ALEXANDRU IOAN CUZA UNIVERSITY OF IAŞI

FACULTY OF COMPUTER SCIENCE

MSC THESIS

Bluetooth Security

Cristian-Adrian Lucaci

July 2016

Scientific coordinator

Lecturer PhD Sorin Iftene

DECLARAŢIE PRIVIND ORIGINALITATEA ŞI RESPECTAREA DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de Disertație cu titlul “Bluetooth Security” este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referință precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referință precisă;
- codul sursă, imagini etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
- rezumarea ideilor altor autori precizează referință precisă la textul original.

Iași, 1 Iulie 2016

Absolvent *Cristian-Adrian Lucaci*

.....

DECLARAŢIE DE CONSIMŢĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de Disertație cu titlul “ Bluetooth Security”, codul sursă al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea Alexandru Ioan Cuza Iași să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursă, realizate de mine în cadrul prezentei lucrări de dizertație .

Iași, 1 Iulie 2016

Absolvent *Cristian-Adrian Lucaci*

.....

ACORD PRIVIND PROPRIETATEA DREPTULUI DE AUTOR

Facultatea de Informatică este de acord ca drepturile de autor asupra programele-calculator format executabil și sursă, să aparțină autorului prezentei lucrări, *Cristian-Adrian Lucaci*.

Încheierea acestui acord este necesară din următoarele motive:

- lucrarea constituie proprietate intelectuală al domnului Cristian-Adrian Lucaci, iar modificarea, copierea sau vânzarea acesteia fără acord constituie infracțiune și se pedepsește penal;

- lucrarea contine idei si concepte rezultate in urma analizei personale;

- lucrarea contine fragmente de text, tehnologii, algoritmi si concepte preluate in mod direct sau prin transcriere din specificatiile oficiale ale grupului Bluetooth Special Interest Group;

- lucrarea contine algoritmi, procese, conceptele si idei prezentate in format text sau grafic dezvoltate sau prelucrate de diferiti cercatatori sau grupuri de cercetatori din domeniu;

- lucrarea contine referinte directe pentru toate elementele preluate din opera altor autori, fiecare referinta descrie numele autorilor, titlul lucrarii precum dupa posibilitati adresa si data publicarii acestora pe internet.

Iași 1 Iulie 2016

Decan *Dr. Adrian Iftene* Absolvent *Cristian-Adrian Lucaci*

....

CONTENTS

1. INTRODUCTION

Information Security is the process of protecting information and information systems from unauthorized access, use, disclosure, disruption, modification or destruction in order to provide confidentiality, integrity and availability. Since the early days of communication, diplomats and commanders have understood the importance of correspondence confidentiality, therefore in 50 B.C Julius Caesar invented the Caesar Cipher in order to prevent sensitive information from being read. By the time of the First World War governments have developed multi-tier classification systems to manage the communication between various fronts, which in turn encouraged creation of code making sections like the "Government Code and Cipher School" in United Kingdom. In the early 21st century rapid advancements were made in telecommunications, computing hardware, software, and data encryption facilitating creating of academic disciplines of computer security within numerous professional organizations.

Information Security in wireless networks is a branch in computer security that prevents unauthorized access to internal network resources in order to capture and record traffic or perform disruptive acts. Identity theft or MAC spoofing occurs when an attacker is able to listen in on network traffic and identify the MAC address of a computer with network privileges. Man-in-the-middle attacks entice computers to connect to a transparent network peer where the traffic can be monitored. Denial-of-Service attacks bombard the network with bogus requests until the network is shut down, as result all the initial handshake codes are re-transmitted by all devices providing the opportunity for the malicious attacker to record these codes for cracking tools.

Global System for Mobile Communications is a popular wireless network indented for remote voice communication between cellular devices dispersed all over the globe.

Wireless Local Area Network is a home wireless network intended to distribute Internet access to a cluster of devices using radio waves.

Bluetooth is a short range wireless network mostly indented for data exchange between low powered devices, however due to its practicability was extended to support audio and voice.

Infrared Data Association is a point-and-shoot optical wireless communication system.

This thesis focuses on the study of Bluetooth Security, but first will introduce the basic principles behind wireless communication and their attributes compared to the legacy wired systems, then will continue with an overview of most common wireless technologies while trying to highlight some characteristics of Bluetooth Technology.

In the first chapter we will present the appliances of wireless systems around the world, like in voice communication, internet connection, data scanning and data transfer. Bluetooth is mostly focusing on exchanging large amounts of data, over a short period of time, across medium distances, between incompatible devices, while maintaining the quality and privacy of wired solutions. Nowadays Bluetooth devices count billions of users from all layers of society, entertainment, scientific, industry, medical and military, therefore opening serious security threats.

In the first chapter we will also take a brief look on the technical aspects of Bluetooth chips. We will summarized the Bluetooth generations, the ACL links used for data transfer versus SCO links for real-time voice communication, piconets and roles of devices, dBm power and class of Bluetooth transmitters, and ranges over different gradients of obstacles.

In the second chapter we will highlight the three most important security properties, authentication, authorization via different pairing protocols and encryption via SAFER+ and AES ciphering algorithms.

In the third chapter we will express the most common vulnerabilities found in wireless networks, but also those particularly to Bluetooth specification about weakness in PIN, vulnerability in the SSP association model selection protocol, deprecated encryption algorithms and users mistakes in configuring his device Bluetooth settings.

In the fourth chapter we will highlight some of the practical attacks against Bluetooth specification while providing some solutions or countermeasures, discuss different types of attacks of disclosure, distortion or disruption of data, some of the theoretical attacks revealed by security specialists, and the continuous MITM threat in wireless networks.

In the fifth chapter we will analyze a real life attack on the latest Bluetooth 4.0 devices, then we will highlight some vulnerabilities in pairing procedures that should be fixed.

In the Appendix is included a brief list of most used acronyms throughout the thesis.

2. BLUETOOTH TECHNOLOGY

Wireless communication systems, such as Bluetooth and others, are based on radio frequency waves to interconnect devices without a direct line of sight, thus communication becomes easier and more convenient than legacy wired networks, as result a rapid growth among common users and business was registered. However RF networks brought some downsides as well, such as due to the wireless broadcasting, eavesdropping and jamming is a lot easier than the wired ones.
Wireless networks fall in four major categories according to their desired purpose:

Global System for Mobile Communications (GSM) intended for cellular networks consist of a set of protocols supporting mobile device communication via voice, and later video, started in 1982 in Europe, currently in the 4th generation (4G).

Wireless Local Area Network (WLAN) intended for computer networks connection within a limited area, such as office, home or building in order to provide access to Internet.

Bluetooth Technology intended for data exchange over short distances between mobile low power and low cost devices forming Personal Area Networks (PANs). Was designed in 1994 by telecom vendor Ericsson as a replacement to RS-232 data cables and later maintained by Bluetooth Special Interest Group (SIG). The name „Bluetooth" was taken by its developer, Jim Kardach, from a book about the king *Harald Bluetooth* of Denmark from twenty century, and represents the anglicized *Haraldr Blátǫnn Gormsson* from Old Norse or *Harald Blåtand Gormsen* from Danish. The technology is under continuous development and evolution (e.g. 4th generation).

Infrared Data Association (IrDA) intended for wireless data transfer over the last meter using point-and-shoot principles, is a wireless optical communication protocol designed in 1993 by industry driven interest group with a physically secure data transfer, line of sight and low bit error rate providing very efficient scanning devices.

Bluetooth is a short-range wireless technology targeting handheld, battery-based personal devices with special needs and constraints that do not cost and consume too much power while maintaining a certain level of performance and adjacent channel rejection. Bluetooth is based on the master-slave concept due to the dynamic nature of the network constellation and lack of an in

place centralized infrastructure (e.g. cellular networks) where any node is able to communicate with any other node in ad-hoc style. At the moment Bluetooth is able to transport data asynchronously or isochronous and voice synchronously.

Bluetooth radio operates at 2.4GHz frequency in the license-free ISM-Band (Industrial, Scientific, and Medical), however since the band is free, Bluetooth has to share it with other systems, like IEEE 802.11b, or home appliances, like microwave oven. Microwave ovens can emit radiation about 1000 times more powerful than the signal one tries to capture causing significant interferences, but fortunately the radiation is only occasional. Since the interferences in the ISM band can be multiple and unexpected, Bluetooth employs FH (Frequency Hopping) spread spectrum technology There are 79 channels used, each with a bandwidth of 1 MHz. During communication, the system makes 1,600 hops per second evenly spread over these channels according to a pseudorandom pattern. The idea is that if one transmits on a bad channel, the next hop, which is only 625 μs later, will hopefully be on a good channel [GePeSm04].

2.1 Bluetooth versions

Bluetooth 1.0A appeared first in 1999 had many problems with compatibility between different devices and also included mandatory Bluetooth hardware device address (BD_ADDR) transmission in the connecting process (rendering anonymity impossible at the protocol level), which was a major setback for certain services planned for use in Bluetooth environments [SIG10A].

Bluetooth 1.0B came later in the same year, December 1999, and tried to fix some of the initial issues [SIG10B].

Bluetooth 1.1 appeared in February 2001 bringing some fixes to errors of its predecessor, support for privacy (encryption) and RSSI (Received Signal Strength Indicator) for controlling power in Bluetooth devices [SIG11].

Bluetooth 1.2 released in November 2003 bringing major improvements like *AFH* (Adaptive Frequency Hopping) with better avoidance of channels with interferences, *eSCO* (extended Synchronous Connection-Oriented) for better audio quality by allowing retransmission of broken

packages, Optional *QoS* (Quality-of-Service) for better error detection, flow controlling and synchronization [SIG12].

Bluetooth 2.0 + EDR (Enhanced Data Rate) appeared in November 2004 and marked the end of the first Bluetooth generation. The second generation aimed to higher transfer rates up to 3 Mb/s, almost tripling the previous bandwidth, and other improvements to power consumption, error handling [SIG20EDR].

Bluetooth 2.1 + EDR appeared in July 2007 and supplements the feature list with *Encryption Pause Resume* to refresh encryption key periodically, *Extended Inquiry Response* to enlarge inquired devices' info, *Simple Secure Pairing* to improve protection against MITM attacks by introducing user interaction, *Near Field Communication* as an *OOB* (Out-of-Band) mechanism, *Sniff Subrating* to reduce power consumption of *HID* (Human Interface Devices), such as mice and keyboards and various improvements of QoS [SIG21EDR].

Bluetooth 3.0 + HS (High Speed) released in April 2009 provides theoretical speeds up to 24 Mb/s obtained via new features such as *L2CAP Enhanced Modes* featuring new L2CAP channel, *Alternative MAC/PHY* enables usage of alternative MACs and PHYs to transport large chunks of data, *Unicast Connectionless Data* permits sending of small amounts of data more rapidly, *Enhanced Power Control* and *Ultra-wideband* [SIG30HS].

Bluetooth 4.0 + LE (Low Energy), nicknamed Bluetooth Smart, released in June 2010 and aims to reduce the power consumption by using a complete new protocol stack required by the appearance of the new chip designs providing single-mode implementation. Chip designs allow for two types of implementation, dual or single modes. Cost-reduced single-mode chips, which enable highly integrated and compact devices, feature a lightweight Link Layer providing ultra-low power idle mode operation, simple device discovery, and reliable point-to-multipoint data transfer with advanced power-save and secure encrypted connections at the lowest possible cost [SIG40LE].

Bluetooth 4.1 + LE announced in December 2013 increments the software stack with new features like increased co-existence support for LTE, bulk data exchange rates and aid developer innovation by allowing devices to support multiple roles simultaneously [SIG41LE].

Bluetooth 4.2 + LE released in December 2014 introduces some key features for Internet of Things such as IP connectivity for Bluetooth Smart enabled devices, the new Internet Protocol Support Profile (IPSP) and IPv6 connection option, and some privacy updates like LE Secure Connections, Link Layer Privacy and Link Layer Extended Scanner Filter Policies [SIG42LE].

2.2 Bluetooth communication

Bluetooth devices that communicate in a group form a *piconet.* A piconet can contain maximum 7 slaves and 1 master. The master is the device initiating the connection and the junction through which all the communication in the piconet pass through, thus all the clocks and frequency hopping intervals are synchronized with the master. Two or more piconets form together a *scatternet* in order to overcome range restrictions, however they still need a neutral device to relay data between piconets. Inside the piconet devices can establish various types of connection for either transferring data or real-time voice.

Asynchronous Connection-Less (ACL) are used to exchange information symmetric in both directions with a total speed up to 1306.9 Kbs, or asymmetric with a total speed of 2178.1 Kbs in sending and 177.1 Kbs in receiving. It also supports data retransmission.

Synchronous Connection Oriented (SCO) is used to transport real-time two-way voice in symmetric mode up to 64 Kbs. Due to the impossibility of retransmitting the packets, voice may become distorted and very high channel *Bit Error Rate* (BER).

Enhanced Synchronous Connection Oriented (eSCO) was introduced in Bluetooth 1.2 to include retransmission of packets. They also have backward compatibility with older devices using SCO.

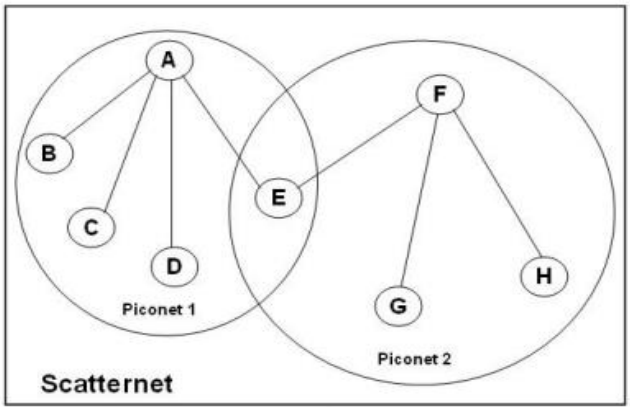

Figure 1.1
Bluetooth topology using ACL links [Ha09]

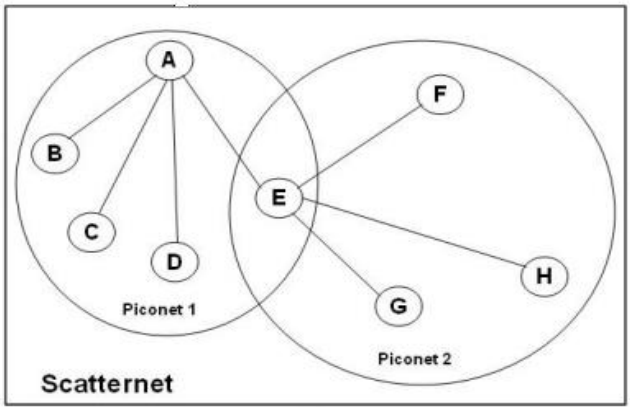

Figure 1.2
Bluetooth topology using SCO links [Ha09]

2.3 Bluetooth characteristics

Bluetooth employs inexpensive compact (5 mm x 5 mm), low power microchips and omnidirectional antennas to connect devices within a medium range with no direct line of sight, therefore are very simple to use and compatible between different manufacturers and types. Still Bluetooth devices must in the same time be very fast and secure. The ranges in which Bluetooth devices can activate depend on multiple factors:

Bluetooth Transmitter (TX) *power* defines the broadcasting range. Usually Bluetooth devices are equipped with either 0 dBm (decibels relative to one milliwatt) transmitter, these are vend as *class III* devices, or 4 dBm transmitter for *class II* devices, or 20 dBm transmitter for *class I* devices. Naturally class I transmitters are the best and most expensive, available only on high-tech equipment, such as military one, and class III are the most common available in ISM.

Bluetooth Receiver (RX) sensitivity can be either -70 dBm (standard level) or -80 dBm (enhanced level).

Path Loss (PL) exponent defines the level of obstacles in the path between the transmitter and the receiver. Theoretically they can be divided into four categories: *none* defining a free space without clutter, PL exponent is *n=2.0, light* defining an office environment with movable walls, *n=2.5, moderate* defining an office with fixed walls, *n=3.0, heavy* defining a building with dense materials, *n=4.0* [Ha09].

Level of obstacles	n	TX power (dBm)	RX sensitivity (dBm)	PL	Range (m)
none	2.0	0	-70	70	32
none	2.0	0	-80	80	100
none	2.0	20	-70	90	316
none	2.0	20	-80	100	1000
moderate	3.0	0	-70	70	10
moderate	3.0	0	-80	80	22
moderate	3.0	20	-70	90	46
moderate	3.0	20	-80	100	100
heavy	4.0	0	-70	70	6
heavy	4.0	0	-80	80	10
heavy	4.0	20	-70	90	18
heavy	4.0	20	-80	100	32

Figure 1.3

Range of Bluetooth devices computed with formula $range = 10^{(PL-40)/(10n)}$ [Ha09]

2.4 Bluetooth protocols

Bluetooth protocol stack can be theoretically partitioned into two different modules *Bluetooth Host* (upper frame) handling the computation work of the Bluetooth functionalities via the higher layers, and *Bluetooth Controller* (lower frame) handling the radio chip via the lower layers.

Host Controller Interface (HCI) ensures the communication between the two modules and the compatibility between manufacturer independent Bluetooth chips.

Physical Layer (PL) contains the Radio Transmitter responsible with processing the radio signals.

Baseband Layer (BL) composed by the lower and upper baseband is responsible with packet formatting, creation of headers, checksum calculations, retransmission procedure, and, optionally, encryption and decryption.

Link Controller (LC) is responsible with implementing the baseband protocols and procedures.

Link Manager (LM) is the entity responsible with managing Bluetooth links.

Link Manager Protocol (LMP) is responsible with setting up the links, negotiating the features and administering connections.

Logical Link Communication and Adaption Protocol (L2CAP) is responsible with formatting large chunks of data into smaller units feasible to the Bluetooth link.

Service Discovery Protocol (SDP) is used to find the services of Bluetooth devices in range.

Radio Frequency Communication (RFCOMM) emulates serial ports over the L2CAP, and therefore it is possible to use existing serial port applications via Bluetooth.

Bluetooth Network Encapsulation Protocol (BNEP) provides networking capabilities for Bluetooth devices. It allows IP (Internet Protocol) packets to be carried in the payload of L2CAP packets.

Telephony Control protocol Specification (TCS) defines the call control signaling for the establishment/release of speech and data calls between Bluetooth devices.

Object Exchange Protocol (OBEX) helps to exchange objects like calendar notes, business cards or data files between devices by using the client-server model [Ha09].

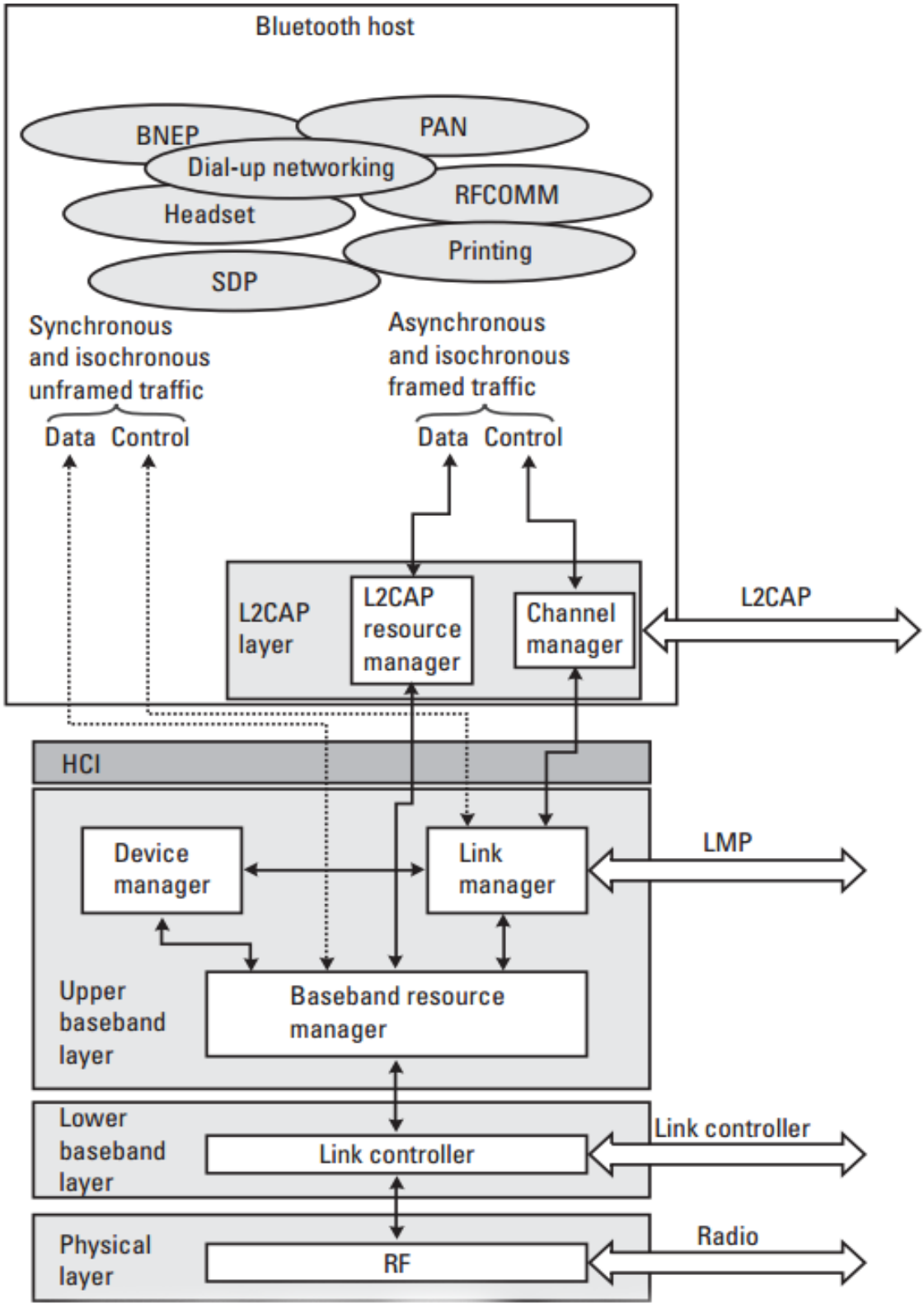

Figure 1.4
A schematic view of the Bluetooth protocol stack architecture [GePeSm04]

3. BLUETOOTH SECURITY

Bluetooth is a wireless network thus every packet is broadcasted through air to all devices in range, however due to the Bluetooth card interface implementation only those cards identifying themselves as destination will submit the packets, the other ones will drop them, this mode is known under *promiscuous* cards. The other type of cards, named *monitor* cards, will submit every packet they receive although there are not the intended destination, making eavesdropping and monitoring a very simple to anyone possessing a device with such a card. In a simple analogy of this model imagine that in a room there are three people, two of them standing at a table talking and the last one standing at bar. If the two people at table talk in a standard language, the third one will be able to hear and understand everything they say. Same principle applies also to Bluetooth communication between two devices, all participants in the "room" will hear them talking, but only those knowing the encryption key will be able to also understand them.

From the security point of view the following properties (procedures) must be ensured:

1. *Authentication* is the procedure by which a unit (the verifier) can convince itself about the (correct) identity of another unit (the claimant) it is communicating with. Note that in cryptography, one often refers to this as the identification, and authentication is reserved for referring to (message or data) authenticity [GePeSm04].
2. *Authorization i*s the process of giving someone permission to do or have access to something. For Bluetooth this means to decide whether a remote device has the right to access a service on the local host and what privileges to gain for it. Usually this involves some form of user interaction. Alternatively, granting access to services can be subject to device-specific settings. Sometimes authorization refers both to administering system permission settings and the actual checking of the permission values when a device is getting access [GePeSm04].
3. *Confidentiality* of data can be achieved by transforming the original data, often called the *plaintext*, into a new text, the *ciphertext*, that does not reveal the content of the plaintext. The transformation should be (conditionally) reversible, allowing the recovery of the plaintext from the ciphertext. To avoid that the transformation itself has to be kept secret to prevent a recovery of the plaintext, the transformation is realized as a parameterized transformation, where only the controlling parameter is kept secret. The controlling parameter is called the

key and the transformation is called *encryption*. A good encryption mechanism has the property that unless the key value is known, it is practically infeasible to recover the plaintext or the key value from the ciphertext [GePeSm04].

First layer of security is always provided by the user who decides the configuration options in terms of connectability and discoverability [Ha09]:

1. *Silent* – device will never accept any connections.
2. *Private or non-discoverable* – device cannot be discovered, connections will only be accepted if the Bluetooth Device Address (BD_ADDR) is known to the prospective master
3. *Public or discoverable devices* – devices can be both discovered and connected to.

Second layer of security is provided by the device manufacturer, which implements the security stack, in this case there are four possible options [Ha09]:

1. *Nonsecure* – device does not implement any security measures at all (e.g. headphones)
2. *Service-level enforced security mode* – devices can establish a non secure ACL link, however if required, authorization, authentication and encryption can be requested at the time of initiation of an L2CAP CO (Logical Link Control and Adaptation Protocol Connection-Oriented) or an L2CAP CL (Logical Link Control and Adaptation Protocol Connection-Less) channel.
3. *Link-level enforced security mode* – security procedures (e.g. authorization, authentication and encryption) are requested at the time of establishment of an ACL link.
4. *Service-level enforced security mode* – similar to *Service-level enforced security mode*, but only for Bluetooth devices above version 2.1+EDR.

3.1 Bluetooth legacy pairing

Pairing represents the process in which one Bluetooth–enabled device in discovery mode can discover and query for services of another Bluetooth–enabled device nearby by establishing a Bluetooth link, or "pairing relationship". The process concludes successful with each party possessing same reusable link key K_{AB}, so that in future they may interact without repeating the process. The flow begins by a PIN (Personal Identification Number) code selection by both devices, note that due to the variety of Bluetooth devices some do not posses input capabilities, thus a fixed unchangeable number is used, usually found in the technical document, such devices

consists of headphones or printers. In this case two devices with fixed PINs cannot be paired. From the PIN the *initialization key* K_{init} is generated. In the second step the initialization key is used to generate the *unit key* K_A or *link key* K_{AB} designating the relationship between the two devices. In the third step the connecting device must identify itself to the target device via the *authentication* stage. Finally in the fourth step devices establish the *encryption key* K_c. Fig. 2.1.1.

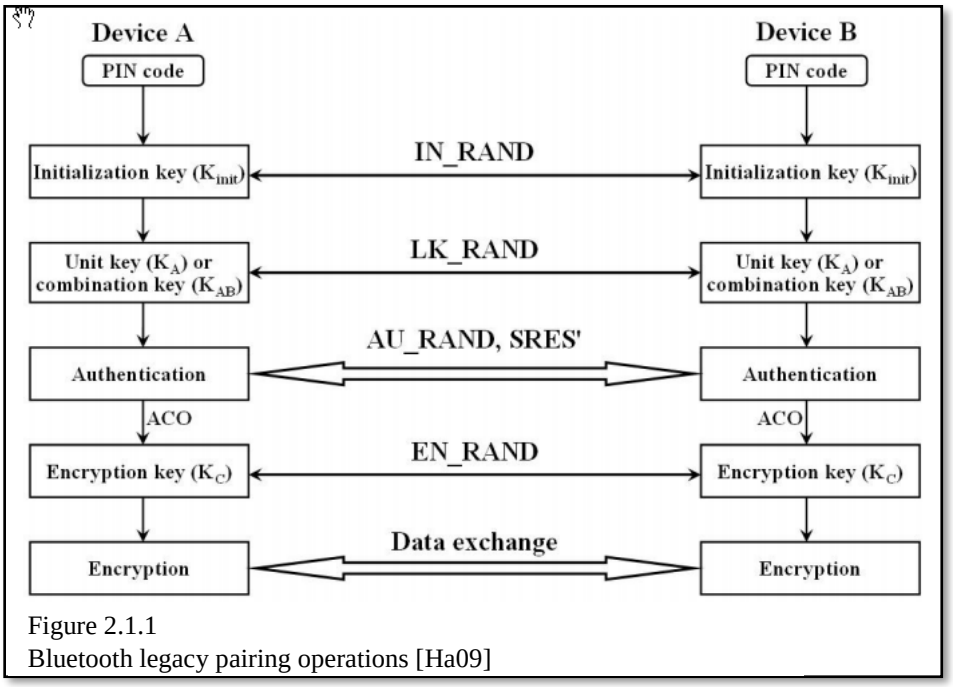

Figure 2.1.1
Bluetooth legacy pairing operations [Ha09]

Initialization key $K_{init} = E_{22}(PIN, L, IN\_RAND)$ generated when devices meet first time.

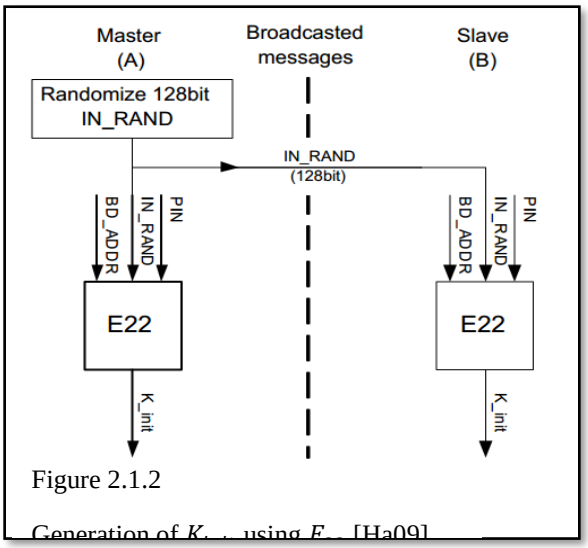

Figure 2.1.2
Generation of K_{init} using E_{22} [Ha09]

E_{22} cipher function

PIN personal identification number

L length of the pin in bytes $1 \leq L \leq 16$

IN_RAND pseudorandom number 16 bytes

BD_ADDR device address is used to augment the PIN number if it is less than 16 bytes [Ha09].

Unit Key K_A=E_{21} $(BD\_ADDR_A, RAND_A)$ is generated by one device, e.g. device A, and then sent to device B encrypted with K_{init} , $A \rightarrow B = K_A \oplus K_{init}$. Device B decrypts K_A with its generated K_{init} , $(K_A \oplus K_{init}) \oplus K_{init} = K_A$, from now on K_A can be used as the link key between devices A and B. However because unit keys are used for several connections any trusted Bluetooth device using the same unit key can impersonate the target device just by duplicating its BD_ADDR. Thus, only devices that have limited resources, i.e., no memory to store several keys, should use K_A, because it provides only a low level of security [Ha09].

Link key $K_{AB} = E_{21}\ (BD\_ADDR_A, LK\_RAND_A) \oplus E_{21}(BD\_ADDR_B, LK\_RAND_B)$ is generated by both devices after exchanging their respective pseudorandom numbers and device addresses. On a closer look we can notice that this is nothing more than a bitwise XOR between devices' unit keys. $K_{AB} = K_A \oplus K_B$, see Figure 2.1.3.

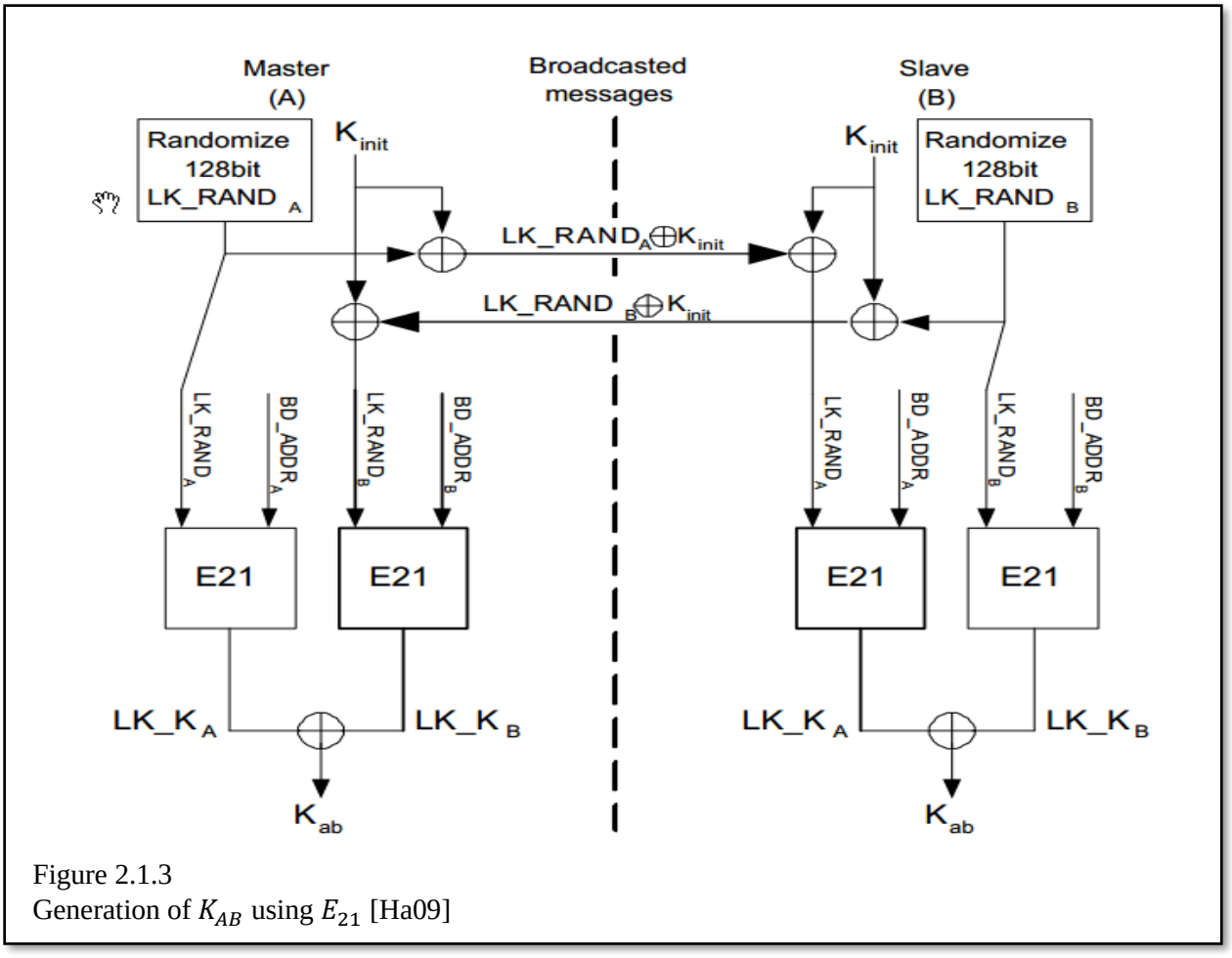

Figure 2.1.3
Generation of K_{AB} using E_{21} [Ha09]

Authentication upon generation of link key K_{AB} each device must authenticate itself to its counterpart. This is done using a challenge-response schema. A challenge-response schema is method through which someone can prove he owns a shared key without actually sending the key though network, thus making it vulnerable for stealing. In the process the parties are divided into claimant, entity asserting possession of the key and verifier, the entity checking claimant's assertion. The process starts with the verifier generating a random token, usually a pseudorandom number and sending it to the claimant, in plaintext, this process is called the challenge. Next the claimant encrypts the token with his key and responds back to the verifier with the ciphertext, this process is called the response. The verifier then performs the decryption of the response, using his shared key with the claimant, and if the tokens match than the claimant is really who pretend to be. Following the above mentioned schema, device A (verifier), generates a pseudorandom 16 bytes number *AU_RAND* and sends it to device B (claimant). Next device B generates a 4 bytes signed response SRES and a 12 bytes authenticated ciphering offset ACO using the following formula $E_1\,(AU\_RAND_A\ ,\ BD\_ADDR_B\ ,K_{Link})$, where K_{Link} is either K_A or K_{AB}. Upon this, device B replays with SRES back to device A and stores ACO for use in the next step of the process. Device A applies the same formula on *AU_RAND* and then compares its SRES obtained value with the SRES received value, and if they match the authentication completes successfully marking the beginning of the next step in the process events, see Figure 2.1.4.

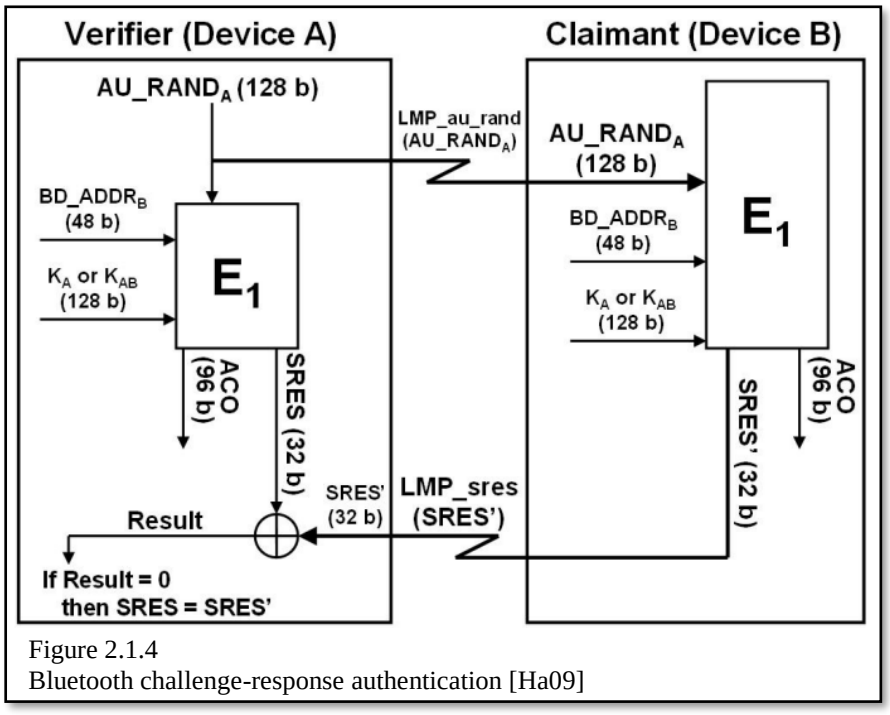

Figure 2.1.4
Bluetooth challenge-response authentication [Ha09]

Encryption of data between two Bluetooth devices is done using an encryption key K_c, generated by both devices using the following formula $K_c = E_3\,(EN\_RAND_A\ , ACO, K_{Link})$, where K_{Link} is either K_A or K_{AB} and EN_RAND_A is generated by master. However, even encrypted, the communication is still vulnerable to reply attacks, thus an auxiliary encryption key is computed for every packet sent, named *keystream, or running key*, using the following formula from K_c, $E_0\,(K_C\ , CLK_{26-1}, BD\_ADDR_A)$, where CLK_{26-1} represents the master's real-time clock. Since the clock is updated for every sent/received message the function's output will also be different. Providing the *keystream,* encryption proceeds by XORing the plaintext with it [Ha09]:

Ciphertext = Plaintext $\oplus$ *Keystream.*

In case of decryption the ciphertext is XORed with the keystream:

Plaintext = Ciphertext $\oplus$ *Keystream = (Plaintext* $\oplus$ *Keystream)* $\oplus$ *Keystream.*

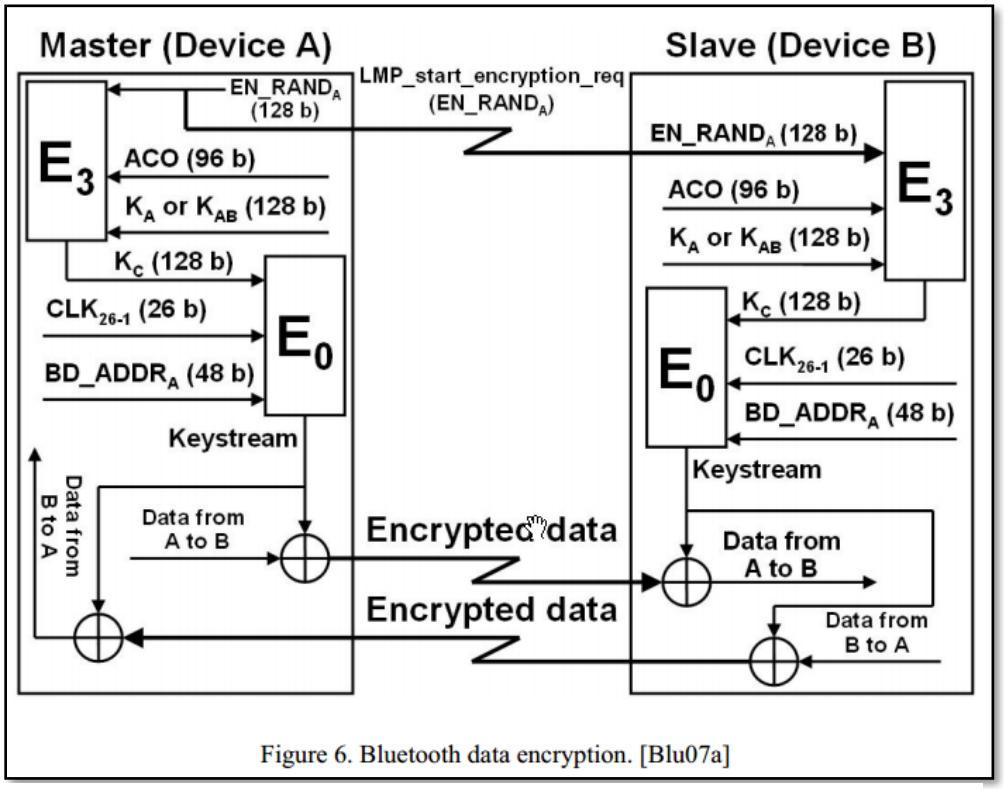

Figure 6. Bluetooth data encryption. [Blu07a]

Figure 2.1.5
Bluetooth data encryption [Ha09]

3.2 Bluetooth repairing the pairing protocol

Several severe attacks have been proposed in [WoStCl05] by researchers Ford-Long Wong, Frank Stajano and Jolyon Clulow including that short Bluetooth PINs can be cracked by brute force search with modest computational resources (see section 4.1), hence it is imperative to find a strong, eavesdropper-resistant linking key. Because of the thrifty constraints in the Bluetooth design there is no way to accomplish this, so the authors resorted to asymmetric cryptography, in particular Diffie-Hellman protocol on elliptic curves, and Encrypted Key Exchange (EKE) as MITM protection [WoStCl05].

The protocol states that the participants in pairing must share a weak password *P*. They hash the password with their device addresses to produce the password hash *s* which is then multiplied with *G*, the common base point of the elliptic curve, to produce *V*. Alice sends her ephemeral public key Gr_A. Both Bob and Alice hash their identifiers with Alice's public key to obtain e_1. Bob multiplies Alice's public key by e_1, adds it to *V*, and multiplies this with Bob's private key r_B. This result is Bob's password-entangled public key Q_B. Bob sends Q_B to Alice. Both parties would hash both of their identifiers and both parties' public keys together to obtain e_2. Alice computes ω, using her secret key r_A, the password hash *s*, and the values of e_1. and e_2 computed earlier. After obtaining ω, and using Bob's password-entangled public key Q_B, Alice is able to calculate $(r_A + e_2).r_B.G$ and derive the shared key. Over at Bob's end, he knows r_B, and using Alice's public key Q_A and the value of e_2 he has computed, Bob would likewise be able to calculate $(r_A + e_2).r_B.G$ and derive the shared key [WoStCl05].

Due to Diffie-Hellman number-theoretic problem can be asserted that passive eavesdropper would not able to compute the shared session key, unless he knows either r_A or r_B. If an active adversary, Eve, may attempt to carry out a protocol run in order to obtain a trace to conduct a password guess (either online or offline) she does not have a high chance of success. If Eve attempts to masquerade as Alice, she has one chance at correctly guessing the password hash *s*, so as to calculate the correct *K* and subsequently send the correct M_1 to Bob. If she fails at one try, Bob will abort the protocol run without revealing more than one wrong guess. If Eve attempts to masquerade as Bob, and she contributes a password-entangled public key while not knowing the password hash *s*, even if she manages to collect an M_1 sent by Alice, Eve would need to solve the

Diffie-Hellman problem to recover the corresponding r_B that would produce the same *K* solution which Alice has calculated.

The protocol also ensures perfect forward secrecy through Diffie-Hellman, so an adversary is not able to calculate past session keys after knowing *P* or *s*.

#	Alice		Bob
	$s = H_0(I_A, I_B, P)$		$s = H_0(I_A, I_B, P)$
	$V = G.s$		$V = G.s$
	Chooses random r_A		
	$Q_A = G.r_A$		
1		$\xrightarrow{Q_A}$	
	$e_1 = H_1(I_A, I_B, Q_A)$		$e_1 = H_1(I_A, I_B, Q_A)$
			Chooses random r_B
			$Q_B = (Q_A.e_1 + V).r_B$
2		$\xleftarrow{Q_B}$	
	$e_2 = H_2(I_A, I_B, Q_A, Q_B)$		$e_2 = H_2(I_A, I_B, Q_A, Q_B)$
	$\omega = (r_A e_1 + s)^{-1}(r_A + e_2)$		
	$K = H_3(h.Q_B.\omega)$		$K' = H_3(h.(Q_A + G.e_2).r_B)$
	$M_1 = H_4(I_A, I_B, Q_A, Q_B, K)$		
3		$\xrightarrow{M_1}$	
			$M_1' = H_4(I_A, I_B, Q_A, Q_B, K')$
			Verifies $M_1 = M_1'$
			$M_2 = H_5(I_A, I_B, Q_A, Q_B, K')$
4		$\xleftarrow{M_2}$	
	$M_2' = H_5(I_A, I_B, Q_A, Q_B, K)$		
	Verifies $M_2 = M_2'$		

Figure 2.2.1
Repairing Bluetooth pairing protocol – password based key agreement [WoStCl05]

3.3 Bluetooth Simple Secure Pairing

In Bluetooth versions up to 2.0+EDR pairing is based only on PIN numbers so the level of security is as strong as the PIN's strength. Since PIN values can vary in length from 1 character to 16 character numeric digits, but usually is 4 characters, thus the max entropy achievable is about 53 bits, not barely close to the 95 bits randomness achieved by using the whole alphanumeric alphabet (62 characters [0…9, a…z, A…Z]). Moreover it has been proved that even the longer 16 character PINs are vulnerable to MITM attacks, so following version 2.1+EDR the pairing procedure was shift entirely to using SSP (Simple Secure Pairing) [SSP01].
SSP employs P192 elliptic curve Diffie-Hellman protocol to construct the linking key using public-private key pairs, a couple of nonces and addresses of the devices. ECDH cryptography has the advantage over standard DH of being less computationally extensive and less space, thus making it more suitable for the common low powered Bluetooth chips. Moreover the brute force search on a 16 character, case sensitive, alphanumeric, private key is very exhaustive and infeasible in reasonable time, eavesdropping attacks are effectively thwarted.
SSP provides measures of protection against MITM attacks by using either out-of-bad channels, such as Near Field Communication, or by employing user's interaction. Depending on user's device capabilities, four models of pairing can be established.

1. Numerical comparison association model is designed for scenarios when both devices feature displays and keyboards such that are capable of displaying a six digit number and also having the user enter "Yes" or "No". As an example of this model is the pairing between a cell phone and computer. At first glance this model may look much like the PIN entry model from legacy pairing, however in this case the number represents an artifact of the security algorithm and not an input to it, thus knowing the number will not help in seeing the final linking key and decryption of exchanged data.

2. Passkey entry association model is designed for scenarios where one of the devices has no output capabilities to display the six digit number, just like in the case of pairing between a computer and keyboard.

3. Just works association model is designed for scenarios where one of the devices has both no output nor input capabilities, such as the case of the pairing between a cell phone and headsets, in

this case a variant of the numerical comparison protocol is used, however no number is ever displayed to the user, but simple request to accept the connection via a button. From the cryptographic point of view the just works model provides same level of security against eavesdropping attacks as the other models, but no protection whatsoever against man-in-the-middle attacks.

4. Out Of Band association model is designed for scenarios when other mechanisms of authentication can be used, such as NFC (Near Field Communication), where users are asked first to touch the two devices together and second to confirm the pairing by entering "Yes". During the touching phase both devices exchange information about devices' addresses and cryptographic keys following by one of the device's request to start the connection. This association model is selected when both devices advertise OOB capabilities into its IO capabilities exchange list.

All the Bluetooth device capabilities and SSP association models construction are depicted in the below table, whereas *DisplayYesNo* indicates the device has display capabilities and at least two buttons mapped "Yes" and "No" for accepting respective declining connection, other notations are self-explanatory.

Device 1	Device 2	Association model
DisplayYesNo	DisplayYesNo	Numerical Comparison
	DisplayOnly	Numerical Comparison
	KeyboardOnly	Passkey Entry
	NoInputNoOutput	Just Works
DisplayOnly	DisplayOnly	Numerical Comparison
	KeyboardOnly	Passkey Entry
	NoInputNoOutput	Just Works
KeyboardOnly	KeyboardOnly	Passkey Entry
	NoInputNoOutput	Just Works
NoInputNoOutput	NoInputNoOutput	Just Works

Figure 2.3.1
SSP association models [SIGWP06]

SSP security protocol is merely consisted by the same steps as in legacy protocol with the mentioned adjustments to the authentication step and usage of ECDH cryptography.

1. Capabilities exchange devices that have never met before, or try re-pairing, first have to exchange their Input/Output capabilities in order to determine the association model.

2. Public key exchange initially each devices generates is own ECDH P192 public-private key pair, moreover this can be done in advance of pairing. Usually the key pair is generated only once per device, although a device can choose at any time to discard and recreate it. Pairing is starting by the initiating device sending its public key, after which the receiver responds with its corresponding one. After the exchange both devices compute the shared Diffie-Hellman key.

3. Authentication stage I the authentication protocol running at this stage depends on the association model selected, however no matter the protocol, each promotes the same purpose to ensure MITM protection.

3.1. Authentication via numerical comparison the protocol starts by generating a pseudorandom 128-bit nonce by each device (step 2) *Na* and *Nb*, in order to counter reply attacks these values must be fresh and unique with each instance of pairing. The responding device then computes a commitment of the devices' public keys and its nonce (step 3c) using a one-way hash function, and forwards it to the initiator. The commitment prevents an attacker from tempering the values. Following in steps 5 and 6 the exchange of their nonces and confirmation by the initiator of receiver commitment. Any error at this point caused by either the environment or an attacker will abort the process, following the restarting from the step of nonce generation (step 2). Otherwise the commitment check succeeds and the protocol proceeds with the generation by both devices of the 6-digit confirmation values to be displayed to the user (step 7a, 7b, 8). Providing the user infirm the match, the protocol is restated from the nonce generation step.
A simple MITM attack resulting in distinction of the two displayed numbers has a probability of success about 1 to 1000000, however a more sophisticated version may attempt to engineer these values, but with less success due to commitment sequence. If an attacker first exchanges nonces with initiating device, it must also send a nonce to the responding device before seeing the nonce from the responding device, therefore the attacker must commit to at least the second of its nonces before knowing the second nonce from the legitimate devices, otherwise the displayed values will not match.

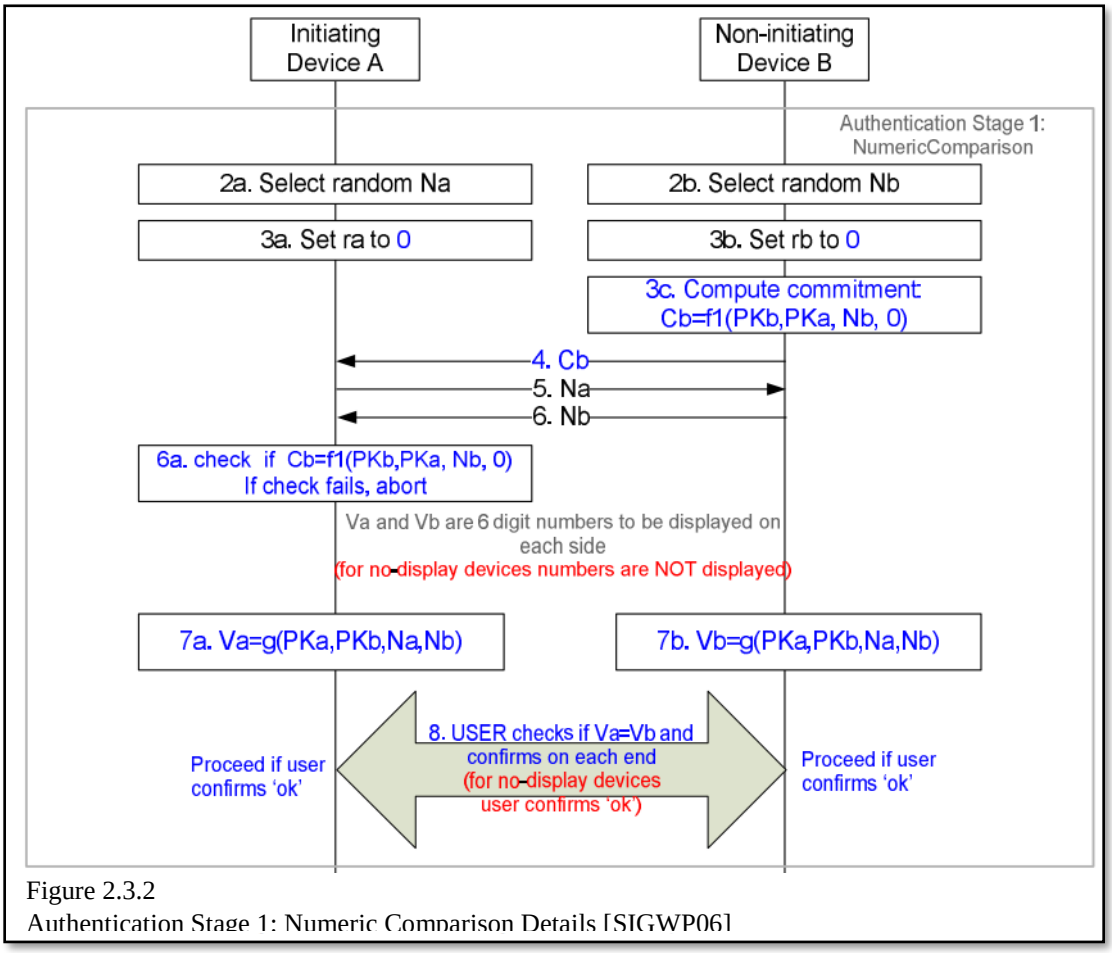

Figure 2.3.2
Authentication Stage 1: Numeric Comparison Details [SIGWP06]

3.2. Authentication via Out-of-Band

Assuming that both devices can transmit and/or receive data over an out-of-band channel then mutual authentication will be executed on the two commitments of the public keys (*Ca* and *Cb*), however if the communication is only possible in one direction (e.g. devices equipped with passive NFC tags can only be read) then authentication will be based on the device receiving the OOB information knowing a random number *r* send via OOB. In this case is mandatory that *r* should remain secret and fresh, moreover if *r* cannot be read then it asserted to 0.

Most important attribute of OOB is its natural protection against MITM attacks, since in this case messages cannot be altered. Another important feature is that the size of the authentication parameters are not restricted by usability, or human read/type capabilities, thus making this model most secure among others, however requires special interfaces.

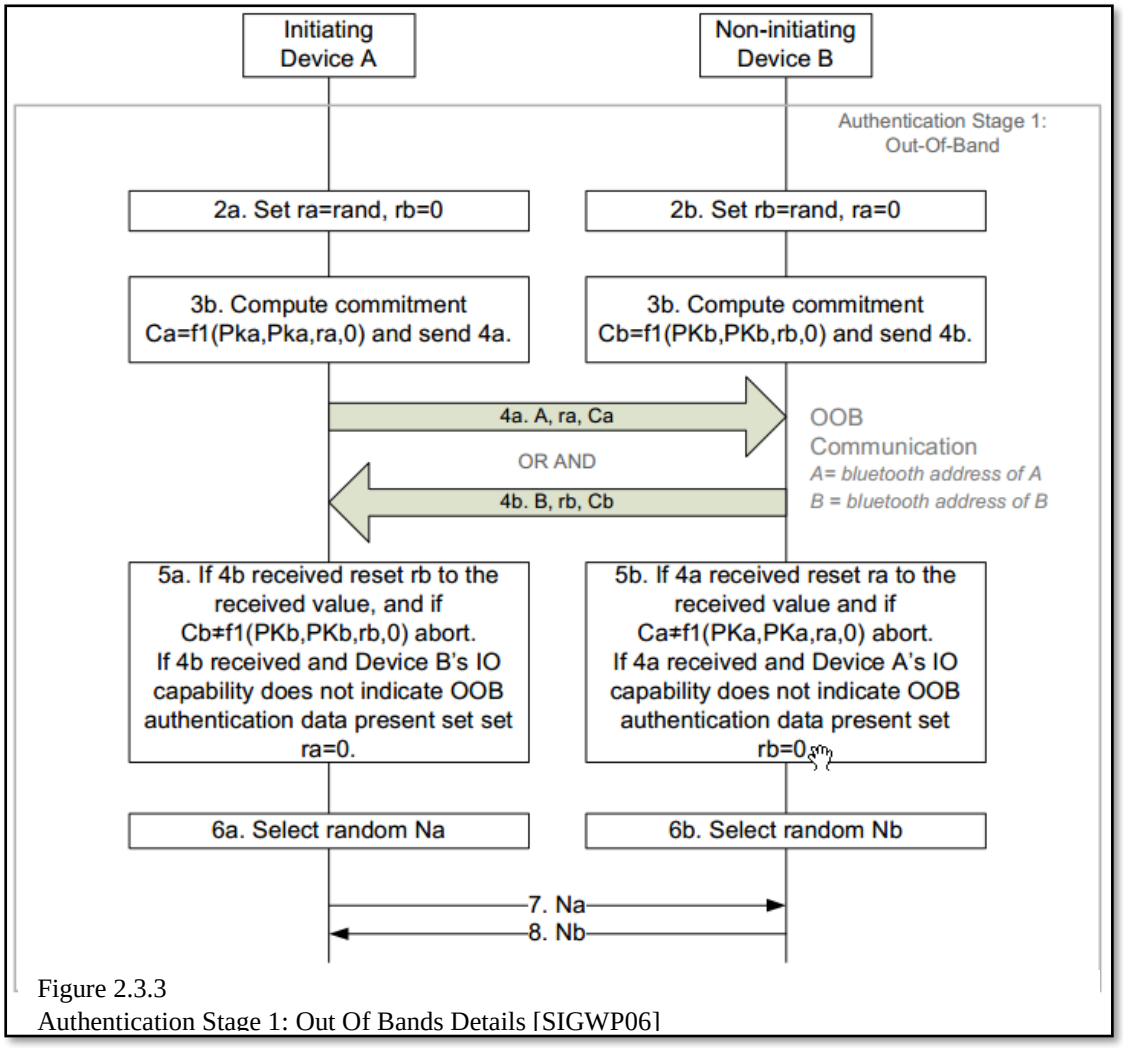

Figure 2.3.3
Authentication Stage 1: Out Of Bands Details [SIGWP06]

OOB is subjective in terms of the initiator role and any of the devices can initiate the pairing, although when computing the linking key in step 12 both parties must input the parameters in same order otherwise resulting keys may differ.

OOB is also subjective to the time of public key exchange step and can occur before or after OOB communication (step 4), but no later than step 5 verification.

Since the direction of the peer's OOB interface cannot be checked before the communication step, a device will always generate, and if possible transmit, a random number *r* to the peer. Initially the device assigns a random number to its *r* and 0 to its peer *r*. When a device receives the actual value of *r* then will update it. If the remote device does not indicate that it has received OOB authentication data, it sets its own r value to 0.

Near Field Communication (NFC) provides devices with a variety of OOB modes and operations, such as different data rates (106 Kbps, 212 kbps, or 424 kbps), different modes of operation (active or passive), or different capabilities to initiate (initiator or reader) and to accept (tag or target) connections:

1. OOB-IO NFC device has the ability to communicate with Bluetooth controller and also transmit/receive data to other NFC device.
2. OOB-O NFC device can only transmit data.

In order to successfully establish a NFC communication at least one device must possess an OOB-IO card, otherwise information cannot be read.

At the end of authentication stage I both devices should share same values *ra* and *rb*.

3.3. Authentication via Passkey Entry

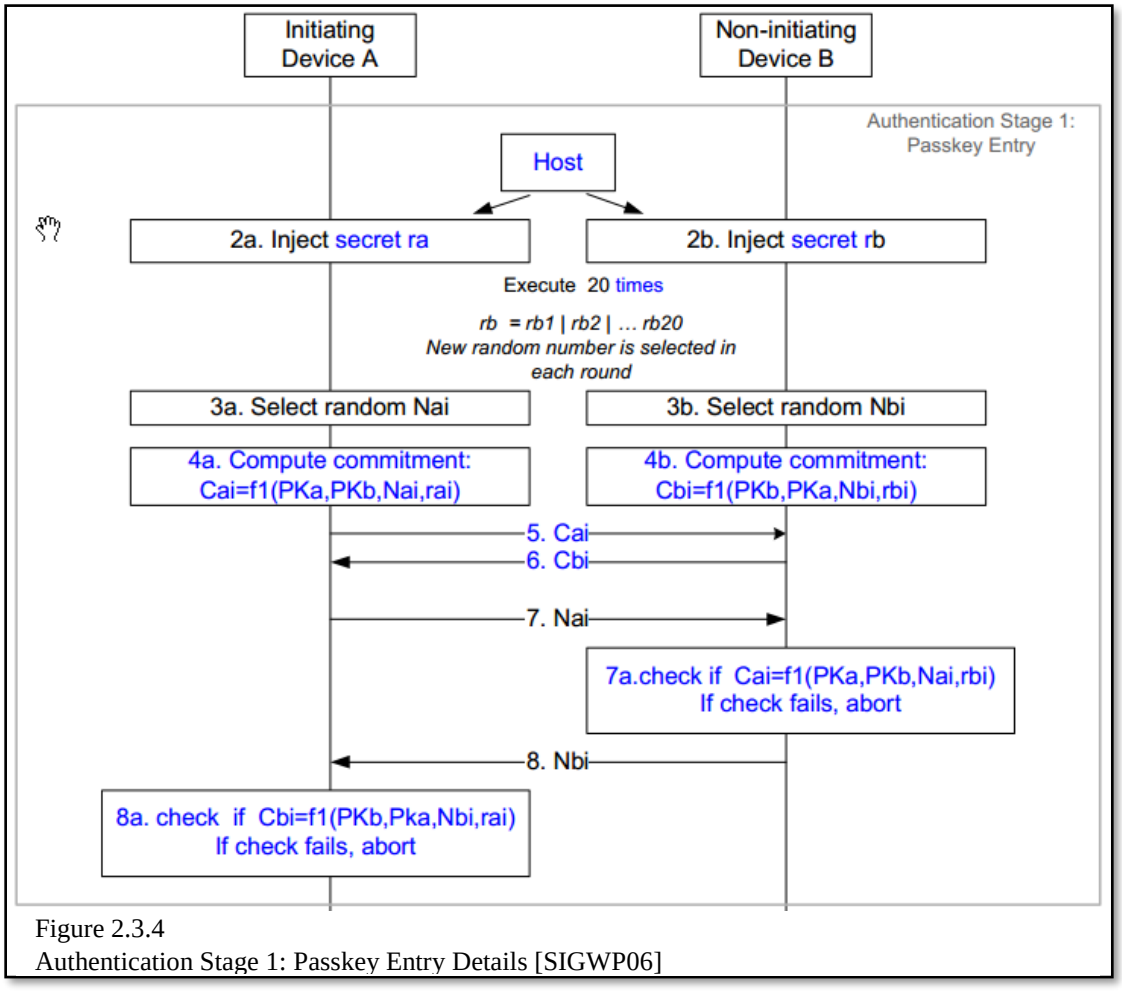

Figure 2.3.4
Authentication Stage 1: Passkey Entry Details [SIGWP06]

The users either input identical passkeys into both devices, or one device displays the passkey and the other one inserts it (step 2). In steps 3 through 8 each side commits each bit of the passkey, a nonce, the two public keys in a one-way hash function. The parties then take turns revealing their commitments until the entire passkey has been mutually disclosed. The first party to reveal a commitment for a given bit of the passkey effectively reveals that bit of the passkey in the process, but the other party then has to reveal the corresponding commitment to show the same bit value for that bit of the passkey, or else the first party will then abort the protocol, after which no more bits of the passkey are revealed.

This gradual disclosure occurs for every bit of the passkey, as example for a 6-digit passkey *k* equals 20 (999999=0xF423F), and serves the purpose of preventing the leakage of more than 1 bit passkey information to an MITM attacker, hence, a MITM attacker who engages first one side, then the other will only gain an advantage of at most 2 bits over a simple brute-force guesser which succeeds with probability 0.000001.

The long nonce in commitment hash prevents a brute-force attack even after protocol has failed.

The public keys are included in commitment hash to prevent an attacker from substituting his public key on both sides of the ECDH exchange in the standard MITM attack on ECDH.

At the end of this stage *Na* takes the value of *Na20* and *Nb* of *Nb20.*

4. Authentication stage II

Each device computes a new confirmation value that includes the previously exchanged values and the newly derived shared key (step 9). The initiating device then transmits its confirmation value which is checked by the responding device (step 10). If this check fails, it indicates that the initiating device has not confirmed the pairing and the protocol is aborted. Otherwise the responding device transmits its confirmation value which is checked by the initiating device (step 11). Likewise if the check fails then the responding device has not confirmed the pairing and the protocol is aborted.

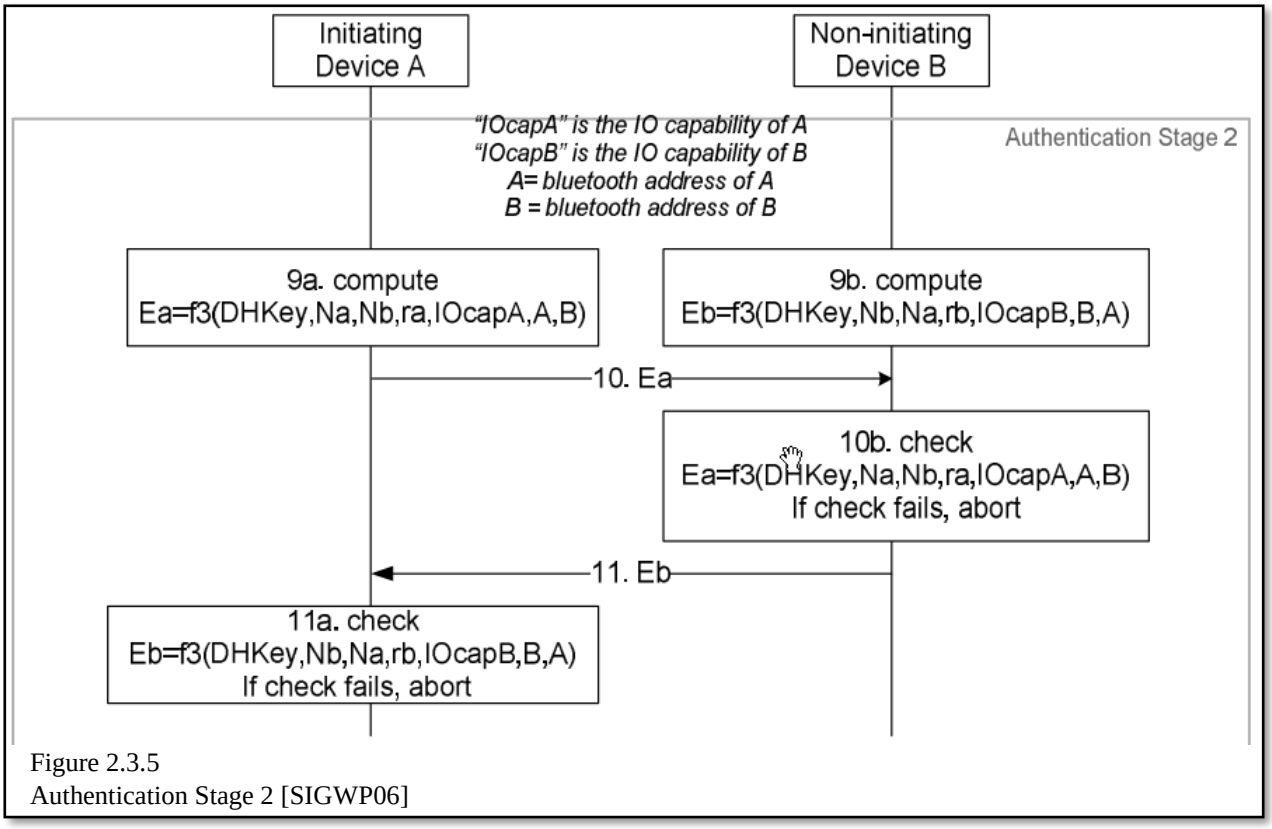

Figure 2.3.5
Authentication Stage 2 [SIGWP06]

5. Link key calculation the parties compute the link key using their Bluetooth addresses, the previously exchanged values (step 12), and the Diffie-Hellman key. The link key is used to ensure the pairing between devices on long term. The nonces ensure as always the freshness of messages even if same ECDH public keys are reused.

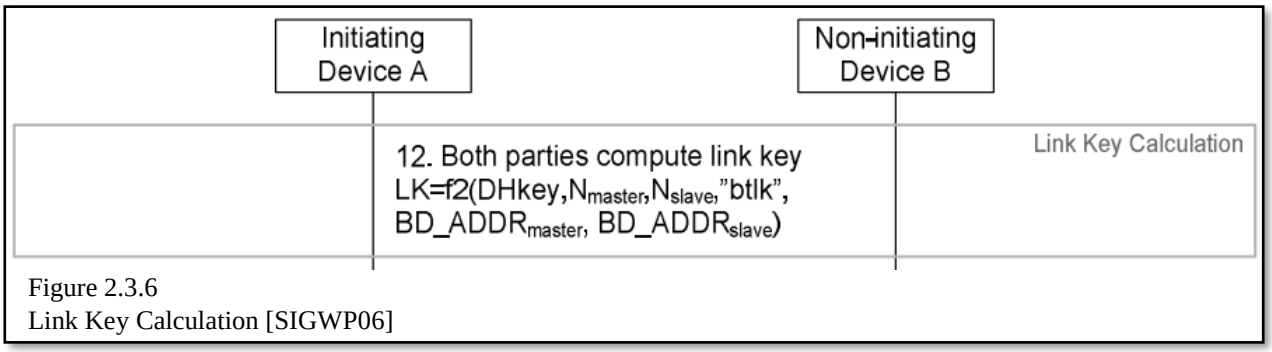

Figure 2.3.6
Link Key Calculation [SIGWP06]

6. LMP authentication and encryption defines the generation of encryption keys in the same way as for the legacy protocol.

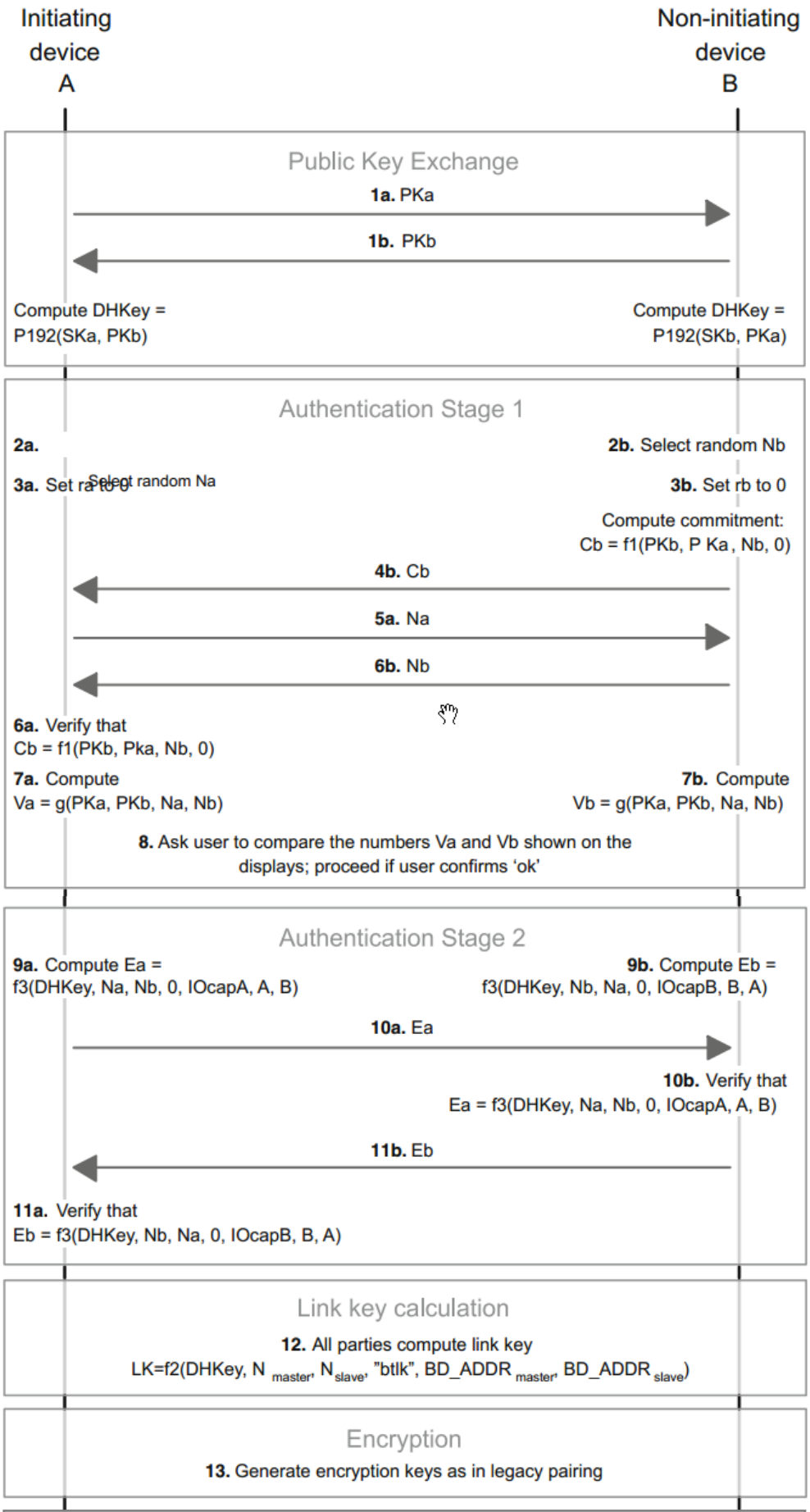

Figure 2.3.7 – overview of the SSP steps [SIGWP06]

3.4 Bluetooth cryptographic procedures

Bluetooth make use of several ciphering functions, namely (E_{22} ,E_{21}) in pairing, E_1 in authentication and (E_3 ,E_0) in encryption state. Among these (E_{22} , E_{21}, E_3, E_1) are variations of the 128-bit block cipher SAFER+ (Secure and Fast Encryption Routine +) algorithm and E_0 representing the encryption algorithm, which uses a stream cipher with a 132-bit initial state.

SAFER+ was developed by Massey et al. in 1998 and submitted against AES (Advanced Encryption Algorithm) in a contest organized by NIST (National Institute of Standards and Technology) in order to replace the obsolete standard at the moment, e.g. DES (Data Encryption Standard), however AES has proved itself more secure and SAFER+ was dumped. Nevertheless Bluetooth specification up to 3.0+HS inclusive has adopted it as the standard ciphering algorithm due to its low computation costs and fast processing rounds with a relatively secure level make it a viable choice for underpowered Bluetooth devices, however in 2010 it was finally superseded by AES. Internally, SAFER+ consists of:

1. KSA a key schedule algorithm producing 17 different subkeys of 16 bytes, denoted K_1 to K_{17}, which are used by the SAFER+ rounds 2 subkeys each, apart from the last one used in the output transformation. Each subkey consists of 16 bytes out of 17, every byte is cyclic-rotated left by 3 bits and finally added to a 16 bytes bias vector.
2. Eight identical rounds. Every round outputs a 16 byte word composed from 2 subkeys and previous output word. The central components of the SAFER+ round are the 2-2 Pseudo Hadamard Transform (PHT), the Armenian Shuffles, and the substitution boxes denoted “e” and “l”. The Pseudo Hadamard Transform takes two input bytes and produces two output bytes using the following formula:

   PHT[a, b] = [(2a + b) mod 256, (a + b) mod 256].

   The Armenian Shuffle is a permutation of 16 bytes.

   The substitution boxes “e” and “l” are non-linear, both replace an input byte with an output byte using the following equations:

   e(x) = (45^x*(mod 257)) mod 256*

   l(x) = y e(y) = x

   $l = e^{-1}$

3. An output transformation representing a XOR between the last round output, e.g. last word, and the last subkey, e.g. K_{17} [ShWo05].

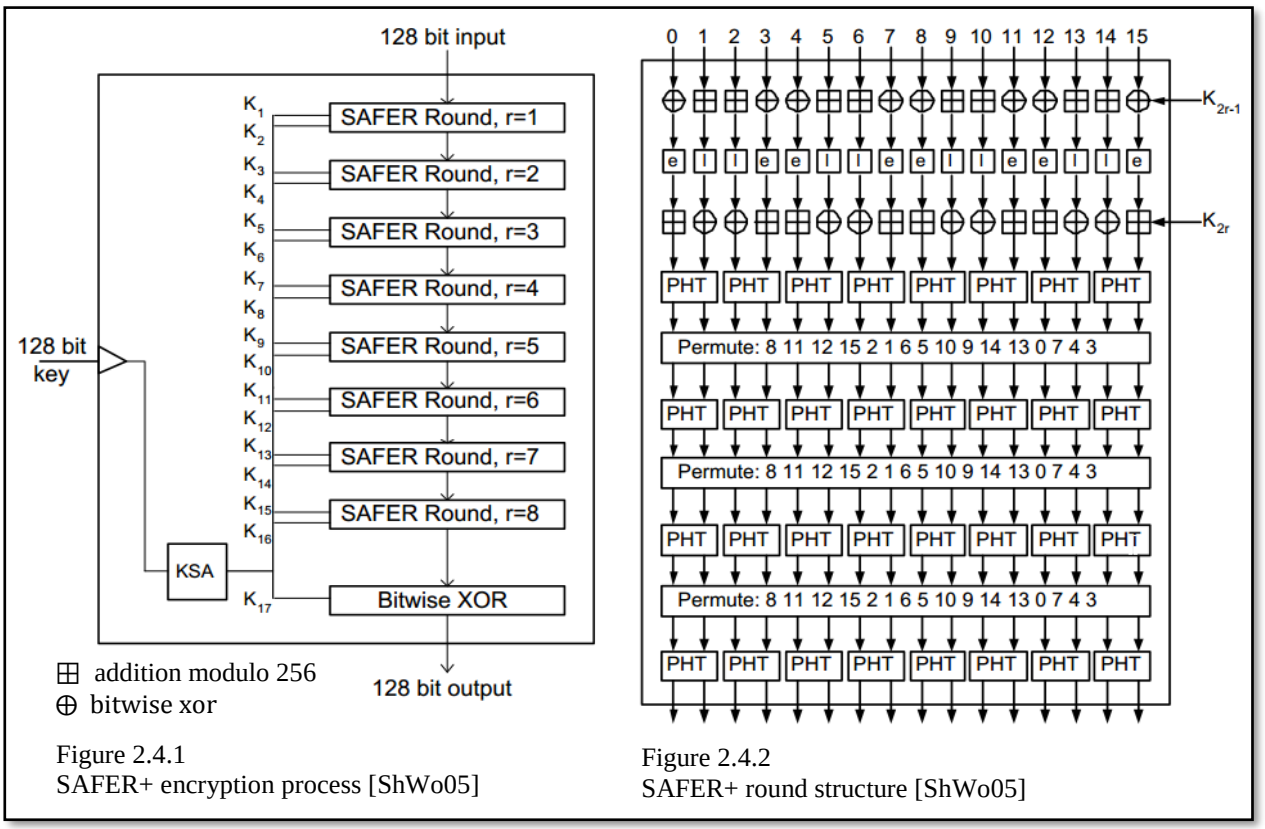

Figure 2.4.1
SAFER+ encryption process [ShWo05]

Figure 2.4.2
SAFER+ round structure [ShWo05]

E_0 *stream cipher* is based on a direct design and uses a Bluetooth proprietary algorithm that has its roots in the so-called summation combiner stream cipher. This cipher was developed by Massey and Rueppel in mid 1980s. The most powerful attacks know on this cipher mechanism are the correlation attacks in combination with exhaustive search over a limited key space, or more common known as initial guessing. Golic, Hermelin and Nyberg have proven that a summation combiner type of stream cipher with a total state space of K-bits (2^K states) will provide only about K/2 bits in security when sufficient key stream data is available. Stream ciphers are ideal in communication systems because they accept text input of various lengths and are designed with low implementation costs, but block cipher algorithms can be used as well in conjunction with a padding scheme. E_0 is built around four independent linear feedback registers and a finite state machine as combining circuitry. The latter is needed to introduce sufficient nonlinearity to make it difficult to recompute the initial state from observing key stream data.

The four linear feedback registers $LFSR_i$, $i = 1, 2, 3, 4$ are each fully characterized by the following four feedback polynomials:

$$LFSR_1 : f_1(t) = t^{25} + t^{20} + t^{12} + t^8 + 1$$

$$LFSR_2 : f_2(t) = t^{31} + t^{24} + t^{16} + t^{12} + 1$$

$$LFSR_3 : f_3(t) = t^{33} + t^{28} + t^{24} + t^4 + 1$$

$$LFSR_4 : f_3(t) = t^{39} + t^{36} + t^{28} + t^4 + 1$$

The output sequence $X_1 = (x_{10}, x_{11}, \dots)$ of register 1 can be expressed formally as:

$$X_1(t) = \sum_{i=0}^{\infty} x_1 \, t_i^i$$

According to theory of linear feedback registers we can rewrite the above as:

$$X_1(t) = \frac{g_1(t)}{f_1(t)}$$

Where $g_1(t) < f_1(t)$ and the polynomial division is carried out by using ordinary polynomial arithmetic but using modulo 2 arithmetic in the coefficients. It is the linear feedback circuit that implements this division operation using delay elements to hold the coefficients and XOR gates to do the modulo 2 operations. Each of the four polynomials is a so-called maximum length polynomial, which means that the periods of the output sequences of $LFSR_s$ have periods $2^{degree_i} - 1$, $i = 1, 2, 3, 4$

$$period\ X_1 : P_1 = 2^{25} - 1$$

$$period\ X_2 : P_2 = 2^{31} - 1$$

$$period\ X_3 : P_3 = 2^{33} - 1$$

$$period\ X_4 : P_4 = 2^{39} - 1$$

The four sequences $X_1 \dots X_4$ are fed symbol by symbol into a so-called summation combiner which adds the four input symbols together as if they were natural numbers, adds the result to a number c_t, depending on the summation combiner's state, and obtains a sum $s_t, t = 0,1, \dots$

$$s_t = x_{1t} + x_{2t} + x_{3t} + x_{4t} + c_t \in \{0,1, \dots 7\}$$

Because c_t takes on only the values 0, 1, 2, 3 the output symbol z_t is the binary result obtained by

$$z_t = s_t \, mod \, 2 = x_{1t} \oplus x_{2t} \oplus x_{3t} \oplus x_{4t} \oplus (c_t \, mod \, 2)$$

The new value c_{t+1} is obtained by rewriting first the result of the computation as a binary vector u_{t+1} (of dimension 2)

$$u_{t+1} = \left[\frac{s_t}{2}\right]$$

$$c_{t+1} = \begin{pmatrix} c_{0\,t+1} \\ c_{1\,t+1} \end{pmatrix} = u_{t+1} \oplus \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} c_t \oplus \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} c_{t-1}$$

Computing modulo 2 in the coefficients, and finally defining the mapping:

$$c_t = 2c_{1t} + c_{0t}$$

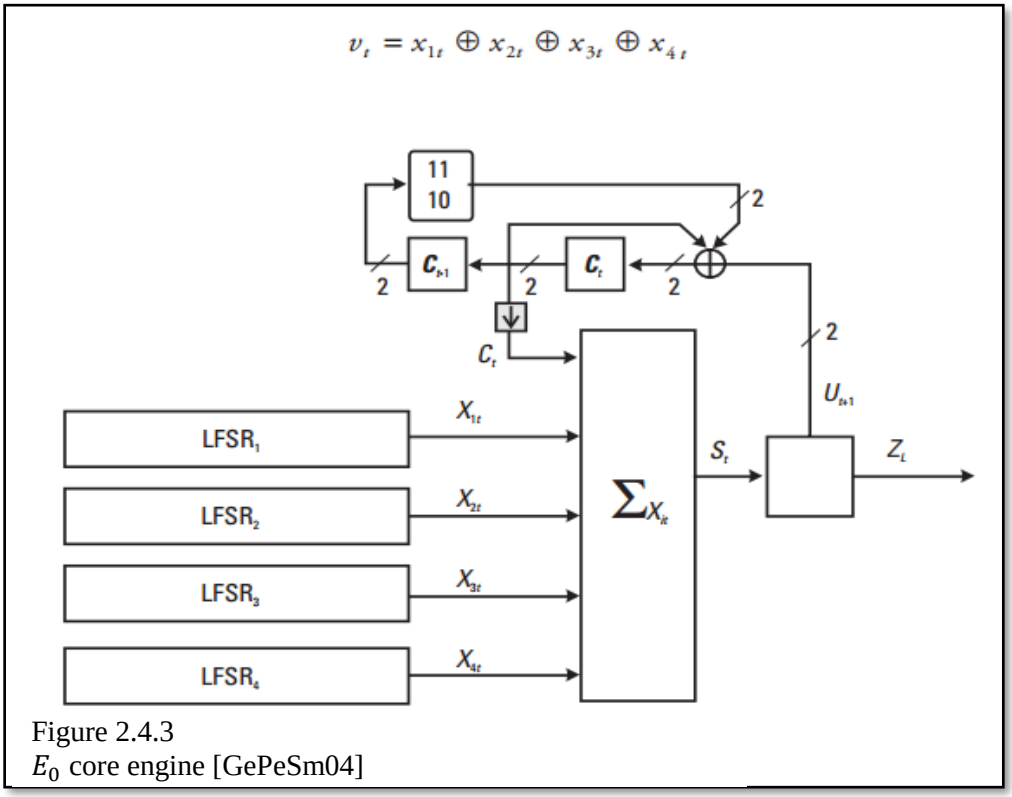

Figure 2.4.3
E_0 core engine [GePeSm04]

Advanced Standard Encryption was also introduced into Bluetooth specification starting with version 4.0+LE as replacement for SAFER+. The algorithm was developed by two Belgian cryptographers "Joan Daemen" and "Vincent Rijment", it belongs to the Rijndael cipher family, and was adopted by the U.S government as most suitable encryption algorithm for top secret classified information. AES is considered secure, fast and compact (1KB of code) [BT40LE].

AES operates on a fixed data block of 16 bytes, represented by a 4x4 matrix, and a key of size 16, 48 or 64 bytes, processed in rounds consisting of 4 different steps. Every round is iterated multiple times depending on key's length, 10 cycles for 16 byte keys, 12 cycles for 48 byte keys, and 14 cycles for 64 byte keys, in the following steps per round:

1. *Byte substitution* is a non-linear substitution step where each byte is replaced with another one according to Rijndael S-box lookup table [FIPS197, Figure 7, pp 16].
2. *Row shifting* is a transposition step where the last three rows of the state are shifted cyclically a certain number of steps.
3. *Column mixing* is a substitution which makes use of arithmetic over GF(2^8) (Galois Field), the finite field of 256 elements, which can be denoted by strings of eight bits or by hexadecimal notation [GF28].
4. *Round key adding* is a simple bitwise XOR of the current block with a portion of the expanded key.

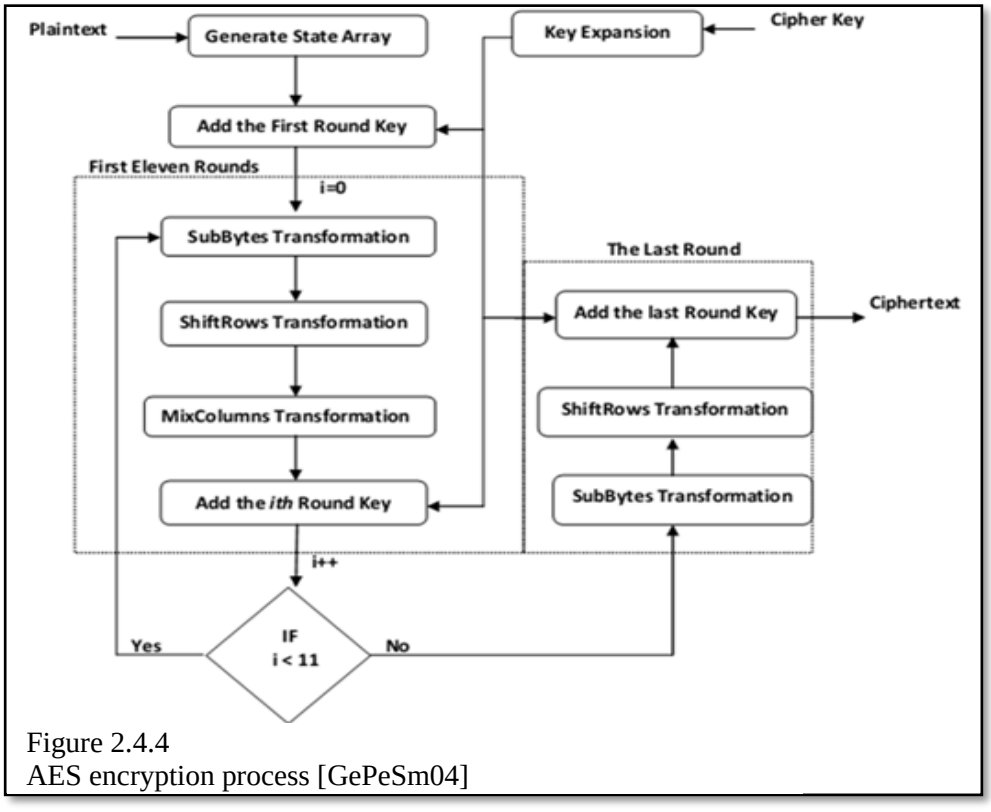

Figure 2.4.4
AES encryption process [GePeSm04]

AES algorithm can be inverted to reproduce the original plaintext by inversing individual steps:

1. *Inverse row shifting* the last three rows of the state are shifted cyclically a certain number of steps in opposite direction.
2. *Inverse byte substitution* is a non-linear substitution of each byte with another one from Rijndael inverse S-box this time.
3. *Round key adding* is its own inversion since XOR operations are associative: $x \oplus x = 0$.
4. *Inverse column mixing*

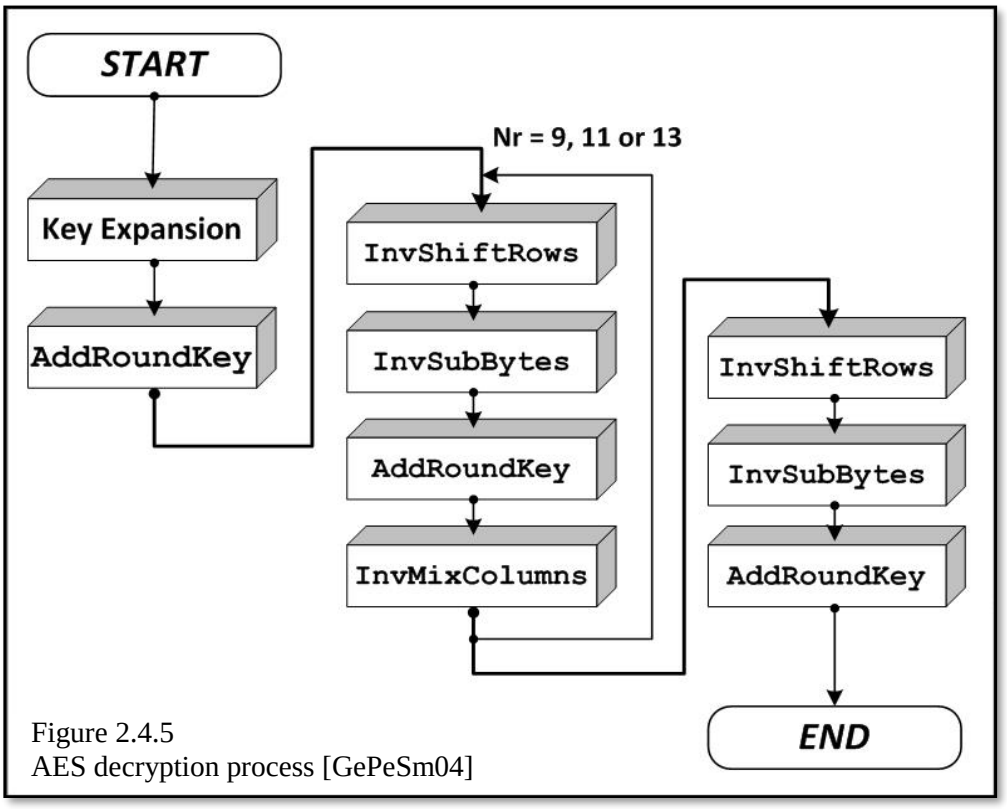

Figure 2.4.5
AES decryption process [GePeSm04]

Elliptic Curve Cryptography (ECC) present in Bluetooth Repairing Procedure and Simple Secure Pairing represents a variant of public-key cryptography based on the algebraic structure of elliptic curves over finite fields. Elliptic curves can have points with coordinates in any field, such $\mathbb{F}_p$ as , Q, R, or C. Elliptic curves with $\mathbb{F}_p$ points in are finite groups.

The primary advantage of using ECC is due to its smaller key size requirements thus less storage and transmission resources, while maintaining the same level of security as an RSA based system, e.g. a 256-bit ECC public key should provide same security as 1056-bit RSA public key.

ECC protocols are based on the problem of the discrete logarithm of a random elliptic curve element, such way that an entity is able to compute point multiplication, but unable to compute the multiplicand given the original and product points.

Given a group *G* and an element $g \in G$, the Discrete Logarithm Problem (DLP) for *G* is: given an element h in the subgroup generated by g find an integer m satisfying $h = g^m$. The smallest integer m satisfying $h = g^m$ is called the logarithm (or index) of h with respect to g, and is denoted $\mathrm{m} = \log_g(h)$ *or* $m = \mathrm{ind}_g(h)$. The best known algorithm to solve the ECDLP is exponential $(O)\sqrt{p}$, which is why elliptic curve groups are used for cryptography .

Elliptic Curves, specifically FIPS P-192 present in SSP, is a plane curve over a finite field which consists of points satisfying the equation:

$E: y^2 = x^3 + ax + b \pmod p,\ a, b \in \mathbb{F}_p$

Elliptic curves are applicable for encryption, digital signatures, pseudo-random generators and other tasks. They are also used in several integer factorization algorithms that have applications in cryptography, such as Lenstra elliptic curve factorization.

NIST recommended 15 elliptic curves, 5 prime curves and 10 binary curves:

- 5 prime $\mathbb{F}_p$ fields for certain primes *p* of sizes 192, 224, 256, 384, and 521 bits. For each of the prime fields, one elliptic curve is recommended [NIST199].
- 5 binary $\mathbb{F}_{2^m}$ fields for *m* equal 163, 233, 283, 409, and 571. For each of the binary fields, one elliptic curve and one Koblitz curve was selected [NIST199].

4. BLUETOOTH ATTACKS

Attacks against Bluetooth devices are unforgiving and not far between, they can target the Bluetooth network infrastructure, the radio signals passing through air, the vulnerable Bluetooth devices within the network due to some poor firmware version, or even the Bluetooth technology itself. Attacks against the network infrastructure is due to the nature of the Bluetooth Piconets, not because the Piconets are weak but because is a mobile ad-hoc wireless network, therefore any attack against an ad-hoc network can be directed also against the Bluetooth network and furthermore any attack against the wireless technology can be applied to the Bluetooth technology. Radio signals are prone to interception and deniability and software designs are prone to human error leaving behind backdoors or weaknesses susceptible for malicious exploitation. Bluetooth attacks are most certainly numerous, however they can be simple or complex, they can function in similarly way or totally different, but eventually they all share a common goal.

Depending on their focus attacks can be filtered easier by their outcome:

- *Disclosure* threats focus on leaking of unauthorized system information to an eavesdropper.
- *Integrity* threats focus on deliberate alteration of information in order to mislead participants.
- *Denial of Service* threats focus on blocking or restricting access to a system to a service.
- *Multi-threats* that combine two or more disclosure and integrity threats.

4.1 Disclosure threats

1. BlueSnarfing attack [Be07] exploits a weakness in some Bluetooth implementations, specifically in OBEX (Object Exchange Protocol) implementation, that allow attackers to connect to a device without alerting its owner and steal information such as phonebooks, notes, calendar, messages. The OBEX protocol was designed to exchange vCards, hence neither authorization nor authentication is required. The attack starts with the connection of an adversary to the OBEX Push Profile, which in term allows him to execute OBEX GET requests for known file names like telecom/pb.vcf containing the phonebook, telecom/cal.vcs containing calendar notes and so on.

2. BlueSnarfing++ [Be07] is an enhancement version of the previous attack, in which the adversary connects to the OBEX FTP server and executes the same get command in order to

subtract files. Since the discovery of the attacks manufacturers have provided users with firmware updates.

3. BlueBugging attack [Be07] exploits a backdoor in the RFCOMM protocol, which allows attackers to execute AT commands, via the AT-parser, to remote control communication devices. RFCOMM is a protocol on top of L2CAP within the Bluetooth protocol stack, which emulates serial RS-232 interfaces via Bluetooth connections. RFCOMM Channels are the "virtual" connections established via RFCOMM. AT Commands stands for *Attention Commands* and are used to access services from GSM/GPRS modems (or SIM cards), like Configurations, SMS Services, MMS Services, Data and Voice links, and most important do not require authentication. Provided that an adversary knows the victims BD_ADDR, he can then connect to the RFCOMM channel 17 and start initiating calls, reading/writing SMS or stealing data. As in most cases manufacturers provided fixes via firmware updates.

4. BlueJacking [Be07] is not an attack that in the sense of exploiting breaches in Bluetooth implementation, but more an abuse of the vCard service in Bluetooth, which allow people to freely exchange business cards. Upon the receipt of a vCard like "You were Bluejacked" some users might draw the conclusion they were infected with some mobile virus. As solution users can turn off Bluetooth connectivity and enable it on use.

5. HeloMoto [Be07] exploits vulnerability in the implementation of the service "trusted devices" present in some Motorola phones. The attacker connects to the OBEX Push Profile, as in BlueSnarf attack, however if this attack is not possible, then he attempts to send a vCard to the target device and immediately cancel the request. In consequence the attacker enters within the "trusted devices" list of the victim, thus allowing him to execute the AT commands present in BlueBug. Motorola has provided a firmware update in order to solve the issue.

6. Offline PIN cracking [ShWo05] is an attack very different than the previous mentioned in terms it is not based on a flawed manufacturer implementation of Bluetooth stack that could easily be fixed with a firmware update, but an attack on the Bluetooth architecture. In order for this attack to succeed an adversary must eavesdrop and log all the communication in pairing protocol. If the pairing was completed long before the attack, an attacker must first issue a re-pairing attack between the victim devices. With all the information available an adversary can now perform a brute force attack on the PIN value. By knowing the *IN_RAND* and *BD_ADDR*

values the attacker can create a list of possible PINs values, and for every entry in the list run the algorithm E_{22} in order to obtain a hypothesis for initialization key K_{init}. Having the K_{init} the attacker can now proceed with decoding messages number 2 and 3 for calculating the hypothesis for linking key K_{ab}. Having the K_{ab} and AU_RAND_A value from message 4, the attacker can now create the hypothesis for *SRES* to be compared with the *SRES* from message 5. If necessary, the attacker can use the value of messages 6 and 7 to re-verify the hypothesis K_{ab} until the correct PIN is found. The attack is described in the following figure:

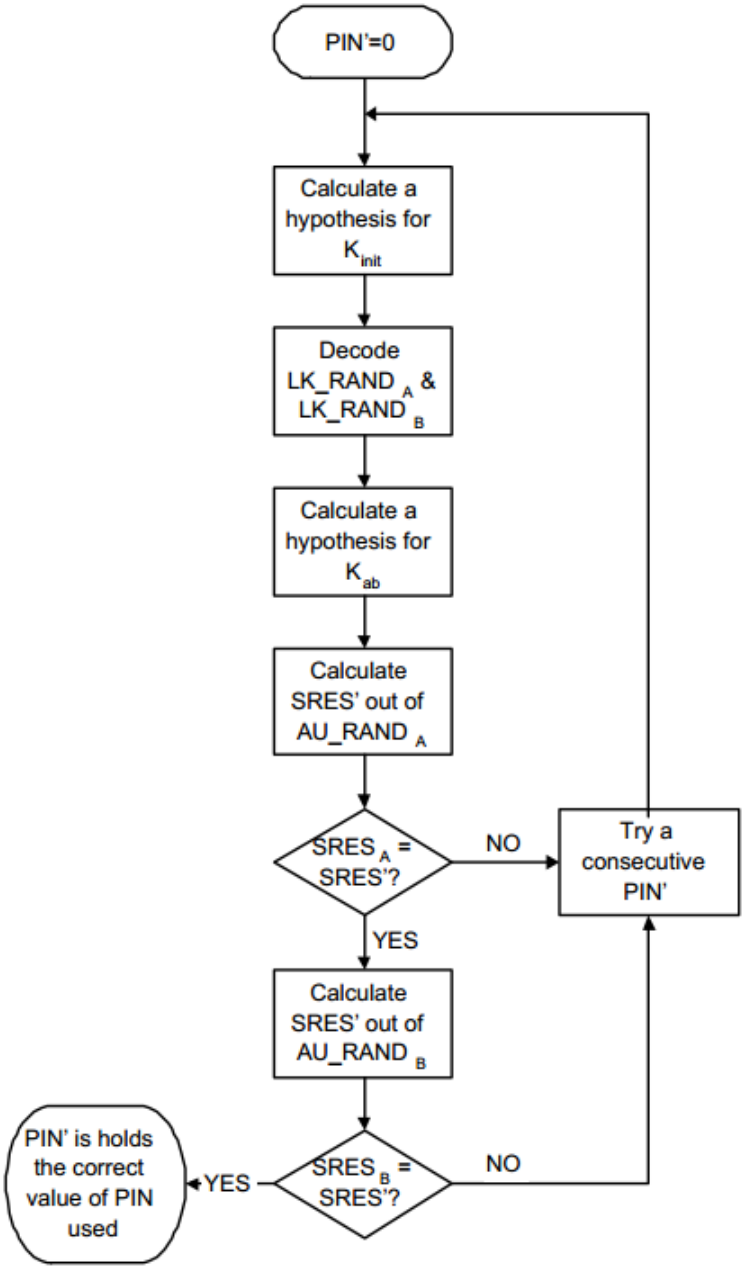

Figure 3.1.1
Offline Pin Cracking attack structure [ShWo05]

Because Bluetooth is a wireless technology, it is impossible to prevent packets from leaking to third parties, therefore the pairing must be done as sparse as possible. Also due to the Bluetooth range limitations (up to 100m) an attacker must be within the proximity of the victims, so the pairing should be done in relatively safe areas avoiding public ones. Although PINs usually measure 4 decimals, Bluetooth allows users to use even longer ones up to 16 decimals.
The engineers *Yaniv Shaked* and *Avishai Wool* have successfully implemented this attack in 2005 on a Pentium IV machine for various PINs length between 4 and 7 decimals. They also proposed some optimizations techniques for the SAFER+ algorithm with results up to 30% in time performance. For additional explanations and results address [ShWo05].

7. Offline Encryption Key Recovery attack [HaHyPaTo13] is an extension of the *Offline PIN cracking* attack and aims to recover the encryption key K_c using the values recorded from in the first attack. Basically an attacker can reconstruct the *ACO* value using the formula $E_1(AU\_RAND_A, BD\_ADDR_B, K_{AB})$. By having the *ACO* the attacker can reconstruct the encryption key with the formula $K_c = E_3(EN\_RAND_A, ACO, K_{AB})$.

8. Brute-Force BD_ADDR scanning [HaHyPaTo13] performs inquires for hidden non-discoverable devices. Because the first 3 bytes of a *BD_ADDR* value are publicly known and fixed by manufacturer, the attack has to focus only on the last 3 bytes. The attack works in the following way: first engineer the *BD_ADDR* for trial, then attempt to create a basic *ACL link* with the remote device using a protocol analyzer. If the connection attempt fails then proceed to the next trial, else perform a remote name inquiry and a disconnection with the target device. An example of such tool is RedFang. Also there exist other implementations and techniques for finding hidden devices, some of them presented in [HaHyPaTo13].

9. BluePrinting attack [HaHyPaTo13] is used to determine the manufacturer, device model, and firmware version of the target device. The purpose of this attack is to determine which devices in range of activity are susceptible for security flaws. This attack can be achieved using BluePrint.

10. Interception of Packets attack was conducted by the authors *Keijo Haataja, Konstantin Hyppönen, Sanna Pasanen, Pekka Toivanen* of the paper [HaHyPaTo13] in order to demonstrate the importance of data encryption and to show how easy it is for an eavesdropper to intercept all packets exchanged via air. In the attack they used a laptop with the LeCroy BTTracer/Trainer v2.2 software connected to the LeCroy BTTracer/Trainer protocol analyzer, in order to analyze

the communication in a cluster of laptops connected via Bluetooth dongles. They setup a piconet master running the chat server and seven piconet slaves running the chat clients. The chat software was implemented on top of RFCOMM, because it is supported in every Bluetooth protocol stack, and applications over RFCOMM are easy to implement. Because the piconet master is in a nonsecure security mode, each of the slaves can establish a connection without authentication, authorization, and encryption, hence the link is unprotected. However an eavesdropper has to synchronize first with the piconet master in order to intercept the packets exchanged via air. The LeCroy's Bluetooth protocol analyzer is able to brief contact the piconet master to extract the *BD_ADDR* of the master and hop sequence information, however it also exposes the eavesdropper identity. Figure 3.3 illustrates the results of intercepted unencrypted packets. Notice the white 16-bit CRC field calculated from the Baseband packet payload also matches the received CRC field. Figure 3.4 illustrates the interception of encrypted packets and the CRC field is now red, that is because the analyzer uses color white to indicate a match between CRC checksums and red to indicate a mismatch.

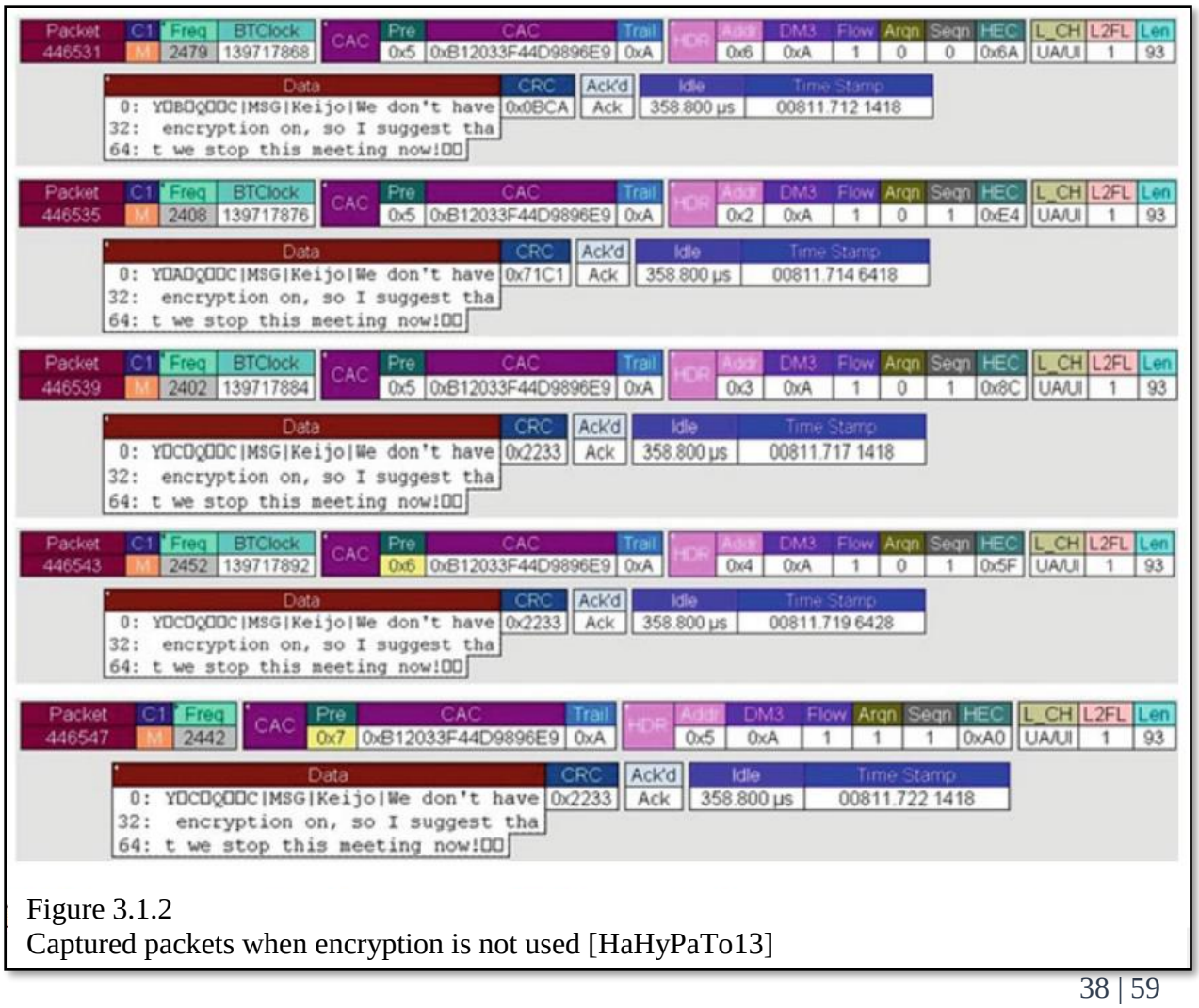

Figure 3.1.2
Captured packets when encryption is not used [HaHyPaTo13]

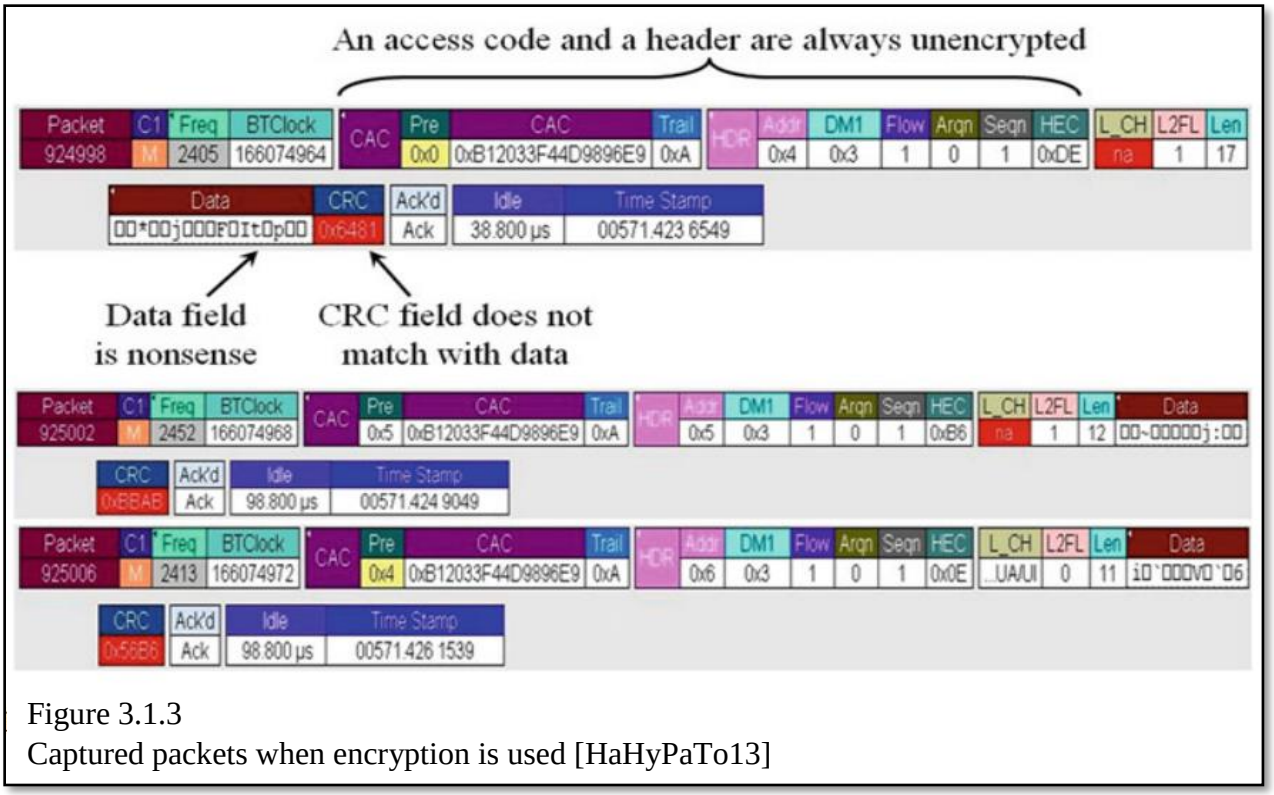

Figure 3.1.3
Captured packets when encryption is used [HaHyPaTo13]

4.2 Integrity threats

1. The re-pairing attack [Be07] is intended to force two paired devices to drop their shared link key K_{AB} and restart the procedure. As mentioned above if two devices have already paired, they will not repeat the process for incoming connections, hence go straight to the authentication procedure. This attack is not dangerous on its own, however provides a key step in success of the *PIN cracking attack,* since the attack requires logging all the pairing messages and chances are the pairing will not occur any time soon.

There are several methods to re-establish the pairing according to [Be07]:

- In the authentication phase the master sends to the slave an *AU_RAND* message and expects to receive the *SRES*. Because the Bluetooth specification allows devices to forget linking keys, an attacker can respond back with a *LMP_NOT_ACCEPTED* message in behalf of the slave. As result the master thinks the slave has lost his linking key and in consequence he also discards his key and re-initializes the pairing with slave. For the success of this attack the packet must be injected before the slave responses.

- In the authentication phase the master sends to the slave an *AU_RAND* message and expects to receive the *SRES*. However if the attacker is able to inject a *IN RAND* message toward the slave, the slave device will be convinced the master has lost the link key and pairing is restarted.
- In the authentication phase the master sends to the slave an *AU_RAND* message and expects to receive the *SRES*. In this moment the attacker can replace the slave's response with a random *SRES* value, hence forcing the authentication phase to restart. If the attacker is able to repeat the alteration for a certain number of failed attempts, the master device is expected to declare that the authentication procedure has failed (implementation dependent) and re-initiate the pairing.

The re-pairing attack is a bit more complicated to execute, because is an active attack that requires packets to be injected at a precise point in the protocol, thus requires special hardware since of-the-shelf Bluetooth dongles do not support such actions. Second is mandatory that Bluetooth specification allows two paired devices to forget their link keys, thus making the attack applicable. Lastly, is required the message to come from the correct *BD_ADDR*, hence the attacker must spoof the source *BD_ADDR* of the message, which in turn requires advanced hardware. And in the last clause a user may become suspicious and refuse to enter the PIN.

2. Reflection attacks [HaHyPaTo13] allow adversaries to relay packets without knowing its content just by impersonating the target devices. These attacks are possible only when the target devices do not hear each other, because they are out of each other's range, and the adversaries have Bluetooth devices with adjustable *BD_ADDRs*, like protocol analyzers. Bluetooth implementations up to 2.0+EDR do not offer any kind of protection against this type of attacks, but in the higher versions Bluetooth implemented Secure Simple Pairing solution.

3. Backdoor attack [HaHyPaTo13] means that an adversary establishes a trusted relationship with the target device through authentication and ensures that this trusted relationship no longer appears to be in the target device's register of authenticated devices. This attack works only on devices vulnerable to this type of threat and the adversary posses their *BD_ADDR*. A list of devices vulnerable to this threat can be found in [HaHyPaTo13], and can be fixed with a firmware update from manufacturers.

4.3 Denial of Service threats

1. BD_ADDR Duplication attack [HaHyPaTo13] is based on the idea that an attacker can place a bug with the victim's *BD_ADDR* in the range of susceptibility in such way that when another Bluetooth device tries to connect to the victim, both the legitimate device and the bug will respond, therefore jamming each other. For most effectiveness the attackers target the piconet masters so that every information passing through the piconet is stomped, thus rendering the network useless.

2. SCO/eSCO attack [HaHyPaTo13] is based on the fact that a real-time two-way voice connection reserves a great amount of bandwidth, in such way that opening a connection of this type with the piconet master ensures that other devices will not be able to access him in reasonable time. For most effectiveness the attackers initiate SCO link with High Quality Voice 1 (HV1).

3. Big Negative Acknowledgement (NAK) attack [HaHyPaTo13] is based on the idea of putting the target device on an endless retransmission loop, in such way that when an attacker requests some information from a device, he responds with NAK, i.e. transmission has failed every time, therefore rendering the device unavailable. This attack requires that the victim is configured to respond to every information request and the adversary knows his *BD_ADDR.*

4. L2CAP Guaranteed Service attack [HaHyPaTo13] is based on the idea that an attacker requests the highest possible data rate, or the smallest possible latency from the target, in such way that all other connections are refused and all throughput is reserved for the attacker. The success of this attack is implementation depended by the L2CAP Quality-of-Service features.

5. Bluetooth OBEX Message attack [HaHyPaTo13] is based on the vulnerability in Nokia 6310i Bluetooth stack that allows attackers to cause the phone to crash or reboot just by sending an invalid Bluetooth OBEX message. Moreover the adversaries must know the victim's *BD_ADDR*.

6. BlueSmack [Be07] is an attack directed to the availability of a Bluetooth device, much like the attack "Ping-of-Death" against IP-based devices. The adversary sends an L2CAP echo request (ping) of large size, approximately 600 bytes, to a Bluetooth device with limited hardware resources. The victim device in consequence reserves an input buffer of fixed length, around 600

bytes, which eventually overflows and generates a segmentation fault, resulting in immediate knock out of the device. This attack can be performed from any linux computer using "bluez-utils" command "12ping –s <num>", where "num" is the packet length.

7. BlueSpamming attack [HaHyPaTo13] is based on the idea that an attacker spams Bluetooth devices with arbitrary files if they support OBEX. *BlueSpam* is an application for Palm OS that performs these attacks. For its success the victim must support OBEX and the attacker must know *BD_ADDR*.

8. Battery Exhaustion attack [HaHyPaTo13] is based on the idea of occupying the target device in such a way that its battery runs out.

4.4 Multi-threats

1. Online PIN Cracking attack [HaHyPaTo13] is based on the idea that an attacker tries to connect with the target device by guessing different PIN values, while changing the source *BD_ADDR* after each failed attempt to bypass the retry delay. This attack is most effective against short or fixed PIN values, i.e. printers, keyboards, headsets. It requires special hardware that can spoof *BD_ADDR* values such as protocol analyzers and a small script on top of Linux "blue-z" library to compute the PIN trial and automate the process for thousands of retries.

2. BTKeylogging attack [HaHyPaTo13] is an extension of the previous attack in conjunction with the *Brute-Force BD_ADDR Scanning attack* in order to crack the keystrokes of a wireless Bluetooth keyboard. The adversary first performs a brute force *BD_ADDR* scanning for the target keyboard, then using either an Online PIN cracking attack or Offline PIN cracking attack against the keyboard's fixed PIN value. Then using a protocol analyzer records all the pairing parameters in order to crack the encryption key K_c. Having the encryption key and its corresponding keystream, an adversary can now decrypt each intercepted keystroke by XORing its payload with the keystream. If successful an attacker can stole emails, credential values and other information using this attack.

3. BTVoiceBugging attack [HaHyPaTo13] is similar to the previous one in which an adversary must first identity the target's *BD_ADDR* value, and then crack the PIN value in order to compute the encryption key necessary to listen the conversation between a Bluetooth headset and phone or

computer. In such cases an attacker can eavesdrop on phone conversations or conference meetings.

4. BTPrinterBugging attack [HaHyPaTo13] is the third type of the bugging attacks and works on the idea of listening the emails or documents sent to a remote Bluetooth enabled printer. In this case is much simpler to crack the PIN, since printers come with default values, usually found in the technical manual, or fixed ones such as "0000" or "1234". For practical cases address document.

5. Cabir [HaHyPaTo13] is a Bluetooth worm running in Symbian mobile phones which support the Series 60 platform. It spreads via messages arriving in the target's inbox as "caribe.sis" file, if the user decides to install it, then the worm will self-activate and auto-scan other Bluetooth devices in its range in order to spread the file "caribe.sis". The purpose of the attack is to spread the worm to as many devices as possible, but it is not intended to damage or steal personal files.

6. Skulls.D [HaHyPaTo13] is a Symbian Installation System (SIS) trojan file that pretends to be a Macromedia Flash Player. The worm spreads itself in similar manner to the *Cabir* worm via messages and device search, however when installed it disables automatically all antivirus and disinfection software, and when user tries to access any application on his phone, the worm will display a flushing skull picture instead.

7. Lasco.A [HaHyPaTo13] is another Symbian Installation System virus file, which spreads in similar manner to the *Cabir* and *Skulls.D*, but in additional is also capable of infiltrating itself in other SIS files in order to be installed alongside other files.

4.5 Man-in-the-middle attacks

1. Bluetooth Nino(No Input No Output) attack [HaHyPaTo13] is a MITM attack on Bluetooth Secure Simple Pairing Numerical Comparison and Passkey Entry models which explores a vulnerability in the IO capabilities exchange phase. Because the exchange is done over an unauthenticated channel which can be controlled by an attacker, therefore the attacker is able to modify the information about capabilities and force the devices to use the association model with the lowest security level, i.e. *Just Works* model (No Screen No Keyboard). It is possible that the victim devices are already paired so an adversary must first issue a re-pairing attack in order to forge the capabilities exchange. However since using *Just Works* model implies the user only to

accept the connection by pushing a button, he may become suspicious in the case where he tries to connect two devices with IO capabilities like two phones, but no numbers were displayed for visual matching, nor neither one of the devices asked to input the displayed number and so might decline the agreement. In the *Nino* attack the authors try to minimize the risk of the user doubtful by cloning both his devices *BD_ADDR* values and user friendly names.

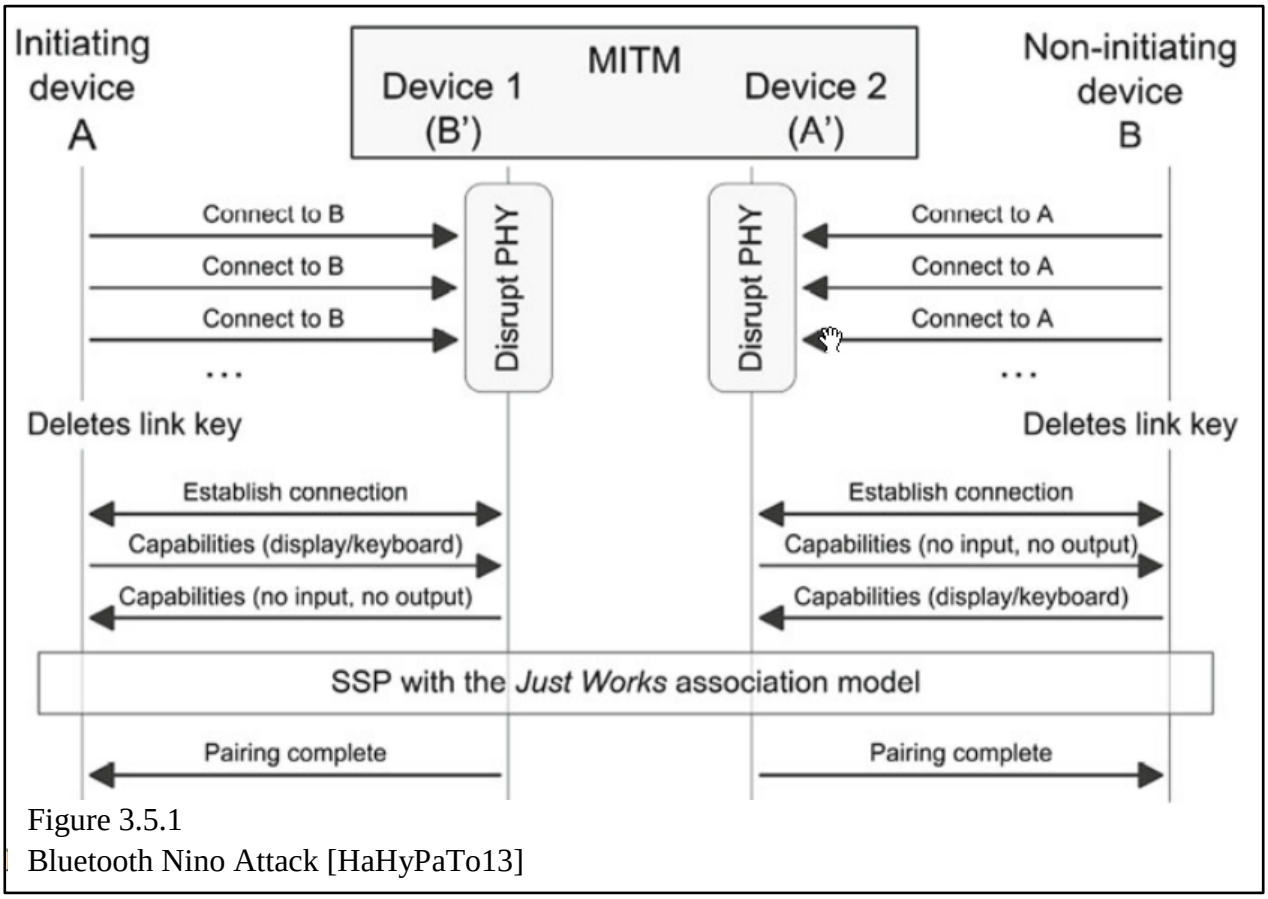

Figure 3.5.1
Bluetooth Nino Attack [HaHyPaTo13]

2. Bluetooth OOB attack [HaHyPaTo13] is a MITM attack on Bluetooth Secure Simple Pairing Out-of-Band association model that requires an attacker to somehow see the victim devices, via a hidden camera or direct line of sight, in order to act before the legitimate users begin the OOB procedure and establish the connection with the spoofed devices to employ the *Just Works* model on the victims.

3. Bluetooth SSP Printer attack [HaHyPaTo13] is a MITM attack on Bluetooth Secure Simple Pairing enabled printers which exploits the fact that in order to improve usability the manufacturers have featured their products with only the *Just Works* association model and will more likely do not require the user to press any button to accept the pairing. However if the

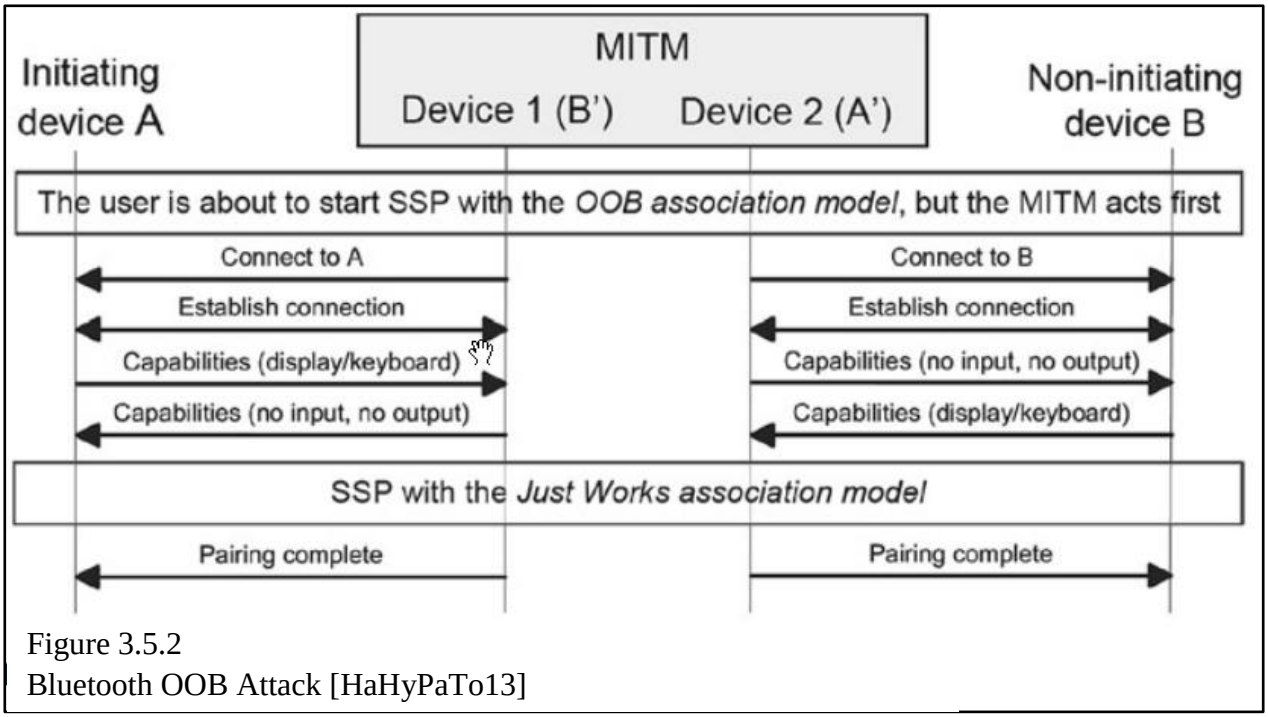

Figure 3.5.2
Bluetooth OOB Attack [HaHyPaTo13]

printer supports MITM protection then the attacker must first issue a capabilities exchange attack. Having these assumptions a MITM can proceed with jamming the legitimate printer in order to disabled any connections with other devices, and impersonate its user friendly name. Therefore, when the user seeks available Bluetooth printers in range, the only printer that is found will be the MITM printer with a different *BD_ADDR* but the same familiar user-friendly name. It is very unlikely that the user will remember the correct *BD_ADDR* and by duplicating also the *BD_ADDR* some collision may occur in the network. In consequence the MITM can now intercept and decrypt all the files send to the MITM printer, and to make it look normally he can then relay the files to the legitimate printer.

4. Bluetooth SSP Headset/Hands-free attack [HaHyPaTo13] is a MITM attack on Bluetooth Secure Simple Pairing enabled headsets and hands-free devices that exploits the fact that most of these devices use the *Just Works* association model in order to make the pairing process user-friendly. First the MITM spoof the legitimate devices *BD_ADDRs* and user-friendly names, and then issue a re-pairing attack, if the devices use other association model, then the MITM also forges the IO capabilities.

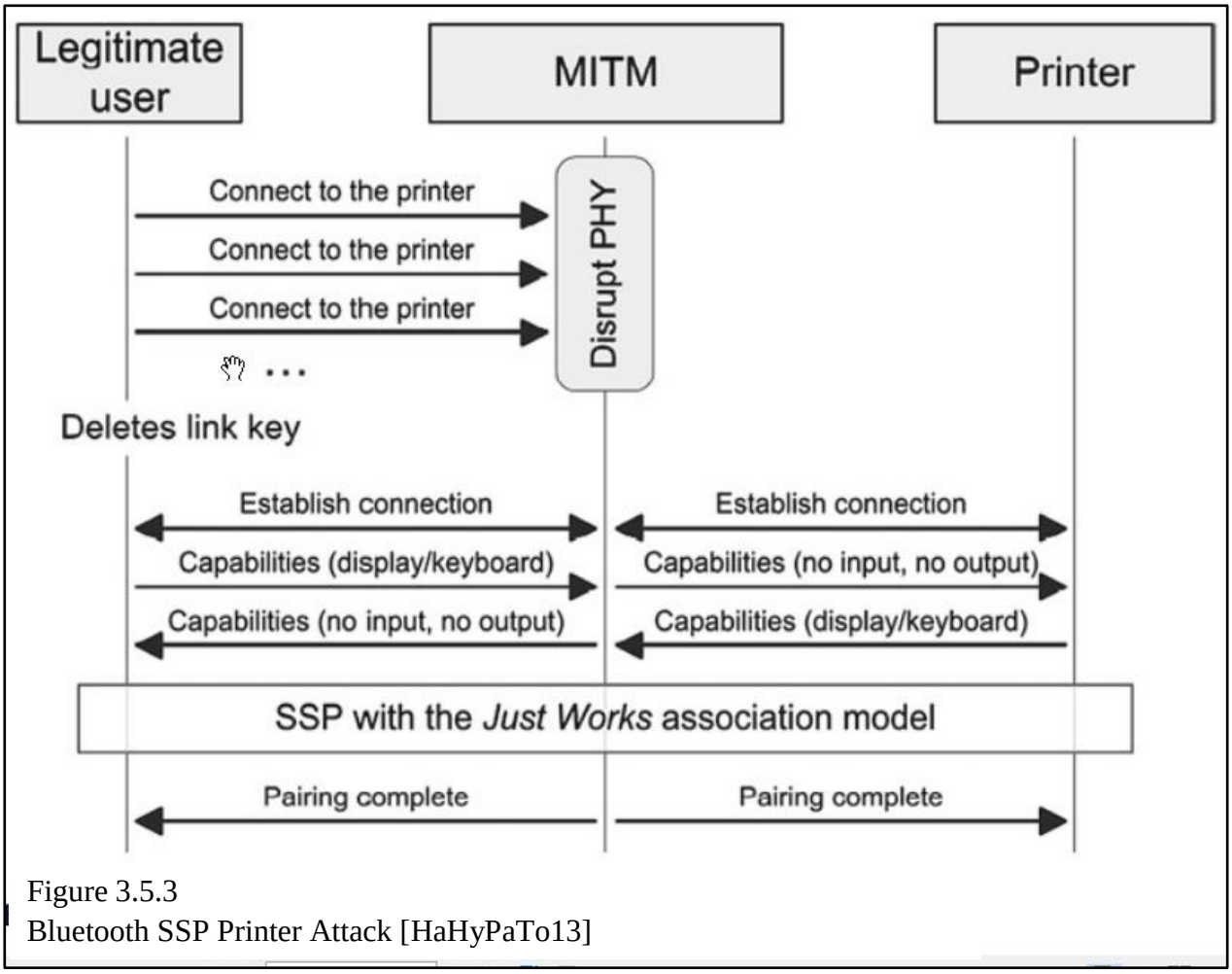

Figure 3.5.3
Bluetooth SSP Printer Attack [HaHyPaTo13]

All the mentioned attacks above only work for the *Just Works* association model so it is imperative to prevent devices from accepting unauthenticated links. This can be accomplished by employing a "security database" in the Bluetooth specification that contains an entry for each service along with the security requirements of that service. In addition Bluetooth device manufacturers can implement an additional window at the user interface level with the message "The second device has no display and keyboard! Is this true?" which can alert the users in case of some wrong doing. Although is not fully possible, Bluetooth should at least recommend manufacturers to make OOB as the mandatory association model to improve the security and usability of SSP. When using the OOB association model it is also required that attackers do not have visual contact with the victims thus a secure environment should be chosen [HaHyPaTo13].

5. IMPLEMENTATION

Simple Secure Pairing introduced in Bluetooth version 2.1+EDR has minimized considerable the amount of vulnerabilities on Bluetooth technology and nowadays devices come with version 4.0 preinstalled. This version is considered very secure both in encryption and replication of information such as no practical attacks ever succeed, however several theoretical attacks have been proposed by researchers during Bluetooth consortiums. In the following chapter is described one use case of these attacks on two victims with Bluetooth 4.0 interacting in a virtual Piconet.

Our attack is based on the idea that in order for two unknown devices to begin the pairing procedure they must first share their Input/Output capabilities through the network in unencrypted manner, hence this list of capabilities can be administered by an intruder in such way that he can convince the legitimate devices that the partner does not have neither keyboard, nor display to handle numbers, therefore the only way they can pair is by just simply accepting the connection (see Just Works association model in chapter 3.3). In this step we hope the intruder to make two connections with each of the legitimate devices and most important to make the victims believe that they are talking directly. From the first connection will result the key to decrypt the information received from the first device and from the second connection the key to re-encrypt this information so that we can pass it to the second device to make the transmission legitimate.

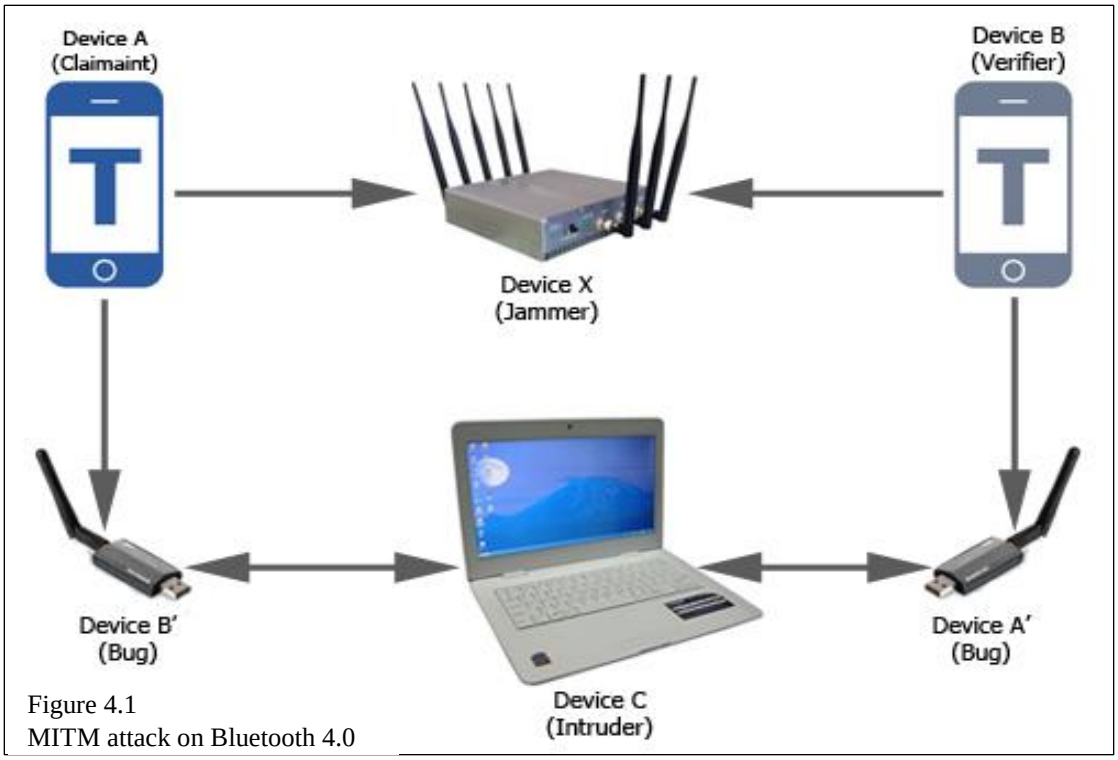

Figure 4.1
MITM attack on Bluetooth 4.0

In the above scenario we introduced two bugging devices, namely A` and B`, with the purpose of duplicating the legitimate devices BD_ADDR value in such way that any incoming packets from device A or device B can be captured and forwarded with the right headers to the destination. Both bugs are connected to a laptop running the software necessary to record and eventual alter the messages. Because wireless antennas broadcast packets through air, all devices within the range will receive these packets, therefore it is mandatory to block the traffic between device A and B and only allow the packets from bugs A` and B`, hence the usage of the jammer. Let's assume that device A sends a request to device B for its capabilities list. First the jammer will block the request from reaching device B, then bug B` will capture this message and forward to device C for analysis. After processing the message will be forwarded to bug A` in order to send it further to device B.

Bluetooth protocol analyzers, such as *LeCroy BTTracer/Trainer*, used by researchers Keijo Haataja, Konstantin Hyppönen, Sanna Pasanen and Pekka Toivanen, provide the hardware and software necessary to sniff traffic, duplicate BD_ADDR values, jam transmissions and broadcast messages in Bluetooth medium, however this analyzers are very hard to acquire and usually cost above 10000 $.

In our practical attacks we setup a virtual piconet on a Windows 7 machine running VirtualBox. In Virtual Box we created two Android 4.4 "Kitkat" guests, one claimant and one verifier, and one Linux Kali 2.0 guest acting as attacker. All three guests have installed the Bluetooth 4.0+LE version and configured to use the Bluetooth interface installed in VirtualBox. On Linux Kali we setup BlueZ libraries and Wireshark software in order to listen the traffic passing through the Bluetooth interface. In the first attack we tried to capture and identify the packets containing the IO capabilities list during the Simple Pairing Procedure between the Android clients. We successfully managed to capture and save the entire procedure in a .pcap file. Next we opened the .pcap file with Wireshark and identified the packets containing "DisplayYesNo" response from both clients but also the number used in matching. In the second attack we tried to capture a simple vcard (electronic business card) file transfer, therefore we selected a vcard file and send it to the paired device. We used Wireshark to capture the entire process. In the post analyses phase we located the packets containing the fields title, tel, adr, email present in a standard vcard.

6. CONCLUSION

This thesis has analzed the Bluetooth technology from four major points of view, first it is important to emphasize the role of Bluetooth devices in our society in order to understand its technical shortcomings in terms of range, level of security and transfer performance. Bluetooth was designed for pocket battery powered devices with low resources operating within a friendzone to facilitate user comfort and replace legacy wired transfers. Having this in mind Bluetooth Special Interest Group designed and released its first generation in 1999, having plenty flaws they continue to improve it reaching the fourth generation in 2016. In the first chapter is presented the entire background history of Bluetooth generations and the main Bluetooth stack components. In the second chapter we focused on Bluetooth security in which we reviewed the Legacy Pairing cryptographic procedures where we talked in detail about the device pairing protocol, the authentication procedure and the encryption in data transfer. Because people have demonstrated quick methods to crack the PIN number, we reviewed the work of Ford-Long Wong, Frank Stajano and Jolyon Clulow repairing the pairing protocol using Elliptic Curve Cryptography. However starting with version 2.1+EDR Bluetooth SIG shifted the entire pairing protocol to Simple Secure Pairing. The new enhanced implementation dropped the usage of PIN numbers in security functions due to the human readability constraints and provided more friendly methods like numeric comparisons or Near Field Communication mechanism and enforced the cryptographic keys used in cipher functions. In the fourth section of the Bluetooth Security chapter we took a depth look into cipher functions *SAFER+*, E_0, *AES,* and *Elliptic Curve Cryptography.* In the third chapter we presented a comparative analysis of most important disclosure attacks, integrity attacks, denial-of-service attacks and combined attacks. Most notable is the *Offline PIN cracking attack* which performs a brute force guess on the PIN value used in a pre-captured pairing session. Although most vulnerabilities were be fixed with the vendor firmware update the above attack is still effective against any devices with Bluetooth version 2.0 inclusive. However we found that researchers have developed a theoretical attack against the Bluetooth Secure Simple Pairing by exploiting a vulnerability in the IO Capabilities exchange phase, that allows them to bypass the MITM protection. This attack has proven unfeasible in practical scenarios but it is open for further investigation.

REFERENCES

[GePeSm04] Christian Gehrmann, Joakim Persson and Ben Smeets
Bluetooth Security
Artech House, Boston, London, 2004

[Ha09] Keijo Haataja
Security Threats and Countermeasures in Bluetooth-Enabled Systems
Department of Computer Science, University of Kuopio, Doctoral Dissertation 2009

[HaHyPaTo] Keijo Haataja, Konstantin Hyppönen, Sanna Pasanen, Pekka Toivanen
Bluetooth Security, Comparative Analysis, Attacks, and Countermeasures
SpringerBriefs in Computer Science, 2013

[WoStCl05] Ford-Long Wong, Frank Stajano and Jolyon Clulow
Repairing the Bluetooth pairing protocol
13th International Workshop, Cambridge, UK, April 20-22, 2005

[ShWo05] Yaniv Shaked and Avishai Wool
Cracking the Bluetooth PIN, 2005
3rd International Conference on Mobile systems, Applications, and Services

[Be07] Andreas Becker
Bluetooth Security and Hacks
Seminararbeit Ruhr-Universitat Bochum, 2007
http://gsyc.es/~anto/ubicuos2/bluetooth_security_and_hacks.pdf

[FIPS197] Federal Information Processing Standards Publication 197
Advanced Encryption Standard, November 2001
http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[SIGWP06] Bluetooth Special Interest Group Core Specification
Simple Pairing Whitepaper , August 2006
https://www.bluetooth.org/docman/handlers/DownloadDoc.ashx?doc_id=86173

[NIST199] National Institute of Standards and Technology
Recommended elliptic curves for federal government use, July 1999
http://csrc.nist.gov/groups/ST/toolkit/documents/dss/NISTReCur.pdf

[SIG10A] Bluetooth Special Interest Group
Bluetooth technical specification 1.0A, July 1999
https://www.bluetooth.org/

[SIG10B] Bluetooth Special Interest Group
Bluetooth technical specification 1.0B, December 1999
https://www.bluetooth.org/

[SIG11] Bluetooth Special Interest Group
Bluetooth technical specification 1.1, February 2001
https://www.bluetooth.org/

[SIG12] Bluetooth Special Interest Group
Bluetooth technical specification 1.2, November 2003
https://www.bluetooth.org/

[SIG20EDR] Bluetooth Special Interest Group
Bluetooth specification core version 2.0 + EDR, November 2004
https://www.bluetooth.com/specifications/adopted-specifications

[SIG21EDR] Bluetooth Special Interest Group
Bluetooth specification core version 2.1 + EDR, July 2007
https://www.bluetooth.com/specifications/adopted-specifications

[SIG30HS] Bluetooth Special Interest Group
Bluetooth specification core version 3.0 + HS, April 2009
https://www.bluetooth.com/specifications/adopted-specifications

[SIG40LE] Bluetooth Special Interest Group
Bluetooth core specification version 4.0, June 2010
https://www.bluetooth.com/specifications/adopted-specifications

[SIG41LE] Bluetooth Special Interest Group
Bluetooth core specification version 4.1, December 2013
https://www.bluetooth.com/specifications/adopted-specifications

[SIG42LE] Bluetooth Special Interest Group
Bluetooth core specification version 4.2, December 2014
https://www.bluetooth.com/specifications/adopted-specifications

[MaKhKu98] J. L. Massey, G. H. Khachatrian, and M. K. Kuregian.
SAFER+. In Proc. First Advanced Encryption Standard Candidate Conference.
National Institute of Standards and Technology (NIST), 1998

APPENDIX

A

ACL	Asynchronous Connection-Less
ACO	Authenticated Ciphering Offset
AES	Advanced Encryption Standard
AFH	Adaptive Frequency Hopping

B

BD_ADDR	Bluetooth Device Address
BER	Bit-Error-Rate
BNEP	Bluetooth Network Encapsulation Protocol

C

CL	Connection-Less
CO	Connection-Oriented

D

dBi	Decibels relative to an isotropic source
dBm	Decibels relative to one milliwatt
DES	Data Encryption Standard
DoS	Denial-of-Service

E

ECC	Elliptic Curve Cryptography
ECDH	Elliptic Curve Diffie-Hellman
EDR	Enhanced Data Rate
eSCO	Extended Synchronous Connection-Oriented

F

FHS	Frequency Hop Synchronization

G

GF	Galois Field
GSM	Global System for Mobile communications

H

HCI	Host Controller Interface
HEC	Header Error Check
HID	Human Interface Devices
HMAC	A keyed-Hash Message Authentication Code
HVI1	High-quality Voice 1

I

IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IMEI	International Mobile Equipment Identity
IO	Input Output
IP	Internet Protocol
IPSec	Internet Protocol Security
IrDA	Infrared Data Association
ISM	Industrial, Scientific, and Medical
IV	Initialization Vector

K

KSA	Key Scheduling Algorithm

L

L2CAP	Logical Link Control and Adaptation Protocol
LAP	Lower Address Part
LC	Link Controller
LM	Link Manager
LMP	Link Manager Protocol

M

MAC	Message Authentication Code
MD5	Message-Digest 5
MITM	Man-In-The-Middle

N

NAK	Negative Acknowledgement
NAP	Nonsignificant Address Part
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
NSA	National Security Agency

O

OBEX	Object Exchange Protocol
OOB	Out-Of-Band

P

PHY	Physical layer
PKI	Public Key Infrastructure
PIN	Personal Identification Number
PL	Path Loss

Q

QoS	Quality-of-Service

R

RAND	Pseudorandom number
RF	Radio Frequency
RFCOMM	Radio Frequency Communication
RSA	Rivest-Shamir-Adleman
RSSI	Received Signal Strength Indicator
RX	Receiver

S

SAFER+	Secure And Fast Encryption Routine +
SCO	Synchronous Connection-Oriented
SDP	Service Discovery Protocol
SHA	Secure Hash Algorithm
SIG	Special Interest Group
SRES	Signed Response
SSH	Secure Shell
SSL	Secure Sockets Layer
SSP	Secure Simple Pairing

T

TX	Transmitter

U

UAP	Upper Address Part
UWB	Ultra-Wideband

W

WEP	Wired Equivalent Privacy
Wi-Fi	Wireless Fidelity
WLAN	Wireless Local Area Network

X

XOR	Exclusive OR