            //
            // label11
            //
            this.label11.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label11.Location = new System.Drawing.Point(0, 328);
            this.label11.Name = "label11";
            this.label11.Size = new System.Drawing.Size(56, 16);
            this.label11.TabIndex = 49;
            this.label11.Text = "A Status:";
            //
            // EHReset
            //
            this.EHReset.BackColor = System.Drawing.SystemColors.Control;
            this.EHReset.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EHReset.Location = new System.Drawing.Point(337, 8);
            this.EHReset.Name = "EHReset";
            this.EHReset.Size = new System.Drawing.Size(288, 24);
            this.EHReset.TabIndex = 30;
            this.EHReset.Text = "Reset";
            this.EHReset.UseVisualStyleBackColor = false;
            this.EHReset.Click += new
System.EventHandler(this.EHReset_Click);
            //
            // EHNextStep
            //
            this.EHNextStep.BackColor = System.Drawing.SystemColors.Control;
            this.EHNextStep.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EHNextStep.Location = new System.Drawing.Point(8, 8);
```

```
            this.EHNextStep.Name = "EHNextStep";
            this.EHNextStep.Size = new System.Drawing.Size(288, 24);
            this.EHNextStep.TabIndex = 25;
            this.EHNextStep.Text = "NextStep";
            this.EHNextStep.UseVisualStyleBackColor = false;
            this.EHNextStep.Click += new
System.EventHandler(this.EHNextStep_Click);
            //
            // MainMenu
            //
            this.MainMenu.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.FileMenu});
            //
            // FileMenu
            //
            this.FileMenu.Index = 0;
            this.FileMenu.MenuItems.AddRange(new
System.Windows.Forms.MenuItem[] {
            this.ExitMenu});
            this.FileMenu.Text = "&File";
            //
            // ExitMenu
            //
            this.ExitMenu.Index = 0;
            this.ExitMenu.Shortcut = System.Windows.Forms.Shortcut.AltF4;
            this.ExitMenu.Text = "&Exit";
            this.ExitMenu.Click += new
System.EventHandler(this.ExitMenu_Click);
            //
            // EHB_Tab
            //
            this.EHB_Tab.BackColor = System.Drawing.SystemColors.Window;
            this.EHB_Tab.Controls.Add(this.label23);
            this.EHB_Tab.Controls.Add(this.EHBHData);
            this.EHB_Tab.Controls.Add(this.EHBStatusB);
            this.EHB_Tab.Controls.Add(this.EHBBData);
            this.EHB_Tab.Controls.Add(this.label38);
            this.EHB_Tab.Controls.Add(this.pictureBox31);
            this.EHB_Tab.Controls.Add(this.EHBStatusCA);
            this.EHB_Tab.Controls.Add(this.label25);
            this.EHB_Tab.Controls.Add(this.label26);
            this.EHB_Tab.Controls.Add(this.label32);
            this.EHB_Tab.Controls.Add(this.EHBLabelB_CA);
            this.EHB_Tab.Controls.Add(this.EHBLabelCA_B);
            this.EHB_Tab.Controls.Add(this.pictureBox26);
            this.EHB_Tab.Controls.Add(this.pictureBox27);
            this.EHB_Tab.Controls.Add(this.pictureBox28);
            this.EHB_Tab.Controls.Add(this.EHBLabelB_A);
            this.EHB_Tab.Controls.Add(this.EHBLabelA_B);
            this.EHB_Tab.Controls.Add(this.pictureBox29);
            this.EHB_Tab.Controls.Add(this.pictureBox30);
            this.EHB_Tab.Controls.Add(this.EHBAData);
            this.EHB_Tab.Controls.Add(this.label21);
            this.EHB_Tab.Controls.Add(this.EHBStatusA);
            this.EHB_Tab.Controls.Add(this.label24);
            this.EHB_Tab.Controls.Add(this.pictureBox25);
```

```
            this.EHB_Tab.Controls.Add(this.EHBReset);
            this.EHB_Tab.Controls.Add(this.EHBNextStep);
            this.EHB_Tab.Location = new System.Drawing.Point(4, 22);
            this.EHB_Tab.Name = "EHB_Tab";
            this.EHB_Tab.Padding = new System.Windows.Forms.Padding(3);
            this.EHB_Tab.Size = new System.Drawing.Size(1112, 782);
            this.EHB_Tab.TabIndex = 4;
            this.EHB_Tab.Text = "Enhanced - Hacked B";
            //
            // EHBNextStep
            //
            this.EHBNextStep.BackColor = System.Drawing.SystemColors.Control;
            this.EHBNextStep.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EHBNextStep.Location = new System.Drawing.Point(8, 6);
            this.EHBNextStep.Name = "EHBNextStep";
            this.EHBNextStep.Size = new System.Drawing.Size(288, 24);
            this.EHBNextStep.TabIndex = 26;
            this.EHBNextStep.Text = "NextStep";
            this.EHBNextStep.UseVisualStyleBackColor = false;
            this.EHBNextStep.Click += new
System.EventHandler(this.EHBNextStep_Click);
            //
            // EHBReset
            //
            this.EHBReset.BackColor = System.Drawing.SystemColors.Control;
            this.EHBReset.Font = new System.Drawing.Font("Microsoft Sans
Serif", 8.25F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EHBReset.Location = new System.Drawing.Point(342, 6);
            this.EHBReset.Name = "EHBReset";
            this.EHBReset.Size = new System.Drawing.Size(288, 24);
            this.EHBReset.TabIndex = 31;
            this.EHBReset.Text = "Reset";
            this.EHBReset.UseVisualStyleBackColor = false;
            this.EHBReset.Click += new
System.EventHandler(this.EHBReset_Click);
            //
            // pictureBox25
            //
            this.pictureBox25.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox25.Image")));
            this.pictureBox25.Location = new System.Drawing.Point(8, 50);
            this.pictureBox25.Name = "pictureBox25";
            this.pictureBox25.Size = new System.Drawing.Size(257, 248);
            this.pictureBox25.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox25.TabIndex = 52;
            this.pictureBox25.TabStop = false;
            //
            // label21
            //
            this.label21.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
```

```
            this.label21.ForeColor =
System.Drawing.SystemColors.ActiveCaption;
            this.label21.Location = new System.Drawing.Point(51, 317);
            this.label21.Name = "label21";
            this.label21.Size = new System.Drawing.Size(177, 16);
            this.label21.TabIndex = 68;
            this.label21.Text = "Bluetooth Device A";
            //
            // EHBStatusA
            //
            this.EHBStatusA.Font = new System.Drawing.Font("Microsoft Sans
Serif", 7.5F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EHBStatusA.Location = new System.Drawing.Point(51, 301);
            this.EHBStatusA.Name = "EHBStatusA";
            this.EHBStatusA.Size = new System.Drawing.Size(266, 16);
            this.EHBStatusA.TabIndex = 67;
            this.EHBStatusA.Text = "_";
            //
            // label24
            //
            this.label24.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label24.Location = new System.Drawing.Point(3, 301);
            this.label24.Name = "label24";
            this.label24.Size = new System.Drawing.Size(56, 16);
            this.label24.TabIndex = 66;
            this.label24.Text = "A Status:";
            //
            // EHBAData
            //
            this.EHBAData.BackColor = System.Drawing.Color.SkyBlue;
            this.EHBAData.Location = new System.Drawing.Point(6, 336);
            this.EHBAData.Multiline = true;
            this.EHBAData.Name = "EHBAData";
            this.EHBAData.ScrollBars = System.Windows.Forms.ScrollBars.Both;
            this.EHBAData.Size = new System.Drawing.Size(276, 283);
            this.EHBAData.TabIndex = 69;
            //
            // EHBStatusCA
            //
            this.EHBStatusCA.ForeColor = System.Drawing.Color.Red;
            this.EHBStatusCA.Location = new System.Drawing.Point(386, 31);
            this.EHBStatusCA.Name = "EHBStatusCA";
            this.EHBStatusCA.Size = new System.Drawing.Size(303, 16);
            this.EHBStatusCA.TabIndex = 85;
            this.EHBStatusCA.Text = "__";
            //
            // label25
            //
            this.label25.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label25.ForeColor = System.Drawing.Color.Red;
            this.label25.Location = new System.Drawing.Point(407, 71);
            this.label25.Name = "label25";
            this.label25.Size = new System.Drawing.Size(35, 22);
```

```
            this.label25.TabIndex = 88;
            this.label25.Text = "CA";
            //
            // label26
            //
            this.label26.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label26.Location = new System.Drawing.Point(672, 302);
            this.label26.Name = "label26";
            this.label26.Size = new System.Drawing.Size(56, 16);
            this.label26.TabIndex = 87;
            this.label26.Text = "B Status:";
            //
            // label32
            //
            this.label32.ForeColor =
System.Drawing.SystemColors.MenuHighlight;
            this.label32.Location = new System.Drawing.Point(325, 31);
            this.label32.Name = "label32";
            this.label32.Size = new System.Drawing.Size(64, 16);
            this.label32.TabIndex = 86;
            this.label32.Text = "CA Status:";
            //
            // EHBLabelB_CA
            //
            this.EHBLabelB_CA.Location = new System.Drawing.Point(473, 141);
            this.EHBLabelB_CA.Name = "EHBLabelB_CA";
            this.EHBLabelB_CA.Size = new System.Drawing.Size(192, 16);
            this.EHBLabelB_CA.TabIndex = 84;
            //
            // EHBLabelCA_B
            //
            this.EHBLabelCA_B.Location = new System.Drawing.Point(486, 78);
            this.EHBLabelCA_B.Name = "EHBLabelCA_B";
            this.EHBLabelCA_B.Size = new System.Drawing.Size(192, 16);
            this.EHBLabelCA_B.TabIndex = 83;
            //
            // pictureBox26
            //
            this.pictureBox26.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox26.Image")));
            this.pictureBox26.Location = new System.Drawing.Point(483, 157);
            this.pictureBox26.Name = "pictureBox26";
            this.pictureBox26.Size = new System.Drawing.Size(184, 24);
            this.pictureBox26.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox26.TabIndex = 82;
            this.pictureBox26.TabStop = false;
            //
            // pictureBox27
            //
            this.pictureBox27.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox27.Image")));
            this.pictureBox27.Location = new System.Drawing.Point(483, 95);
            this.pictureBox27.Name = "pictureBox27";
            this.pictureBox27.Size = new System.Drawing.Size(184, 24);
```

```
            this.pictureBox27.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox27.TabIndex = 81;
            this.pictureBox27.TabStop = false;
            //
            // pictureBox28
            //
            this.pictureBox28.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox28.Image")));
            this.pictureBox28.Location = new System.Drawing.Point(330, 50);
            this.pictureBox28.Name = "pictureBox28";
            this.pictureBox28.Size = new System.Drawing.Size(144, 136);
            this.pictureBox28.TabIndex = 80;
            this.pictureBox28.TabStop = false;
            //
            // EHBLabelB_A
            //
            this.EHBLabelB_A.Location = new System.Drawing.Point(289, 276);
            this.EHBLabelB_A.Name = "EHBLabelB_A";
            this.EHBLabelB_A.Size = new System.Drawing.Size(378, 16);
            this.EHBLabelB_A.TabIndex = 79;
            //
            // EHBLabelA_B
            //
            this.EHBLabelA_B.Location = new System.Drawing.Point(300, 207);
            this.EHBLabelA_B.Name = "EHBLabelA_B";
            this.EHBLabelA_B.Size = new System.Drawing.Size(370, 16);
            this.EHBLabelA_B.TabIndex = 78;
            //
            // pictureBox29
            //
            this.pictureBox29.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox29.Image")));
            this.pictureBox29.Location = new System.Drawing.Point(297, 295);
            this.pictureBox29.Name = "pictureBox29";
            this.pictureBox29.Size = new System.Drawing.Size(370, 24);
            this.pictureBox29.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox29.TabIndex = 77;
            this.pictureBox29.TabStop = false;
            //
            // pictureBox30
            //
            this.pictureBox30.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox30.Image")));
            this.pictureBox30.Location = new System.Drawing.Point(297, 227);
            this.pictureBox30.Name = "pictureBox30";
            this.pictureBox30.Size = new System.Drawing.Size(370, 24);
            this.pictureBox30.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox30.TabIndex = 76;
            this.pictureBox30.TabStop = false;
            //
            // EHBStatusB
            //
```

```
            this.EHBStatusB.Font = new System.Drawing.Font("Microsoft Sans
Serif", 7.5F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.EHBStatusB.Location = new System.Drawing.Point(722, 300);
            this.EHBStatusB.Name = "EHBStatusB";
            this.EHBStatusB.Size = new System.Drawing.Size(256, 16);
            this.EHBStatusB.TabIndex = 92;
            this.EHBStatusB.Text = "_";
            //
            // EHBBData
            //
            this.EHBBData.BackColor = System.Drawing.Color.SkyBlue;
            this.EHBBData.Location = new System.Drawing.Point(709, 335);
            this.EHBBData.Multiline = true;
            this.EHBBData.Name = "EHBBData";
            this.EHBBData.ScrollBars = System.Windows.Forms.ScrollBars.Both;
            this.EHBBData.Size = new System.Drawing.Size(266, 283);
            this.EHBBData.TabIndex = 91;
            //
            // label38
            //
            this.label38.Font = new System.Drawing.Font("Microsoft Sans
Serif", 12F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label38.ForeColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(128)))),
((int)(((byte)(0)))));
            this.label38.Location = new System.Drawing.Point(715, 315);
            this.label38.Name = "label38";
            this.label38.Size = new System.Drawing.Size(260, 16);
            this.label38.TabIndex = 90;
            this.label38.Text = "Bluetooth Device B = Hacker";
            //
            // pictureBox31
            //
            this.pictureBox31.Image =
((System.Drawing.Image)(resources.GetObject("pictureBox31.Image")));
            this.pictureBox31.Location = new System.Drawing.Point(709, 47);
            this.pictureBox31.Name = "pictureBox31";
            this.pictureBox31.Size = new System.Drawing.Size(266, 245);
            this.pictureBox31.SizeMode =
System.Windows.Forms.PictureBoxSizeMode.StretchImage;
            this.pictureBox31.TabIndex = 89;
            this.pictureBox31.TabStop = false;
            //
            // label23
            //
            this.label23.Font = new System.Drawing.Font("Microsoft Sans
Serif", 10F, System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.label23.ForeColor = System.Drawing.Color.Firebrick;
            this.label23.Location = new System.Drawing.Point(383, 362);
            this.label23.Name = "label23";
            this.label23.Size = new System.Drawing.Size(222, 24);
            this.label23.TabIndex = 94;
            this.label23.Text = "Known Information from Hack";
            //
```

```
            // EHBHData
            //
            this.EHBHData.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))), ((int)(((byte)(255)))),
((int)(((byte)(192)))));
            this.EHBHData.Location = new System.Drawing.Point(313, 389);
            this.EHBHData.Multiline = true;
            this.EHBHData.Name = "EHBHData";
            this.EHBHData.RightToLeft = System.Windows.Forms.RightToLeft.No;
            this.EHBHData.ScrollBars = System.Windows.Forms.ScrollBars.Both;
            this.EHBHData.Size = new System.Drawing.Size(376, 230);
            this.EHBHData.TabIndex = 93;
            //
            // Form1
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.ClientSize = new System.Drawing.Size(991, 684);
            this.Controls.Add(this.tabControl1);
            this.Menu = this.MainMenu;
            this.Name = "Form1";
            this.Text = "Form1";
            this.Load += new System.EventHandler(this.Form1_Load);
            this.tabControl1.ResumeLayout(false);
            this.OriginalTab.ResumeLayout(false);
            this.OriginalTab.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox5)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox4)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox3)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox2)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndInit();
            this.OriginalHackedTab.ResumeLayout(false);
            this.OriginalHackedTab.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox10)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox8)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox9)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox6)).EndInit();
            this.EnhancedTab.ResumeLayout(false);
            this.EnhancedTab.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox11)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox17)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox15)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox16)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox14)).EndInit();
```

```
((System.ComponentModel.ISupportInitialize)(this.pictureBox13)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox12)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox7)).EndInit();
            this.EH_Tab.ResumeLayout(false);
            this.EH_Tab.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox19)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox20)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox21)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox22)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox23)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox24)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox18)).EndInit();
            this.EHB_Tab.ResumeLayout(false);
            this.EHB_Tab.PerformLayout();

((System.ComponentModel.ISupportInitialize)(this.pictureBox25)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox26)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox27)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox28)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox29)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox30)).EndInit();

((System.ComponentModel.ISupportInitialize)(this.pictureBox31)).EndInit();
            this.ResumeLayout(false);

            }
            #endregion

            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            [STAThread]
            static void Main()
            {
                  Application.Run(new Form1());
            }

            private void ExitMenu_Click(object sender, System.EventArgs e)
            {
                  this.Close();
            }
```

```
            private void OriginalNextStep_Click(object sender,
System.EventArgs e)
            {
                  OriginalState++;
                  if(OriginalStateA.Count>OriginalState)
                  {
                        OriginalStatusA.Text =
((State)OriginalStateA[OriginalState]).Text;
                        OriginalStatusB.Text =
((State)OriginalStateB[OriginalState]).Text;

      if(((State)OriginalStateA[OriginalState]).SendText!="")
                        {

      OriginalAData.Text+=System.Environment.NewLine+((State)OriginalStateA[O
riginalState]).SendText;

      OriginalHData.Text+=System.Environment.NewLine+"From A:
"+((State)OriginalStateA[OriginalState]).SendText;

      OriginalLabelA_B.Text=((State)OriginalStateA[OriginalState]).SendText;
                        }
                        else
                              OriginalLabelA_B.Text = "";

      if(((State)OriginalStateB[OriginalState]).SendText!="")
                        {

      OriginalBData.Text+=System.Environment.NewLine+((State)OriginalStateB[O
riginalState]).SendText;

      OriginalHData.Text+=System.Environment.NewLine+"From B:
"+((State)OriginalStateB[OriginalState]).SendText;
                              OriginalLabelB_A.Text =
((State)OriginalStateB[OriginalState]).SendText;
                              OriginalLabelB_A.TextAlign =
System.Drawing.ContentAlignment.TopRight;
                        }
                        else
                              OriginalLabelB_A.Text = "";
                        if(OriginalState==7)
                        {
                              OriginalHData.Text+=
System.Environment.NewLine+"-------------- CAN NOW CALCULATE Kab ------------
------------";
                              OriginalHData.Text+=
System.Environment.NewLine+"Kab ->From IN_RANDa, BD_ADDRb, and PIN
information";
                              OriginalHData.Text+=
System.Environment.NewLine+"----------------------------------------------------
-----------------------------------";
                        }
                  }
                  else
                        System.Windows.Forms.MessageBox.Show("Pairing and
authentication process completed.");
            }
```

```
            private void OriginalReset_Click(object sender, System.EventArgs
e)
            {
                  OriginalState =-1;
                  OriginalLabelA_B.Text = "";
                  OriginalLabelB_A.Text = "";
                  OriginalStatusA.Text = "";
                  OriginalStatusB.Text = "";
                  OriginalAData.Text = "";
                  OriginalHData.Text = "";
                  OriginalBData.Text = "";
            }

            private void HackedNextStep_Click(object sender, System.EventArgs
e)
            {
                  HackedState++;
                  if(HackedStateA.Count>HackedState)
                  {
                        HackedStatusA.Text =
((State)HackedStateA[HackedState]).Text;
                        HackedStatusB.Text =
((State)HackedStateB[HackedState]).Text;
                        if(((State)HackedStateA[HackedState]).SendText!="")
                        {

      HackedAData.Text+=System.Environment.NewLine+((State)HackedStateA[Hacke
dState]).SendText;

      HackedLabelA_B.Text=((State)HackedStateA[HackedState]).SendText;
                        }
                        else
                              HackedLabelA_B.Text = "";
                        if(((State)HackedStateB[HackedState]).SendText!="")
                        {

      HackedBData.Text+=System.Environment.NewLine+((State)HackedStateB[Hacke
dState]).SendText;
                              HackedLabelB_A.Text =
((State)HackedStateB[HackedState]).SendText;
                              HackedLabelB_A.TextAlign =
System.Drawing.ContentAlignment.TopRight;
                        }
                        else
                              HackedLabelB_A.Text = "";
                  }
                  else
                        System.Windows.Forms.MessageBox.Show("Pairing and
authentication process completed.");
            }

            private void EnhancedNextStep_Click(object sender,
System.EventArgs e)
            {
                  EnhancedState++;
                  if(EnhancedStateA.Count>EnhancedState)
```

```
                  {
                        EnhancedStatusA.Text =
((State)EnhancedStateA[EnhancedState]).Text;
                        EnhancedStatusB.Text =
((State)EnhancedStateB[EnhancedState]).Text;
                        EnhancedStatusCA.Text =
((State)EnhancedStateCA[EnhancedState]).Text;

      if(((State)EnhancedStateA[EnhancedState]).SendText!="")
                        {

      EnhancedAData.Text+=System.Environment.NewLine+((State)EnhancedStateA[E
nhancedState]).SendText;

      EnhancedLabelA_B.Text=((State)EnhancedStateA[EnhancedState]).SendText;

      EnhancedHData.Text+=System.Environment.NewLine+"From A:
"+((State)EnhancedStateA[EnhancedState]).SendText;
                        }
                        else
                              EnhancedLabelA_B.Text = "";

      if(((State)EnhancedStateB[EnhancedState]).SendText!="")
                        {

      EnhancedBData.Text+=System.Environment.NewLine+((State)EnhancedStateB[E
nhancedState]).SendText;
                              EnhancedLabelB_A.Text =
((State)EnhancedStateB[EnhancedState]).SendText;
                              EnhancedLabelB_A.TextAlign =
System.Drawing.ContentAlignment.TopRight;

      EnhancedHData.Text+=System.Environment.NewLine+"From B:
"+((State)EnhancedStateB[EnhancedState]).SendText;
                        }
                        else
                              EnhancedLabelB_A.Text = "";

      if(((State)EnhancedStateCA[EnhancedState]).SendText!="")
                        {
                              EnhancedLabelCA_B.Text =
((State)EnhancedStateCA[EnhancedState]).SendText;

      //EnhancedHData.Text+=System.Environment.NewLine+"From CA:
"+((State)EnhancedStateCA[EnhancedState]).SendText;
                        }
                        else
                              EnhancedLabelCA_B.Text = "";
                        if(((State)EnhancedStateB[EnhancedState]).CAText!="")
                        {
                              EnhancedLabelB_CA.Text =
((State)EnhancedStateB[EnhancedState]).CAText;
                              EnhancedLabelB_CA.TextAlign =
System.Drawing.ContentAlignment.TopRight;

      //EnhancedHData.Text+=System.Environment.NewLine+"From B:
"+((State)EnhancedStateB[EnhancedState]).CAText;
```

```
                        }
                        else
                              EnhancedLabelB_CA.Text = "";
                  }
                  else
                        System.Windows.Forms.MessageBox.Show("Pairing and
authentication process completed.");
            }

            private void HackedReset_Click(object sender, System.EventArgs e)
            {
                  HackedState =-1;
                  HackedLabelA_B.Text = "";
                  HackedLabelB_A.Text = "";
                  HackedStatusA.Text = "";
                  HackedStatusB.Text = "";
                  HackedAData.Text = "";
                  HackedHData.Text = "";
                  HackedBData.Text = "";
            }

            private void EnhancedReset_Click(object sender, System.EventArgs
e)
            {
                  EnhancedState =-1;
                  EnhancedLabelA_B.Text = "";
                  EnhancedLabelB_A.Text = "";
                  EnhancedStatusA.Text = "";
                  EnhancedStatusB.Text = "";
                  EnhancedAData.Text = "";
                  EnhancedHData.Text = "";
                  EnhancedBData.Text = "";
            }




            private void EHReset_Click(object sender, System.EventArgs e)
            {
                  EHState =-1;
                  EHLabelA_B.Text = "";
                  EHLabelB_A.Text = "";
                  EHStatusA.Text = "";
                  EHStatusB.Text = "";
                  EHAData.Text = "";
                  EHBData.Text = "";
            }

            private void EHNextStep_Click(object sender, System.EventArgs e)
            {
                  EHState++;
                  if(EHStateA.Count>EHState)
                  {
                        EHStatusA.Text = ((State)EHStateA[EHState]).Text;
                        EHStatusB.Text = ((State)EHStateB[EHState]).Text;
                        EHStatusCA.Text = ((State)EHStateCA[EHState]).Text;
                        if(((State)EHStateA[EHState]).SendText!="")
```

```
                        {

      EHAData.Text+=System.Environment.NewLine+((State)EHStateA[EHState]).Sen
dText;

      EHLabelA_B.Text=((State)EHStateA[EHState]).SendText;

                        }
                        else
                              EHLabelA_B.Text = "";
                        if(((State)EHStateB[EHState]).SendText!="")
                        {

      EHBData.Text+=System.Environment.NewLine+((State)EHStateB[EHState]).Sen
dText;
                              EHLabelB_A.Text =
((State)EHStateB[EHState]).SendText;
                              EHLabelB_A.TextAlign =
System.Drawing.ContentAlignment.TopRight;

                        }
                        else
                              EHLabelB_A.Text = "";
                        if(((State)EHStateCA[EHState]).SendText!="")
                        {
                              EHLabelCA_B.Text =
((State)EHStateCA[EHState]).SendText;

                        }
                        else
                              EHLabelCA_B.Text = "";
                        if(((State)EHStateB[EHState]).CAText!="")
                        {
                              EHLabelB_CA.Text =
((State)EHStateB[EHState]).CAText;
                              EHLabelB_CA.TextAlign =
System.Drawing.ContentAlignment.TopRight;

                        }
                        else
                              EHLabelB_CA.Text = "";
                  }
                  else
                        System.Windows.Forms.MessageBox.Show("Authentication
Failed. Communication Failed. Start Over Again.");
            }

        private void Form1_Load(object sender, EventArgs e)
        {

        }

        private void EHBReset_Click(object sender, EventArgs e)
        {

                EHBState = -1;
                EHBLabelA_B.Text = "";
```

```
                EHBLabelB_A.Text = "";
                EHBStatusA.Text = "";
                EHBStatusB.Text = "";
                EHBAData.Text = "";
                EHBBData.Text = "";

        }

        private void EHBNextStep_Click(object sender, EventArgs e)
        {
                  EHBState++;
                  if(EHBStateA.Count>EHBState)
                  {
                        EHBStatusA.Text = ((State)EHBStateA[EHBState]).Text;
                        EHBStatusB.Text = ((State)EHBStateB[EHBState]).Text;
                        EHBStatusCA.Text =
((State)EHBStateCA[EHBState]).Text;
                        if(((State)EHBStateA[EHBState]).SendText!="")
                        {

      EHBAData.Text+=System.Environment.NewLine+((State)EHBStateA[EHBState]).
SendText;

      EHBLabelA_B.Text=((State)EHBStateA[EHBState]).SendText;

                        }
                        else
                              EHBLabelA_B.Text = "";
                        if(((State)EHBStateB[EHBState]).SendText!="")
                        {

      EHBBData.Text+=System.Environment.NewLine+((State)EHBStateB[EHBState]).
SendText;
                              EHBLabelB_A.Text =
((State)EHBStateB[EHBState]).SendText;
                              EHBLabelB_A.TextAlign =
System.Drawing.ContentAlignment.TopRight;

                        }
                        else
                              EHBLabelB_A.Text = "";
                        if(((State)EHBStateCA[EHBState]).SendText!="")
                        {
                              EHBLabelCA_B.Text =
((State)EHBStateCA[EHBState]).SendText;

                        }
                        else
                              EHBLabelCA_B.Text = "";
                        if(((State)EHBStateB[EHBState]).CAText!="")
                        {
                              EHBLabelB_CA.Text =
((State)EHBStateB[EHBState]).CAText;
                              EHBLabelB_CA.TextAlign =
System.Drawing.ContentAlignment.TopRight;

                        }
```

```
                        else
                              EHBLabelB_CA.Text = "";
                  }
                  else
                        System.Windows.Forms.MessageBox.Show("Authentication
failed. Communication Failed. Start Over Again.");
            }



      }
}
```

.....

File name: State.cs

```
using System;

namespace BluetoothThesis
{
      /// <summary>
      /// Summary description for State.
      /// </summary>
      public class State
      {
            public string Text;
            public string SendText;
            public string CAText;
            public State(string Text,string SendText)
            {
                  this.Text = Text;
                  this.SendText = SendText;
                  this.CAText = "";
            }
            public State(string Text,string SendText,string CAText)
            {
                  this.Text = Text;
                  this.SendText = SendText;
                  this.CAText = CAText;
            }
      }
}
```

Curriculum Vitae

Patricia Aidee Mendoza Sigala was born on October 03, 1979 in Chihuahua, Chihuahua, Mexico. As the second daughter of Martha A. Sigala Armendáriz and Miguel J. Mendoza Maldonado, Patricia graduated in June 1998 from the Preparatoria Del Chamizal EMS-2/3, at Ciudad Juarez, Chihuahua, and entered the University of Texas at El Paso in the spring of 1998 to pursue a degree in Electrical Engineering.

While pursuing her bachelor's degree, Patricia worked at the Department of Electrical and Computer Engineering from the University of Texas at El Paso as a Network System Administrator. Moreover, during summer 2003, she had the opportunity to work as a student intern updating test control graphics for the Eastman Kodak Company at Rochester, NY. Patricia obtained her bachelor's degree in Electrical Engineering in May 2004.

In September 2004, just after the completion of her bachelor's studies, Patricia had the opportunity to present and publish a project document named "A Web-based Vital Sign Telemonitor and Recorder for Telemedicine Applications" in the 26th IEEE Engineering and Biology Society (EMBS) Annual International Conference at San Francisco, CA.

In January 2005, she started a Master's degree in Computer Engineering at the University of Texas at El Paso. While pursuing her master's degree, she combined responsibilities by working as a Test Engineer at Delphi Automotive Systems at Cd. Juarez, Chihuahua, and also working as a Teaching Assistant at the Department of Electrical and Computer Engineering at the University of Texas at El Paso.

Permanent Address: Camino Valencia Oriente 1307-27
Jardines de Aragón Etapa VII
Cd. Juárez, Chih., Mex., 32472

This Thesis was typed by Patricia Aidee Mendoza.