A STUDY OF BLUETOOTH FREQUENCY HOPPING SEQUENCE: MODELING AND A PRACTICAL ATTACK

By

WAHHAB ALBAZRQAOE

A THESIS

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

MASTER OF SCIENCE

Computer Science

2011

ABSTRACT

A STUDY OF BLUETOOTH FREQUENCY HOPPING SEQUENCE: MODELING AND A PRACTICAL ATTACK

By

WAHHAB ALBAZRQAOE

The Bluetooth is a wireless interface that enables electronic devices to establish short-range, ad-hoc wireless connections. This kind of short-range wireless networking is known as Wireless Personal Area Networks (WPAN). Because of its attractive features of small size, low cost, and low power, Bluetooth gains a world wide usage. It is embedded in many portable computing devices and considered as a good replacement for local wire connections. Since wireless data is inherently exposed to eavesdropping, the security and confidentiality is a central issue for wireless standard as well as Bluetooth. To maintain security and confidentiality of wireless packets, the Bluetooth system mainly relies on the Frequency Hopping mechanism to equivocate an adversary. By this technique, a wireless channel is accessed for transmitting a packet. For each wireless packet, a single channel is selected in a pseudo random way. This kind of randomness in channel selection makes it difficult for an eavesdropped to predict the next channel to be accessed. Hence, capturing Bluetooth wireless packets is a challenge. In this work, we investigate the Frequency Hopping sequence and specifically the hop selection kernel. We analyze the operation of the kernel hardware by partitioning it into three parts. Based on this modeling, we propose an attacking method for the hop selection kernel. The proposed method shows how to expose the clock value hidden in the kernel. This helps to predict Bluetooth hopping sequence and, hence, capturing Bluetooth wireless packet is possible.

ACKNOWLEDGMENTS

First, I would like to express my thankfulness to Allah: "And say: 'All the praises and thanks be to Allah.'" The Quran (17:111).

I also would thank the support and the scholarship of my country Iraq.

I would like to express my gratitude to my advisor Dr. Guoliang Xing for his guidance, inspiration and encouragement. Without his invaluable advice and support, this work would not be possible.

I am grateful to my friend Jun Huang. In particular, I would like to thank Jun for coming up the idea of using jamming signals to crack Bluetooth hopping sequence. I am thankful to Jun for weekly meetings with me, helping me with operating the USPR devices and Ubuntu setup, and even linguistic corrections in this thesis. I would like to thank him for taking time to exchange ideas with me. His kindness, support, and suggestions are all unforgotten.

I am thankful to my friend Assaad AlSahlani. His support and encouragement are appreciated.

I would like to thank the Iraqi Cultural Office in Washington D.C represented by Prof, Abdulhadi Al Khalili who has been supporting me in this scholarship.

A special thanks to my family. Their encouragement and support through my study are invaluable.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

Introduction

The success of portable consumers products such as laptops, notebooks, PADs, cell phones, and many portable devices is based on continuous success of cost and size reduction of their microelectronics circuit. Along with the advance of mobile technology, there is an increasing demand on sharing documents, music and multimedia files between users within a small area. Having these devices communicating among themselves within small area is called Personal Area Networks (PAN). It is obvious that wiring portable devices is not an efficient solution. An alternative solution is to employ wireless links. Therefore, many wireless standards have been developed for enabling personal area networking over short ranges wireless connection, among them Bluetooth is a tone of the most successful technology.

The Bluetooth wireless communication technology is defined under the IEEE 802.15.1 Standard [1]. Because of its attractive features, the Bluetooth is considered an efficient solution for wireless personal area networking. The Bluetooth is designed to support small-size, low cost, low power consumption wireless links. For example, a Bluetooth USB adapter does exceed the size of the "W" button on a standard computer keyboard. In terms of power consumption, to establish a wireless link within 10 meters of range, the Bluetooth adapter only incurs a power consumption ranging from 0.25 up to 2.5 mWatt. The retail price of the Bluetooth USB adapter is about 0.01$ in the market. Bluetooth radio system works in the

unlicensed Industrial, Scientific, and Medical (ISM) band at 2.4 GHz.

Because of these attractive features, we can see a world wide usage of the Bluetooth devices. Bluetooth technology is embedded in most of portable electronic devices. According to the Bluetooth Special Interest Group (SIG), the number of Bluetooth enabled devices was about 1 billion devices in 2006 with a shipping rate of 10 million device per week. During 2008, 2 billion products had been shipped with Bluetooth enabled wireless interface [2]. Today, many local wired connections such: headset of cell phones, printers, computer mouse and keyboard can be replaced with Bluetooth wireless links. And this what makes it an important electronic technology of human modern life. However, as wireless data is inherently exposed to eavesdropping, the confidentiality is a central issue for all wireless networks as well as Bluetooth wireless network. This is because sensitive data, such as voice data, secret documents, and/or private pictures, is often carried over by wireless links.

The Bluetooth system basically relies on two techniques in order to keep confidentiality of transmitted data over wireless links, which are:

(a) Encryption: the Bluetooth employs a stream cipher, known as E_0, to encrypt wireless data.

(b) Frequency Hopping: by employing this technique, the Bluetooth radio system is continuously switching between wireless channels. This will be discussed in more details throughout this work.

Regarding the Bluetooth stream cipher E_0, the cryptanalysis of this encryption is studied in the literature. There are many researches, including [7, 13], proposed attacking Bluetooth stream cipher. The second kind of defence is achieved by the Frequency Hopping technique. This technique is originally proposed for avoiding performance downgrading due to noisy channels. The Frequency Hopping is the main security fence that users rely on for data confidentiality. The challenge for an attacker is to predict the next channel of a Bluetooth link, so that it can tune its radio to follow the link for eavesdropping. The Frequency Hopping

mechanism is the main topic of this work. By analysing the Hop Selection kernel, we found a security threat that can be exploited to predict Bluetooth hopping sequence. Furthermore, we propose an attacking method for the Hop Selection kernel.

1.1 Related Work

This section can be mainly divided into two topics: in the first topic, the cryptanalysis of the Bluetooth stream cipher is reviewed. While in the second topic, the Bluetooth Frequency Hopping and related researches are reviewed.

First: The Bluetooth stream cipher E_0

Despite the cryptanalysis of the E_0 stream cipher is beyond the thesis scope, it is useful to review some related works. The E_0 stream cipher is designed specifically for Bluetooth [7]. Cracking the E_0 stream cipher is studied by Levy et al [13] who proposed a uniform framework for cryptanalysis of the Bluetooth E_0 cipher. This method assumes that the E_0 is employed outside the Bluetooth environment. In other words, the encrypted stream of data is not affected by the Frequency Hopping mechanism of Bluetooth. Lu et al [14] proposed a practical attack on Bluetooth encryption; the author assumes that an attacker can observe the stream of encrypted data which implies the stream is outside the effect of Frequency Hopping. Shaked et al [7] proposed a method for cracking the Bluetooth PIN (Personal Identification Number). The PIN can be used for decrypting Bluetooth wireless packet, in case of being encrypted. This method requires that an attacker witnesses the pairing process of two, or more, Bluetooth devices. Because this is not practical condition in most cases, the author proposed some method to enforce already paired Bluetooth devices to break their connection and proceed with a repairing process.

Second: Frequency Hopping

The Frequency Hopping technique is originally proposed to improve Bluetooth performance by avoiding noisy channels. Since the Bluetooth operates in the free unlicensed ISM band,

there is a probability to face some noised channels by other wireless radios in the ISM band. The Frequency Hopping technique is a potential security feature because of the difficulty of capturing Bluetooth wireless packets. Capturing a wireless packet requires an attacker to predict the next channel to be employed by Bluetooth. In a piconet setup, the MAC address and the clock of the piconet master are utilized to generate pseudo random hopping sequence, where the hopping sequence is a string of pseudo random numbers.

With respect the Frequency Hopping, Spill et all [6] proposed a mechanism for extracting the MAC address and the first 6 bits of the device clock of the targeted piconet mater device. In order to hop along with Bluetooth, All the bits of the device clock, except the first bit, along with the device MAC address bits are required to be known. The first 6 bits of the clock are not enough. Hence, there is a missing part which is the rest bits of the clock CLK_{25-7}. In this work, we base on the results of Spill [6] in order to expose the missing part of the clock. By this way, we will be able to hop just like the targeted Bluetooth.

Some of the proposed attacks target the pairing process of the Bluetooth. Sensitive information is exchanged through the pairing process of Bluetooth devices. This includes information about the hopping sequence synchronization, encryption, encryption keys, and so on. However, listening to the pairing process in not possible all the time. Hence, the authors proposed breaking an established links in order to force users to proceed with pairing procedure again. This kind of attack also assumes that wireless packets are not affected by the Frequency Hopping mechanism and can be captured by an attacker. Among these researches, Haataja et al [9, 10, 11] proposed Man-In-The-Middle kind of attack on Bluetooth. This can be achieved by listening to the pairing process which is often impractical to be achieved at early communication stages. Therefore, the author proposed several methods for breaking an established connection. This can be conducted by jamming wireless connections. Another way is sending some counterfeiter packets to legitimate users so they might think something wrong going on and initiates the repairing process.

1.2 Motivation

The Bluetooth wireless network plays an important role in today's computing world. This wireless technology system relies on the Frequency Hopping technique to maintain the confidentiality of its wireless data since Bluetooth encryption can not be trusted any more. Bluetooth employs a special hardware for hopping sequence generation. This hardware, which is called the *Hop Selection kernel*, is basically a kind of pseudo-random number generator device. Now imagine what could happen if an eavesdropper captures some Bluetooth wireless packets transmitted from a Bluetooth enabled wireless keyboard to a PC when you are accessing a bank account? In this case, the password of your bank account is vulnerable and may be exposed. Another example is the leakage of secrets documents when being transmitted to a Bluetooth enabled wireless printer. This kind of attack is called "passive attack". On the other hand, an active attacker can fake sensitive requests and send them to a computer machine on behalf of its Bluetooth enabled wireless keyboard. Therefore, we found that investigate the security threats of Bluetooth is important; and specially the Hop Selection Kernel as it plays a central role in Bluetooth security.

1.3 Thesis Contribution

In this work, we first propose modeling the Hop Selection kernel by dividing it into three parts. This kind of partitioning is important to understand the function of the kernel by studying the role of each part individually, and this is the basis for the attacking method. Second, we propose a systematic attacking procedure against the Hop Selection kernel of Bluetooth. The proposed attacking method is based on examining the design details of the Frequency Hopping kernel and analyzing the functionality of its consisting hardware components. The analysis of the kernel, which we present in this work, is suitable to expose weak points that can be exploited by an attacker against the Hop Selection Kernel.

In later chapters of the thesis, we introduce useful definitions and background about the

Bluetooth radio system. This will be presented in Chapter 2 where we will review important technologies used by Bluetooth for channel bandwidth spacing as well as overview of the Hop Selection Kernel. This includes defining the essential parameters utilized for determining the hopping sequence. These parameters are the clock and MAC address of the Bluetooth device. The network topology will also be considered in Chapter 2 as it necessary for understanding the formation of Wireless Personal Area Network by Bluetooth devices. This is known as *Piconet*.

In chapter 3, we mainly focus on analyzing the functionality of the Hop Selection kernel by examining its hardware components. We analyze the Hop Selection kernel by dividing its function into three components or parts: A, B, and C. Then, we analyse each part separately and find a special mathematical expression that rules its behaviour. Based on the analysis of chapter 3, we introduced the proposed attacking method in chapter 4. The chapter includes two main topics: in the first topic, we present the theoretical analysis of the proposed attacking method. The method suggests manipulating the kernel parts: A, B, and C individually. In the second topic, we present the simulation software of the attacking method using Matlab environment. The implementation challenges are discussed as well. Chapter 5 has two sections: in the first section we derive the conclusion of the thesis and emphasize the results that we obtained in this work. While in the second section we will be talking about the future works and directions of the thesis. This includes adopting the attacking procedure on experimental devices such as the USRP and commercial hardware devices such as the Ubertooth.

Chapter 2

Background

The Bluetooth is a universal radio system that enables electronic devices to establish short-range wireless connections. This wireless technology adopts packet-based communication scheme which is similar to other standards of data communication systems such as WiFi and BigZee. Another point of similarity to other standards is utilization of the free unlicensed ISM 2.4 GHz band. However, the Bluetooth system is characterized by many aspects that make it different from others. One of these features is the Frequency Hopping technique. The Bluetooth system adopts the Frequency Hopping for accessing wireless channels. The hopping sequence is determined based on the Bluetooth MAC address and clock. The hopping sequence selection is achieved by the Hop Selection kernel. In this chapter, we review some important characteristics about the Bluetooth wireless system. Furthermore, we discuss definitions of the hop selection kernel and its input parameters.

This chapter is organized as follows: in section 2.1, a top view of the Bluetooth Radio will be presented; where we focus on the radio spectrum and the packet-based communication scheme. In section 2.2, the Hop Selection Kernel is presented along with the definition of two important inputs for the kernel, that are: the device identity, or the device MAC address, and the Bluetooth device clock. The Adaptive Frequency Hopping (AFH) will also be reviewed as well. In section 2.3, the Bluetooth networking topology is discussed with the

pairing process.

2.1 Bluetooth Radio System: A Top View

This section presents the general characteristics of the Bluetooth radio system.

2.1.1 Bluetooth Radio Spectrum

In order to enable a world wide usability of the Bluetooth technology, the radio spectrum must be open and can be used by public without the need for licenses. Toward this end, the free unlicensed spectrum of the ISM Band was the choice for Bluetooth. This band, the Industrial, Scientific, and Medical, is globally available and this enables users from connecting their portable devices wherever they travel. In most of the countries, including the United States and Europe, the ISM band ranges from 2400 MHz up to 2483.5 MHz [3].

The freedom and world wide availability of the ISM band have some negative side effect as well. The side effect of using the free ISM band is possibility of facing noisy channels. This noise is simply caused by other wireless radios operating in the same frequency of the Bluetooth radio on the same time slot. Therefore, a significant amount of interference might be added by those radios affecting the Bluetooth wireless link throughput or performance downgrading. Some of the options available for the Bluetooth radio to mitigate the interference effects are: interference suppression and avoidance. Interference avoidance is the most attractive solution since suppression requires adding more components and circuitry to deal with unwanted frequencies. Avoidance can be achieved in time and/or frequency. The time avoidance is suitable in the case of a pulsed jamming source of interference; while avoidance in frequency is more practical solution. This is because the free bandwidth offered through the ISM band is about 80 MHz and most radio systems are band-limited which increases the probability of finding a part of the ISM spectrum where there is no dominating interference [3].

This reason and a number of properties make the Frequency Hopping (FH) is the best channel accessing technique for the Bluetooth radio. The Bluetooth radio system defines 79 hop carries within the ISM band. Each carrier is spaced by 1 MHz of channel bandwidth. The following equation rules this kind of division:

$$f = 2402 + K \text{ MHz}, \quad k = 0, 1, ..., 78 \tag{2.1}$$

Where f, which is measured in MHz, represents the channel frequency that will be used for transmission. The parameter K is an index that is used to specify one channel among the 79 channels. Determining the value of the index K is the responsibility of a hardware unit called *The Hop Selection kernel*. The analysis of this hardware component is the main topic of chapter 3.

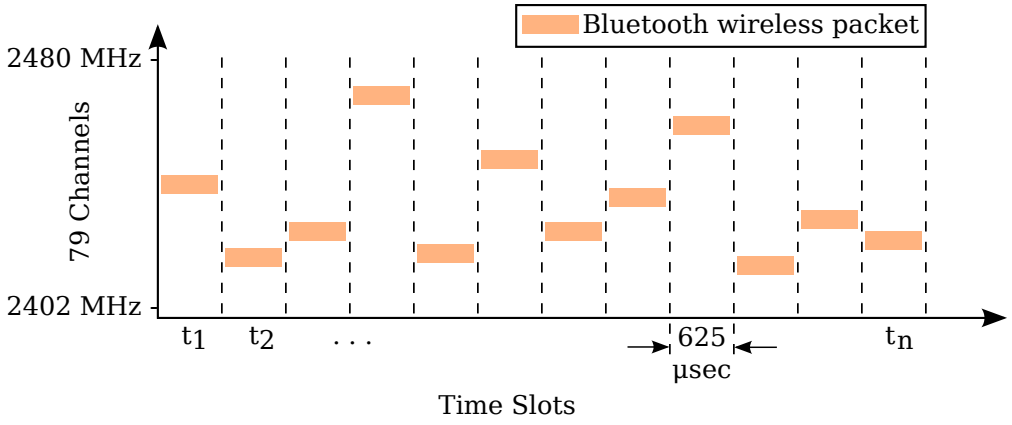

Figure 2.1: Transmission of Bluetooth wireless packets over multiple channels. For interpretation of the references to color in this and all other figures, the reader is referred to the electronic version of this thesis.

For the Hop Selection kernel, coordination of hopping sequence is not allowed by the FCC rules. Instead of this, a large number of pseudo-random hopping sequence is defined. The kernel hardware selects a particular hopping sequence based on the MAC address of the Bluetooth device. The Bluetooth clock is used to determine the phase in the hopping sequence. The nominal dwell time of utilizing a hopping channel is defined to be 625 μ second. Therefore, the Bluetooth Hop Selection kernel generates about 1600 hops per second that are

selected in a pseudo-random fashion. Based on this pseudo-random scheme of hop selection, the Bluetooth wireless packets between a sender and a receiver can be illustrated as in Fig. 2.1. As a counterexample of using the Frequency Hopping Scheme, Figure 2.2, illustrates transmitting wireless packet over a single channel as an opposite option to the frequency hopping over 79 channels:

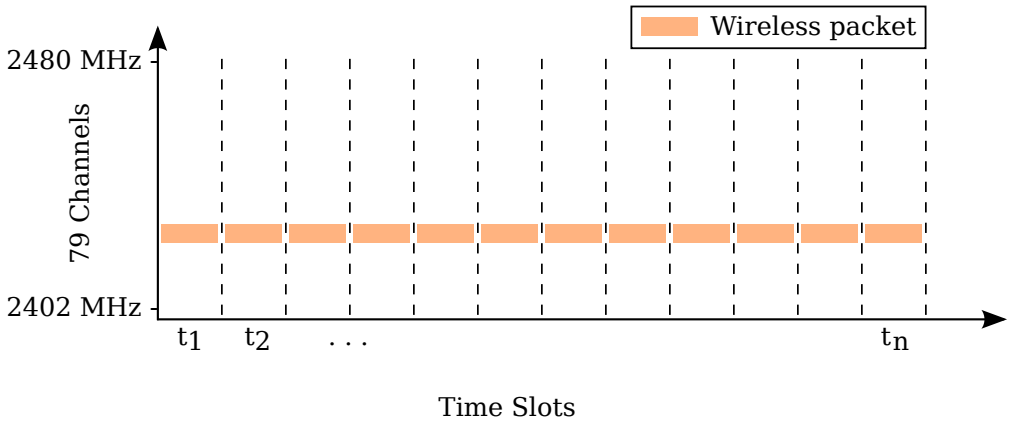

Figure 2.2: Transmission of wireless packets over single channel.

2.1.2 Bluetooth Communications: Packet Based System

The Bluetooth wireless system employs a packet-based communication scheme. The bit stream that is ready for transmission is fragmented into packets. There are 16 packets types are defined by the Bluetooth standard. All packet types share the same format except the payload portion which varies from type to type. The general Bluetooth packet format is illustrated in Fig. 2.3.

72 bits	54 bits	0-2745 bits
Access Code	Header	Payload

Figure 2.3: The Bluetooth wireless packet format.

On each wireless channel, a single wireless packet can be transmitted. For the next packet, a new wireless channel is selected in a pseudo random way. Packets sent by one

party should be confirmed upon receiving by the other party. This is known as Acknowledgement/Negative Acknowledgement (ACK/NACK) in terms of networking. It is defined by the Bluetooth designer that even time slots are used by the *piconet master*. While, odd time slots are used by slaves Bluetooth devices participating in the piconet. By this way, the Transmission/Receiving timing is maintained [3]. Bluetooth network topology is discussed in section 2.3.

2.2 Hop Selection Kernel

As discussed in the previous section, the Frequency Hopping scheme was originally proposed to mitigate interference effects caused by co-users sharing the unlicensed ISM band, such as WiFi and BigZee. In this channel accessing scheme, a single channel is selected for a specific amount of time then another channel is utilized. This rapid switching technique among relatively large number of narrow bandwidth channels adds an important security feature to the Bluetooth radio system. By its hopping over wireless channels, Bluetooth presents a capability of equivocating. This capability makes it difficult for an attacker to sniff Bluetooth wireless packets. There is no doubt that monitoring all wireless channels defined in the ISM band is not a practical solution. The other way to sniff Bluetooth wireless packets is to predict the next wireless channel to be accessed by Bluetooth. This is the main topic of this work and we investigate the possibility of this kind of prediction.

2.2.1 Design Principles

The Hop Selection kernel applies a special selection mechanism for the hopping sequence. The kernel is designed to meet the following principles:

(a) The kernel selects the hopping sequence based on the device identity, which is the MAC address, and the device clock.

(b) The selected hopping sequence is allowed to be repeated every 23 hours.

(c) For every 32 consecutive hops should span over 64 MHz of the defined spectrum.

(d) The number of hopping sequences should be very large.

(e) All frequencies should be used with equal probability, in average.

(f) Any change in the clock and/or the MAC address selection mechanism should instantaneously be changed.

These design requirements are necessary for suitable operation in terms of the Frequency Hopping.

2.2.2 Unit Inputs

As it is mentioned previously, the Hop Selection kernel simply controls the frequency hopping using the index K. Adjusting this index to a new value is translated as switching from one channel to another by the Bluetooth radio, i.e., selecting a new frequency for transmission. A top view of the Hop Selection kernel is illustrated in Fig. 2.4.

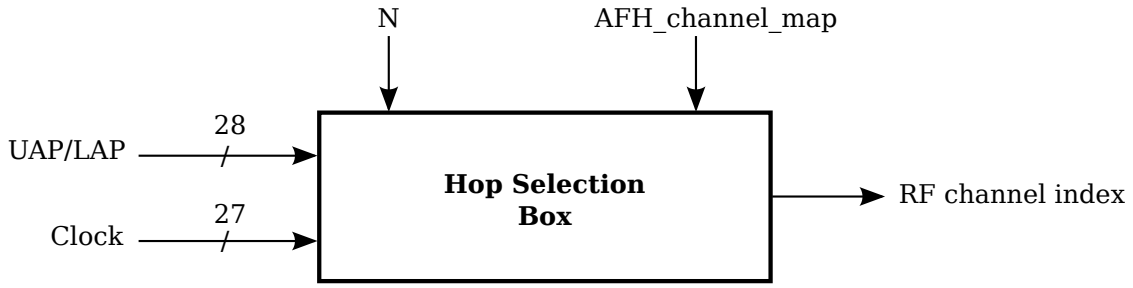

Figure 2.4: The Hop Selection Box of Bluetooth, [1].

For the Hop Selection kernel, a large number of hopping sequences is defined. The selection kernel determines a single hopping sequence based on the value of MAC address and the clock of the Bluetooth device. The input parameters of the Hop Selection kernel, or Hope Selection Box, are as follows:

(a) CLOCK, this is the device digital clock.

(b) UAP/LAP, which is simply the device MAC address.

(c) AFH_channel_map, this input is a table contains the status of each wireless channel. these parameters are related to the Adaptive Frequency Hopping mechanism which is discussed in section 2.2.5.

(d) N, is also provided by the AFH technique. This input parameter is basically an integer represents the number of good channels that can be used by the Bluetooth radio.

The following three subsections are dedicated to discuss these important input parameters of the Hop Selection kernel.

2.2.3 Bluetooth identity: the device MAC address

In the networking world, standards impose assigning a special ID number for each device participating in the network; this kind of IDs is called *The device MAC address*. The same standard is defined for the Bluetooth wireless system. The device MAC address should be unique over the network. To be more specific, sharing the MAC address by two, or more, Bluetooth devices is not allowed. The MAC address of Bluetooth device consists of 48 binary bits binary. The MAC address binary number is composed of three parts as illustrated in Fig. 2.5. These three parts are as follows:

LSB MSB

company_assigned						company_id					
LAP						UAP		NAP			
0000	0001	0000	0000	0000	0000	0001	0010	0111	1011	0011	0101

Figure 2.5: The standard format of the Bluetooth MAC address.

(1) The Lower Address Part LAP: this part consists of 24 bits.

(2) The Upper Address Part UAP: this part has 8 bits only.

(3) The Non-significant Address Part NAP: this part holds the rest of the 48 bits, which are 16 bits.

2.2.4 Digital Clock of the Bluetooth Device

The Blutooth clock is basically a kind of free-running digital counter device. Such kind of clocks is different from many known clocks because it not related to the real time of day. This means the Bluetooth clock can be initialized with any value [1]. The Bluetooth clock counter consists of 28 bits, hence, it is capable of counting up to $(2^{28} - 1)$, as shown in Fig. 2.6. This counter is designed in such a way that its counting sequence can be repeated every 23 hours. The Bluetooth clock rate is 3.2 kHz [1]. Therefore, the Least Significant Bit (LSB) of this counter ticks, or flips, every 312.5 μ second.

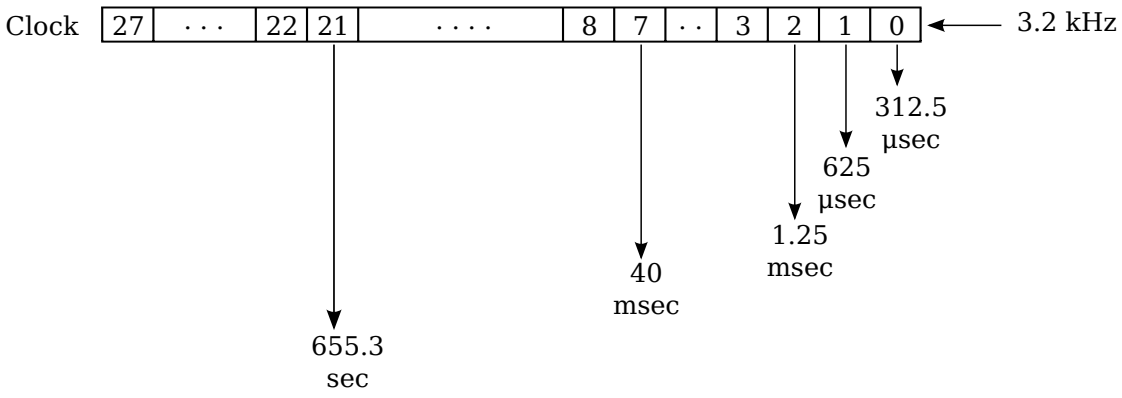

Figure 2.6: The timing of Bluetooth system clock.

Figure 2.6 illustrates a basic schematic digram of Bluetooth clock counter with some important bits and their flipping rates. The Bluetooth clock is an essential input parameter with respect to the Hop Selection kernel. It is important to mention that the Least Significant Bit of the clock, namely CLK_0, is not involved in the operation of the Hop Selection kernel.

2.2.5 Adaptive Hop Selection Facility

The Adaptive Frequency Hopping (AFH) technique was proposed to improve the Bluetooth performance by avoiding transmission over noisy channels. Because the Bluetooth operates in the free unlicensed ISM spectrum, there might be some other wireless sources sharing the same frequency of Bluetooth in the same time period. Therefore, some of the 79 channels can be noised and should be avoided by Bluetooth to maintain its throughput. The adaptive Frequency Hopping utilizes some measurements techniques to detect noisy channels and decide which one is "Good" and which one is "Bad". Good channels can be used by Bluetooth while bad channels should be avoided. These mechanism of detecting the status of wireless channels is beyond the scope of this work. In general, the AFH technique provides the Hop Selection kernel with two essential input parameters, which are:

1. The AFH_channel_map: is basically a string of 79 cells, where a cell consists of 2-bit. Each cell holds the channel classification report of the corresponding channel. The classification includes one of the following status: Good, Bad, Unknown. Based on the provided report, or the status of each channel, the Hop Selection kernel can decide to use the channel or not.

2. The parameter N: this parameter represents the number of Good channels that can be used by the Bluetooth wireless system.

It is obvious that within noisy environment, the number of usable channels will be less that 79 channels. The minimum number of channels that is allowed for Bluetooth to operate with is 20 channels. This is the minimum requirement for normal operation of the Bluetooth wireless system. Thus, we can express the parameter N in terms of the allowed range as follows:

$$20 \leq N \leq 79 \tag{2.2}$$

The value of the input parameter N plays an important role in the operation of the Hop Selection kernel. In the next two chapters, we will further analyse the effect of this parameter on the operation of the Hop Selection kernel.

2.3 Bluetooth Network Topology

The network topology of the Bluetooth wireless network is a decentralized kind of networking known as the *Ad Hoc* wireless networking. This scheme is recognizable aspect of the Bluetooth system and it is required by the IEEE standard of Wireless Personal Area Networks as follows[1]: *"The functions and services required by an IEEE 802.15.1-2005 device to operate within ad hoc networks."*

In general, wireless networking systems can be classified as centralized and decentralized networking. In the centralized type, the wireless network requires a pre-existing infrastructure in order to establish the network connection. As an example of such kind of networks: a group of routers that are connected by a wired network, a mobile network established by one or more central nodes, called base stations, where nodes rely on a wired backbone infrastructure, and even a WiFi access points connected via a wired or a wireless infrastructure can be a good examples of that. It is obvious that the central node, i.e. the base station, is strictly distinguishable from other nodes in the network, which are called terminals.

The opposed to this networking scheme is the decentralized networking scheme. In such networking scheme, connected devices don't rely on a pre-existing infrastructure. The ad-hoc wireless network can be classified as a decentralized type of wireless network. In the decentralized networking system there is no distinguishing features between the participating nodes. In other words, nodes are not classified as base station or terminals. The connectivity style of wireless ad-hoc network relies on peer to peer communication, with no need for wired infrastructure. The Bluetooth wireless adopts the ad-hoc network topology. The Bluetooth ad-hos networking futures can be summarized as follows [3]:

- **Infrastructure**: there is no underling infrastructure, such as wired network, to support the connectivity between portable wireless Bluetooth units.

- **Central node**: this kinds of nodes is not defined in the Bluetooth ad-hoc Network.

- **Coordination of communication**: there is no such kind of coordination for an Ad Hoc network. In other words, many Bluetooth ad-hoc networks can exist in the same area at the same time.

With respect to Bluetooth, the IEEE standard states that an ad-hoc Wireless Personal Area Network (WPAN) can be formed in a spontaneous manner by a number of Bluetooth devices, typically from 2 up to 8 devices. And this kind of networks should require no infrastructure and is limited in temporal and spatial extent [1]. This formation of Bluetooth devices is called *piconet*, which is basically a personal area network of Bleutooth devices. One of the Bluetooth devices participating in a piconet should be the *piconet master*, which is the one that initiates the PAN formation. All other participating devices are slaves. Moreover, a piconet master device controls Hopping sequence pattern of the whole piconet. This is necessary in order to make all slave devices associated with a piconet hop just like the piconet master. By this way, the slave devices can capture wireless packets sent by the master device [3]. More details about hopping sequence selection mechanism will be discussed in chapter 3. Figure fig:piconet1 illustrates the formation of Bluetooth piconet network.

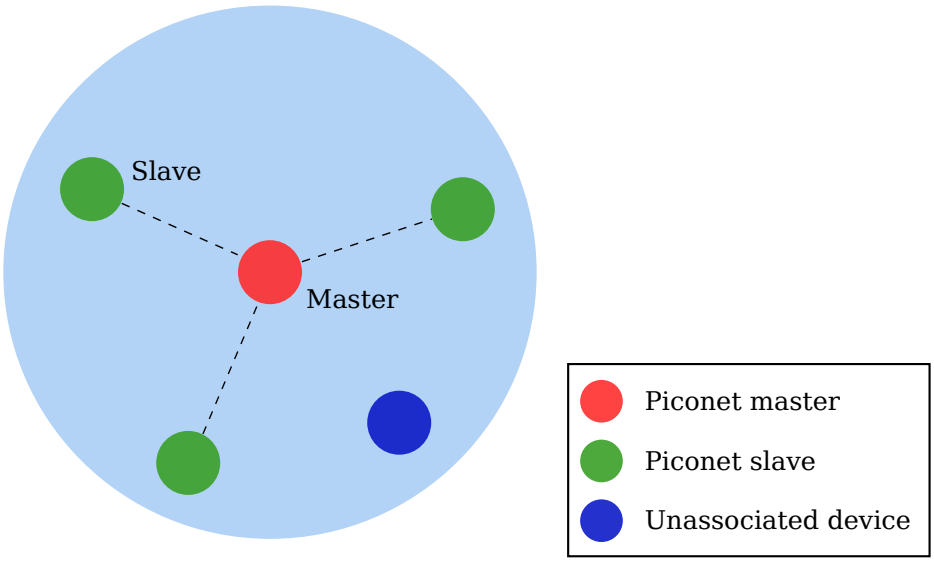

Figure 2.7: Bluetooth wireless network: piconet formation.

In order for two Bluetooth devices to start communication, the Bluetooth devices should go through the *pairing* process. In the pairing stage, Bluetooth devices exchange essential information for link establishment such as: the encryption status, encryption keys, hopping sequence synchronization and so one. To synchronize their hopping sequence, the piconet master sends a special information packets to the slave(s) that contains the master device MAC address and clock. By using these two parameters, slave(s) can hop along with the piconet master device and receive transmitted wireless packets normally.

Chapter 3

The Hop Selection Kernel: Modeling and Analysis

We have seen in the previous chapter that the hopping behaviour is totally mastered by the piconet master device. In the Bluetooth device, there is a built-in hardware called *Hop Selection Kernel*. This piece of hardware is responsible for determining the hopping sequence of the Bluetooth system. Despite the importance of the hop selection hardware, most of researchers avoid diving into the design details of such box. It is even considered as "black box", such as in [3]. Therefore, analysis the components and understanding the function of such box are the main goals of this chapter.

We start this chapter by defining the set of input and output of the hop selection kernel along with internal components of the kernel; this is in section 3.1. In section 3.2, we present the proposed model for the kernel. This modeling is useful for better understanding the functionality of the Hop Selection Kernel. In this modeling, we propose dividing the kernel into three parts: Part A, Part B, and Part C. The function of each part in investigated in sections 3.3, 3.4, and 3.5 respectively. Examining and analysing the role of each part is important to understand the vulnerable points in each part, and demonstrate security threats for the kernel.

3.1 Hop Selection Kernel: Internals

The block diagram of the Hop Selection Box and its main input parameters were presented in section 2.2. In this section, we discuss the internal components of the selection kernel, the input parameters of each component, and the RF channel table. The kernel internals block diagram is illustrated in Fig. 3.1:

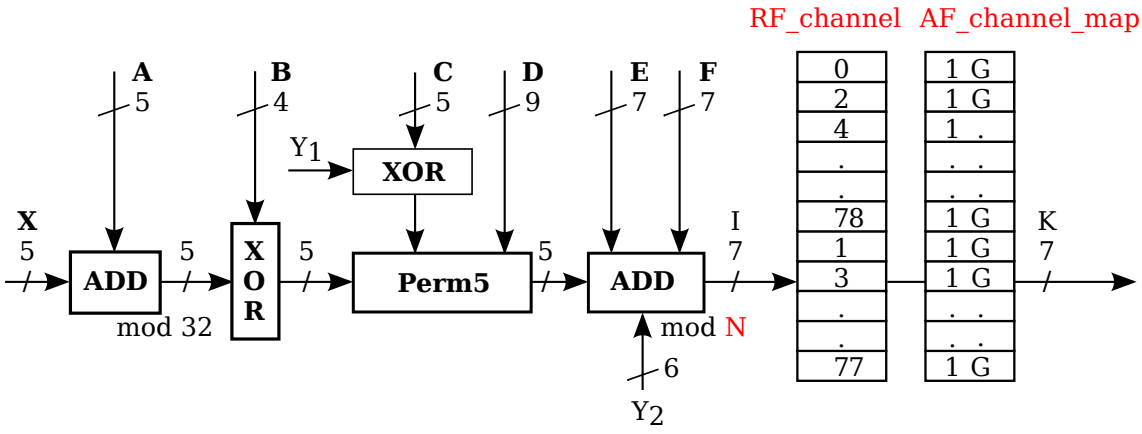

Figure 3.1: The Hop Selection kernel: internal components

3.1.1 The kernel components

The selection kernel is composed of the following components:

1. **ADD** block: this is a 5-bit adder. The inputs to this component are A and X; while the output is the parameter Z'. This block is followed by mod_{32} operation.

2. **XOR** block: there are two inputs for the XOR block: the input parameter B and the parameter Z'. This block outputs the parameter Z.

3. **Perm5** block: the input for the Perm5 block is the parameter Z; while the output is the parameter R. The function of this block is controlled by the input parameters: C, Y_1, and D. The definitions of the input parameters will include more details.

4. The second **ADD** block: the inputs to this block are E, F, Y_2, and R. The output of this component is a 7-bit parameter, which we call I (In this work, we name this segment of the selection kernel as I since it is unnamed by the IEEE Standard [1]). This block is followed by mod_N operation; where N is one of the kernel input parameters.

3.1.2 The input parameters

The selection kernel inputs are defined in section 2.2. The kernel components inputs are mainly derived from: the Bluetooth device MAC address and the device clock. The device MAC address determines the hopping sequence of the Bluetooth; while the device clock determines the phase inside the hopping sequence. The input parameters for the kernel components are defined as follows:

(1) X: a 5-bit binary number that can be directly derived from the device clock bits {6,5,4,3,2}. Formally, we can express this as follows:

$$X = CLK_{6-2} \tag{3.1}$$

(2) A: a 5-bit binary number that can be obtained by XORing (XOR is the logical bit-wise XOR operation) the MAC address bits {27,26,25,24,23} with the device clock bits {25,24,23,22,21}, and it is expressed as follows:

$$A = MAC_{27-23} \oplus CLK_{25-21} \tag{3.2}$$

(3) B: a 4-bit binary number that is directly derived from the MAC address bits {22,21,20,19}, and it is expressed as follows:

$$B = MAC_{22-19} \tag{3.3}$$

(4) C: a 5-bit binary number that can be obtained by XORing the MAC address bits

{8,6,4,2,0} with the device clock bits {20,19,18,17,16}, and it is expressed as follows:

$$C = MAC_{8,6,4,2,0} \oplus CLK_{20-16} \tag{3.4}$$

(5) D: a 9-bit binary number can be obtained by XORing the MAC address bits {18-10} with the device clock bits {15-7}, and it is expressed as follows:

$$D = MAC_{18-10} \oplus CLK_{15-7} \tag{3.5}$$

(6) E: a 7-bit binary number that is directly derived from MAC address bits {13, 11, 9, 7, 5, 3, 1}, as follows:

$$E = MAC_{13,11,9,7,5,3,1} \tag{3.6}$$

(7) F: a 7-bit binary number that can be obtained by multiplying the device clock bits {27-7} by 16 followed by mod_N operation, and it is expressed as follows:

$$F = [16 \times CLK_{27-7}] \mod_N \tag{3.7}$$

(8) Y_1: a 5-bit binary number that is derived from the second bit of the device clock, namely the CLK_1 bit. In fact, the clock bit CLK_1 is duplicated into 5 bits to be suitable for 5-bit XOR operation with the parameter C. Therefore, Y_1 can take one of the following forms:

$$Y_1 = \begin{cases} 00000 & \text{if } CLK_1 \text{ is } 0 \\ 11111 & \text{if } CLK_1 \text{ is } 1 \end{cases} \tag{3.8}$$

(9) Y_2: a 6-bit binary number that can be obtained by multiplying CLK_1 by 32. Therefore,

Y_2 can take one of the following forms:

$$Y_2 = \begin{cases} 000000 & \text{if } CLK_1 \text{ is } 0 \\ 100000 & \text{if } CLK_1 \text{ is } 1 \end{cases} \tag{3.9}$$

Note that the first bit of the device clock, namely CLK_0, is not involved in the operation of the hop selection kernel [1].

3.1.3 The RF channel table

The RF channel table and the AFH_channel_map are shown at the right side of Fig. 3.1. The RF channel table is used to determine the value of the output parameter K. This table is basically a register contains the RF channel indices. In other words, the table holds all possible values of the parameter K, which is defined in equation 2.1. The list of channel indices is ordered so that all even RF channel indices are listed first and then all odd indices are listed. This is important to have a 32-hop segment spans over 64 MHz [1].

It is important to consider the role of the input parameter AFH_channel_map in more details. This input parameter is associated with the RF channel table. It provides the kernel with useful information about the status of the Bluetooth wireless channels. By detecting the noise level, each wireless channels can be classified as:"Good", "Bad", or "Unknown". Good channels can be used by the Bluetooth; while, bad channels should be avoided. The mechanism of determining the status of a wireless channel is beyond the scope of this work. The mechanism of calculating the output parameter K can be stated as follows: the kernel utilizes the parameter I to address the RF channel table; and then the value of the output parameter K is determined based on the contents of the RF channel table. Note the two inputs parameters N and AFH_channel_map are related. That is marking any channel as "Bad" channel by the AFH_channel_map leads to decreasing the value of N by one; where N is the number of Good channels as defined in section 2.2.5. The RF channel table is a

constant one-to-one function that maps the value of the parameter I into a specific value at the kernel output parameter K. This mapping function can be expressed as follows:

$$I = \begin{cases} k/2 & \text{if } k \text{ is even} \\ \lfloor k/2 \rfloor + 39 & \text{if } k \text{ is odd} \end{cases} \tag{3.10}$$

In the following, we will consider the hop selection kernel as it contains no RF channel register. The parameter I can be considered as the output of the kernel. This is because the sequence of the parameter I can directly be obtained when the sequence of the kernel output K is obtained. The revised version of the hop selection is illustrated by the block diagram of Fig. 3.2.

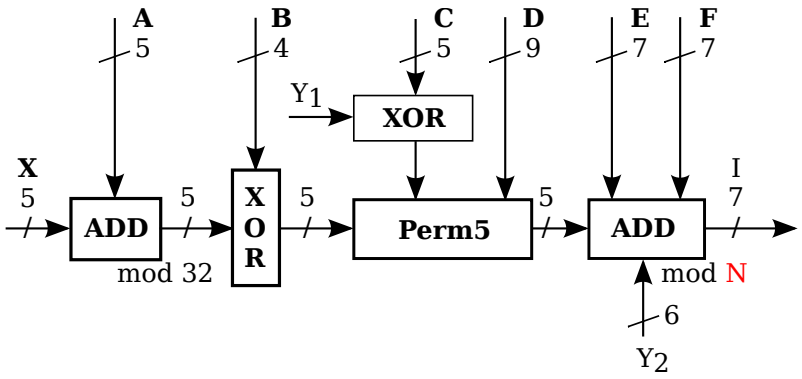

Figure 3.2: A revised version of the Hop Selection kernel block diagram.

3.2 The Hop Selection Kernel: Modeling

In the previous section, the internal details and the inputs definitions of the hop selection kernel were discussed. For better understanding of the selection kernel operation, we suggest partitioning the kernel into three parts: Part A, Part B, and Part C (The word "Part" will be associated with kernel partition's letter to distinguish them from the kernel input parameters: A, B, and C). The kernel partitioning is illustrated in Fig. 3.3:

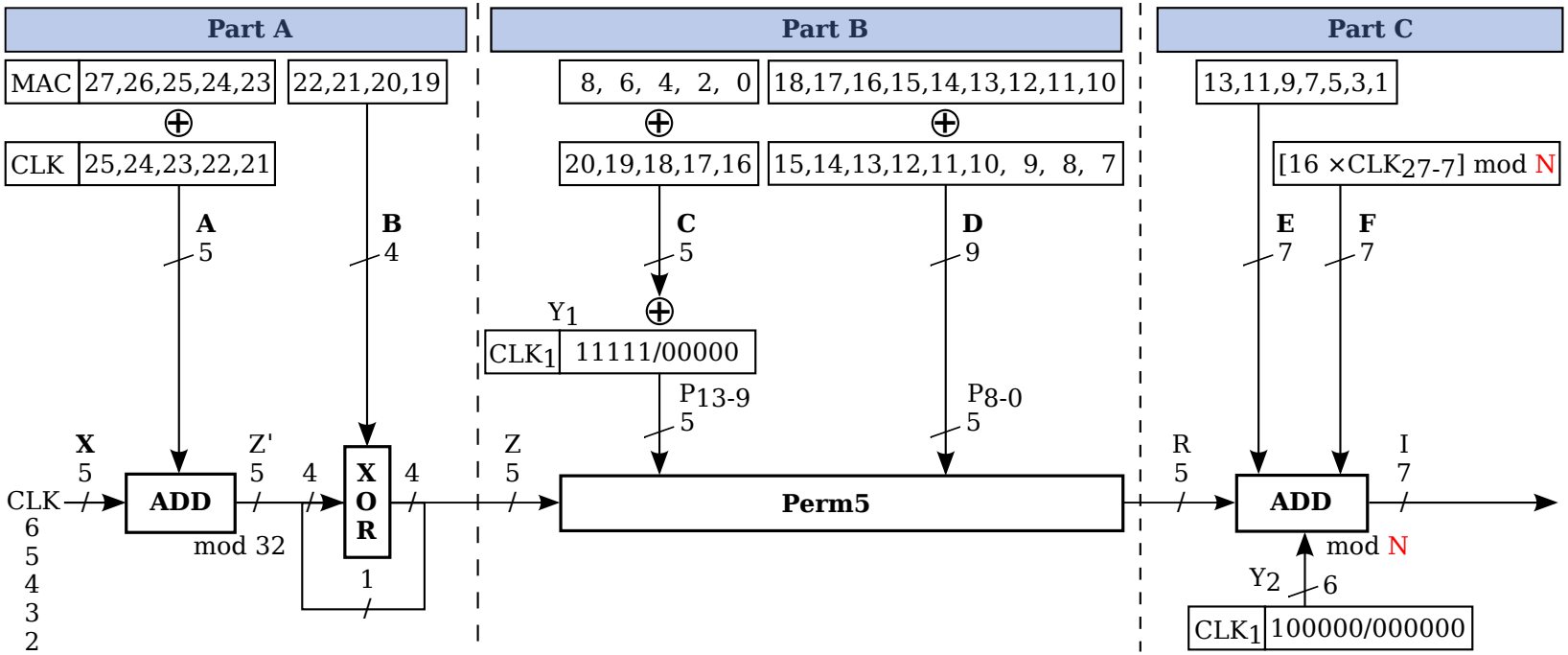

Figure 3.3: Dividing of the HSK into three parts: A, B, and C.

This kind of partitioning is useful since it breaks the problem into smaller parts that each can be conquered individually. Based on the analysis of each part, we find a security threat where as eavesdropper can predict the value of the device MAC address and the device clock seeded inside the kernel. In the following three sections, we discuss the function of each part of the hop selection kernel.

3.3 Part A of the Selection Kernel

Part A is important for the Bluetooth Hop Selection kernel as it initializes the hopping sequence generation procedure. The structure of Part A is illustrated in Fig. 3.4, where it is composed of two principle components as follows:

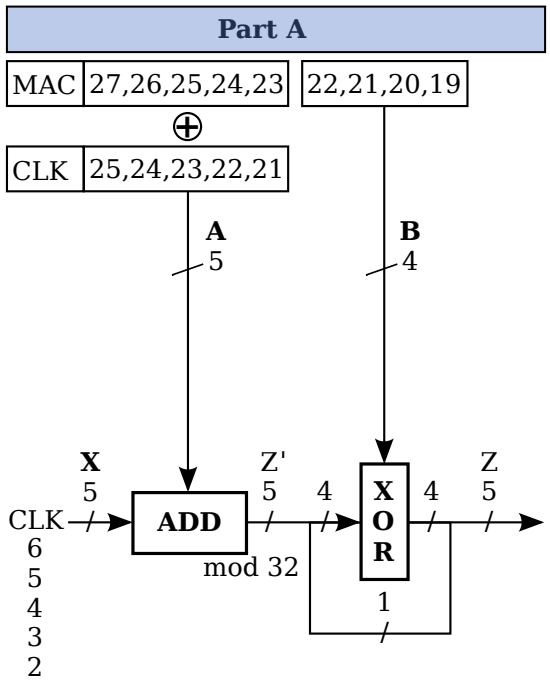

Figure 3.4: The hop selection kernel: Part A.

(1) The **ADD** block: is a 5-bit binary adder block. There are two input parameters for this block, which are: X and A and both are 5-bit binary numbers. This ADD block is followed by mod_{32} operation, and its output is the parameter Z'.

(2) The **XOR** block: is a 4-bit XOR block. There are two inputs for the XOR block: the parameter Z'_{3-0} and the input parameter B; which are both 4-bit binary numbers. The output of this XOR block is Z_{3-0}.

By reviewing the components definition of Part A, we defined the input parameters to Part A that are: X, A, and B. On the other hand, the output of Part A is a 5-bit binary number Z_{4-0}. The most significant bit of Z, namely Z_4, can be obtained by forwarding the most significant bit of Z', namely Z'_4 directly to the output Z.

3.3.1 Observations and Discussion

In this subsection, we will try to find a mathematical expression for describing each component in Part A; this would be useful for understanding and predicting their performance. First, we describe the input parameter X that initializes the hopping sequence generation procedure. The input X provides Part A, and the whole kernel, with a natural counting sequence. This is true because X is directly derived from the device clock portion CLK_{6-2}. The natural counting sequence goes from 00000 ($\equiv 0_{10}$) up to 11111 ($\equiv 31_{10}$) then resets again to 00000 ($\equiv 0_{10}$). Recall that X is a 5-bit binary parameter.

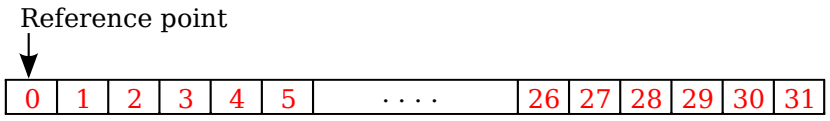

Figure 3.5: The natural counting sequence provided by the input A.

Figure 3.5 illustrates the natural counting sequence and its reference point where it starts counting. Next we examine the **ADD** and the **XOR** components of Part A.

3.3.1.1 The adder block

The operation of this block is ruled by the following equation:

$$Z' = (X + A) \mod 32 \tag{3.11}$$

The **ADD** block in Part A shifts the natural counting sequence by the value of A parameter. As an example, assuming the value of A is 5 ($\equiv 00101_2$) at the moment, then the counting sequence of the parameter Z' would look like: {5,6,...,31,0,1,2,3,4} and it illustrated in Fig. 3.6.

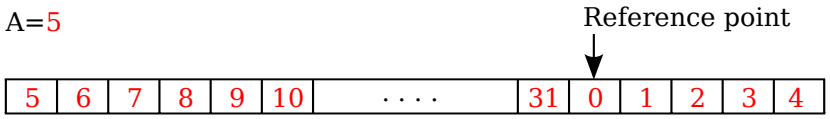

Figure 3.6: The shifted natural counting sequence of Z'.

The following observations are important about the adder block:

(1) The input A can be considered as a constant value for a while. The parameter A is derived from: (a) the MAC address portion MAC_{27-23} which is constant; and (b) the device clock portion CLK_{25-21}, which also can be considered as a constant for a while. The clock portion CLK_{25-21} requires about 655 seconds to alter. This is a long period of time when it is compared with the hops generation rate of the Bluetooth. The Bluetooth generates 1600 hops per second.

(2) The $\mod_{32}$ operation, which follows the **ADD**, acts as clipper that only keeps the 5 least significant bits of the adder's result. Any extra bits of the addition operation will be neglected. This can be justified by the Fact 3.1.

FACT 3.1: A special case of $\mod_N$ operation, if the divider integer N can be expressed as

2^m, then the mod_N will be restricted to a logical AND operation with $(2^m - 1)$. This can be expressed formally as follows:

$$x \mod 2^n \equiv x \wedge (2^n - 1) \tag{3.12}$$

3.3.1.2 The XOR block

The operation of the XOR block can be defined as a group of functions G as follows:

$$G = \{g_i \ , \ 0 \leq i \leq 15\} \tag{3.13}$$

Group of functions means that for every value of i, g_i can be defined as a one-to-one mapping function from the input space to the output space. This can be formally expressed as:

$$g_i : \ Z' \rightarrow Z, \ 0 \leq i \leq 15, \tag{3.14}$$

The input space for the XOR block is the set of all possible values of Z'_{3-0}, which are contained in {0, 1, ..., 15}. On the other hand, the output space is the set of all possible values of Z_{3-0}, that are also contained in {0, 1, ..., 15}. Note that the value of i is the value of the constant input parameter B. Therefore, the version of the g_i function and its effect can be determined in advanced in case of knowing the MAC address. The **XOR** basically conducts permutation on the shifted input sequence provided by Z'. Another important fact is, for any XOR operation, the input set, which is the domain, is the same as the output set, or the codomain. This means the XOR operation does not produce a new set of elements rather than those defined in the input set. Figure 3.7 illustrates the shifted counting sequence, shown in Fig. 3.6, under the effect of the **XOR** when $B = 5$:

5	4	7	6	3	2	....	30	25	28	24	27	26

Figure 3.7: The shifted counting sequence under the effect of the XOR block.

As a result, Part A acts as one-to-one mapping function from the input space X to the output space Z. This can formally be expressed as follows:

$$Part_A : \ X \rightarrow Z \tag{3.15}$$

The input space for Part A is the set of all possible values for the parameter X, namely the 32 elements contained in the set {0, 1, ..., 31}; and the output space is all possible values for the parameter Z, that are also contained in the set {0, 1, ..., 31}. This can be expressed in other words: Part A provides the selection kernel with an input sequence that is a shifted-permuted version of the natural counting sequence {0, 1, ..., 31} initialized by the parameter X. The shifting amount and permutation scheme are determined based on the values of A and B.

3.4 Part B of the Kernel: "Perm5" Stage

Part B of the Hop Selection Kernel is also known as "Perm5 stage" (We will use "Part B" or "Perm5 stage" alternatively to refer to the same part of the kernel.) [1].

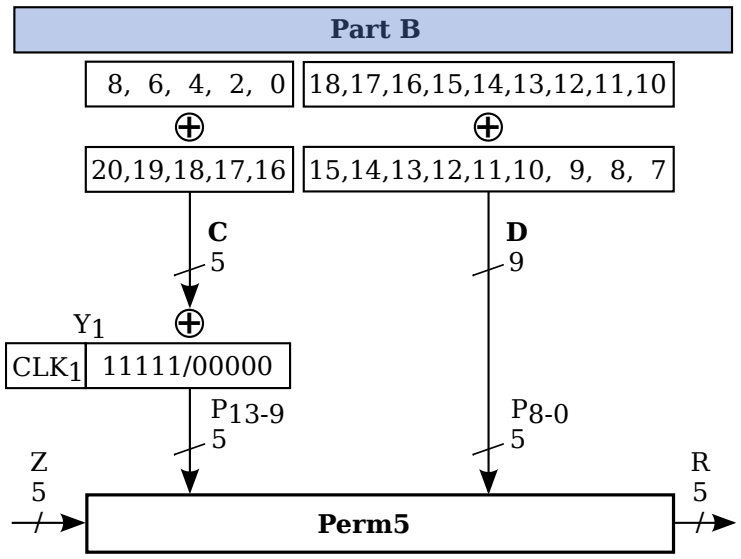

Figure 3.8: The hop selection kernel: Part B.

Part B is illustrated in Fig. 3.8. There is only one block in Part B which is the "Perm5" block. The input to this block is the parameter Z. And it is controlled by a 14-bit control word called the control word P. On the other hand, the output of Part B_{12} is a 5-bit binary number called R parameter, as shown in Fig. 3.8.

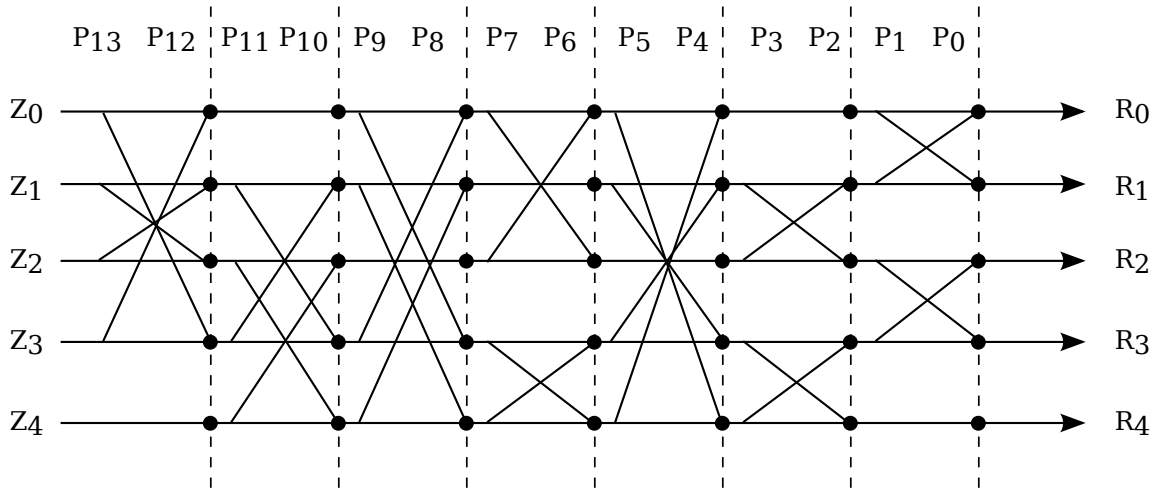

Figure 3.9: Permutation operation of the Perm5 Block.

For understanding the role of the Perm5 stage, let us examine the internal design of this block. This block consists of 14 cascaded cells; each cell is responsible for swapping two bits of the input parameter Z. The internal structure of Perm5 block is illustrated in Fig. 3.9. The cell can be constructed by using 2 multiplexers devices, each one is a 2×1 multiplexer. The cell body is illustrated in Fig. 3.10. Note that each cell is also provided by a control line "Control". The status of the control line helps the cell to make a decision about swapping the incoming bits or not. If the control line is set to "1" then the cell would swap the two input bits. Otherwise, the cell would forward the two input bits directly to the output regardless of their contents. By cascading 14 cells, there would be 14 control lines which compose the control word P of the Perm5 stage. This control word is denoted as P_{13-0} and it is composed of two parts:

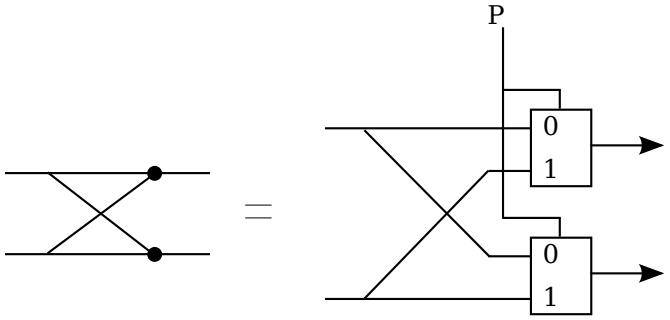

Figure 3.10: Diagram of cell body that is constructed by two MUX (2 × 1), [1].

(1) The first part P_{13-9}: a 5-bit word can be obtained by XORing two input parameters: C and Y_1 as in the following equation:

$$P_{13-9} = C \oplus Y_1 \tag{3.16}$$

(2) The second part P_{8-0}: a 9-bit word is directly derived from the input parameter D, as follows:

$$P_{8-0} = D \tag{3.17}$$

The role of Part B is reordering the 5 bits of the input Z. In other words, it shuffles the five bits of the input Z, and produces permutation version of Z's bits. As an example of Part B operation, when Z=8 ($\equiv 01000_2$), Part B presents one of the following values:

$$\{(00001_2), (00010_2), (00100_2), (01000_2), (10000_2)\} \tag{3.18}$$

By reviewing the architecture of Part B or the Perm5 stage, we defined the input parameters to Part B, which are Z and P, and defined the output parameter R.

3.4.1 Observations and Discussion

Just like Part A, we start the analysis by finding a mathematical expression for describing the behaviour of this Part. This kind of description is useful for better understanding the effects of Part B.

3.4.1.1 Mathematical description

As it is mentioned previously, Part B produces a permutation version of the input bits. This means the codomain set of elements is the same as the domain set of elements; there is no new elements are added by Part B. Based on the experimental results, we found that the Perm5 stage is a one-to-one mapping function from the input space Z to the output space R. The version of the mapping function is determined by the value of the control word P. Therefore, Part B represents a group of functions P_f as follows:

$$P_f = \{P_{fj} \ , \ 0 \leq j \leq 16383\} \tag{3.19}$$

The term "Group of Functions" means that for each value of j, there is a P_{fj} that can be defined as a one-to-one mapping function from the input space to the output space. This can be formally expressed as follows:

$$P_{fj} : \ Z \rightarrow R, \ 0 \leq j \leq 16383 \tag{3.20}$$

The input space for Par B is the set of all possible values of the parameter Z, namely {0, 1, ..., 31} and the output space is the set of all possible values of R which are contained in the same set {0, 1, ..., 31}. Also it is important to note that the value of j is the instantaneous value of the control word P. Note there are 16384 possible values of the control word P since $2^{14} \equiv 16384$.

3.4.1.2 The effect of the device clock on Part B

In order to study the performance of Perm5 stage, we have to understand the flipping rates of some bits of the device clock. The "flipping rate" refers to the required period of time for a bit, or a sequence of bits, to alter its value. With respect to Part A, The behaviour of Part A is directly affected by the device clock CLK_{6-2} and Part A presents a different value on its output if the input value of the clock portion CLK_{6-2} is changed. Part B is affected by the clock portion: CLK_{20-7} and CLK_1.

As we mentioned previously in Chapter 2, there is a specific frequency for each bit of the device clock. In other words, the flipping rate is different for each bit, or part, of the device clock. We can list the the required time for one flip for some important parts of the device clock as follows:

(a) One flip of $CLK_1 = 625$ μ second.

(b) One flip of $CLK_{6-2} = 2 \times$ period of $CLK_1 = 1250$ μ second.

(c) One flip of $CLK_{20-7} = 2^6 \times$ period of $CLK_1 = 40000$ μ second.

(d) One flip of $CLK_{25-21} = 2^{20} \times$ period of $CLK_1 = 655360000$ μ second.

Because CLK_1 flips twice faster than the clock portion CLK_{6-2}, we can conclude that Part B applies two different control words for each element presented by the input Z. The first control word, which we denote as P, is produced when the value of Y_1 is 00000. On the other hand, the second control word, denoted as P', is produced when the value of Y_1 is 11111. These two values of Y_1 affect the first part of the control word P. This is expressed as follows:

$$P = (P_{13-9} \oplus 00000)|P_{8-0}, \tag{3.21a}$$

$$P' = (P_{13-9} \oplus 11111)|P_{8-0}, \tag{3.21b}$$

This important case is illustrated in Fig. 3.11.

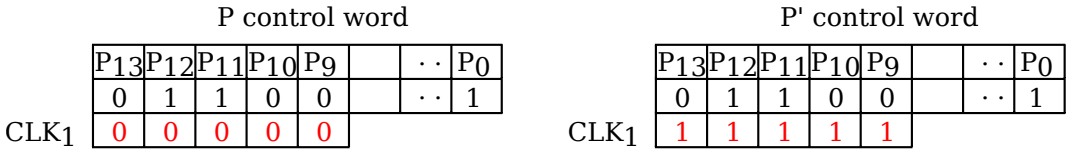

Figure 3.11: Z's elements are manipulated by two different control words.

For this reason, the input sequence supplied by Z contains 32 elements while the sequence at the output R contains 64 elements. This is true because each element in the input sequence Z is mapped into two images at the output sequence R. This is illustrated in Fig. 3.12.

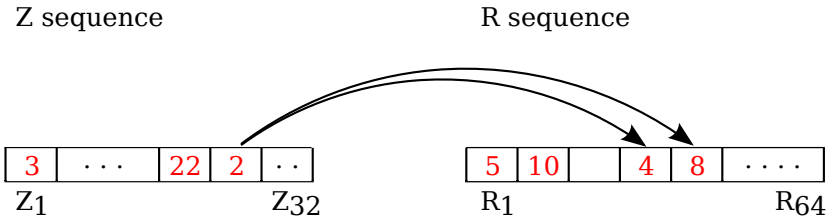

Figure 3.12: Z's elements are mapped into two images in the sequence of R.

It is important to notice that we have two elements are excepted from the effect of Part B. These two elements are: 0 and 31. The output sequence of Part B contains the following pairs of elements: {0,0} and {31,31} according to the occurrence of 0 and 31 respectively. This can be justified as follows: the first element 0 is not affected by the operation of Part B because the binary representation of the input element 0_{10} is 00000_2; this means permuting these five bits does not produce any new element, i.e. the output will stay 0 whatever the value of the control word P. The same argument is true with respect to the input element $31_{10} \equiv 11111_2$. This is illustrated in Fig. 3.13.

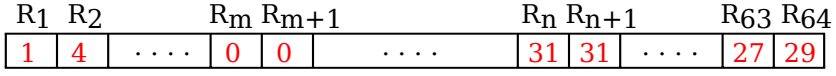

Figure 3.13: The unaffected elements 0 and 31 within the sequence R.

In general, we can state that Part B is a one-to-one mapping function from the input space Z to the output space R. This is expressed as follows:

$$Part_B : \ Z \rightarrow R \tag{3.22}$$

The input space, or the domain, is the set of all possible values of the parameter Z that are the 32 elements contained in the set S_{32} = {0,1,2,....,31}. On the other hand, the output space R has 64 elements that are all derived from the set S_{32}. The elements of the set S_{32} are duplicated in the output space.

3.5 Part C of the Kernel

Part C is the third part of the Hop Selection Kernel. This part is illustrated in Fig. 3.14.

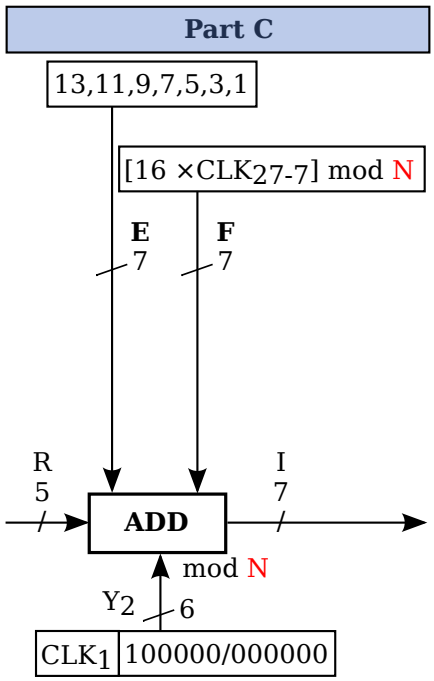

Figure 3.14: The hop selection kernel: Part C.

Part C consists of one adder block **ADD**, which is followed by mod_N operation, where N

is the number of "Good" channels that is determined by the Adaptive Frequency Hopping (AFH) facility. The inputs to the ADD block of Part C are:

(1) E: a 7-bit binary number that is a constant since it is derived from the MAC address.

(2) F: a 7-bit binary number that is ruled by equation 3.7.

(3) Y_2: a 6-bit binary number that is rules by equation 3.9

(4) R: a 5-bit binary number supplied by Part B of the selection kernel.

By defining the input parameters to this block, we actually defined the inputs for Part C of the kernel. On the other hand, the output of the this adder block is K; this is, which is also the output of the kernel. The parameter I is a 7-bit binary number.

3.5.1 Observations and Discussion

Since Part C has one adder block **ADD** followed by mod_N operation, then it is formally expressed as follows:

$$I = [E + Y_2 + F + R] mod_N \tag{3.23}$$

The effects of Pact C can be explained by the following points:

1. The input parameter R provides a sequence of 64 elements. Each element can range from 0 ($\equiv 00000_2$) to 31 ($\equiv 11111_2$).

2. The role of parameter Y_2 is to span these incoming elements over 64 MHz. It simply adds 0 to odd elements of R and adds 64 to even elements.

3. The parameters E and F try to redistribute the sequence again over 79 channels. Note the value of the parameter F is changed when the input X is rested to 0. In other words, for a complete 64 elements presented by the R parameter, the value of F does not change.

4. The mod_N follows the adder ADD block basically acts as clipper that keeps the addition result within the range of from 0 to 78.

Part C is the interface of the selection kernel and outputs the final hopping sequence. So monitoring and analysing the outgoing sequence of this part is the only way to predict the future behaviour of the selection box. Part C can also be seen as the guard of the kernel because breaking the mod_N operation seeded in this part is not trivial. Therefore, the first step to crack the Hop Selection kernel is to attack Part C. At the first glance, this mission seems to be hard but we will see in the next chapter how to deal with it.

In chapter 3, we discussed the analysing and modeling of the selection kernel. We started by defining the internal components of the kernel. Then we presented our proposed method for dividing the kernel; that is partitioning the kernel into three parts. This analysis is important for better understanding the kernel function. Afterward, we discussed the function of each part individually. This included finding a mathematical expression that rules the kernel parts. This analysis steps are considered as the basis for the next chapter where we discuss the kernel attacking method.

Chapter 4

The Hop Selection Kernel: Possible Attacking Method and Simulation

In chapter 3, the hop selection kernel was modeled and analysed. The modeling strategy suggests dividing the kernel into three partitions: A, B, and C. The suggested model is a kind of breaking the original problem into three smaller problems that each can be treated separately. In this chapter, we relies on the analysis and the kernel model proposed in the previous chapter to develop a suitable method for attacking individual parts. This leads to a systematic method for cracking the whole kernel operation. Cracking the kernel basically means revealing the numerical value of the device clock, hence, predicting the future hopping sequence is possible.

The suggested methods for cracking the kernel parts are based on some useful results obtained by Spill [6] who showed how the device MAC address and the first six bits of the clock, namely the clock portion CLK_{6-1}, can be obtained.

In sections 4.1, the proposed method for cracking Part A is presented. Removing the effects of Part A is discussed in this section. Cracking Part B of the kernel is discussed in section 4.2. The P control word is targeted. The cracking method proposes constructing a table all possible outcomes of Part B. This table is constructed by considering all possible

incomes to Part B. Then monitored patters can be compared against the table to extract the P control word. In section 4.3, a suitable method for cracking Part C is presented. The proposed strategy invest wireless jamming to reduce the value of the input parameter N into 32. by this way, the mod_N operation seeded in Part C follows Fact 3.1 and can be cracked. In section 4.4, we review a simple scenario that shows a general idea of cracking the kernel in total. In section 4.5, the simulation software and the implementation challenges are discussed. We also present useful solutions that can be adopted to overcome the stated implementation difficulties.

4.1 Possible Method for Attacking Part A

In chapter 3, we discussed the operation of Part A. This part shifts and permutes the natural counting sequence generated by the input parameter X. The shifting and permutation processes are conducted via the ADD block and the XOR block respectively. The amount of shifting is determined based on the value of input parameter A to the ADD block. On the other hand, the permutation function is determined based on the value of the input B of the XOR block.

The main goal of attacking Part A is to expose the values of the input parameters: A and B. This would provide more information about the input sequence supplied by Part A to other parts of the Kernel; in other words, we can predict this kind of sequence. To be more specific, we should be able to answer the following questions in terms of breaking Part A:

(a) How the natural counting sequence of the input X is shifted by the ADD block? What is the value of input parameter A?

(b) How the shifted sequence has been permuted by the XOR block? To answer this question, the value of the parameter B should be known.

4.1.1 Discussion

In chapter 2, we discussed the the results of Spill [6]. The paper proposed a method to extract the device MAC address and the first six bits of the clock. Hence, these two parameters are known by the literature [6]. This leads to the following facts:

(i) The value of the input B is known since it is part of the MAC address.

(ii) The value of the input parameter X associated with each hop of the sequence is known as well.

We continue to calculate the value of the input parameter A. To keep things simple, consider the Hop Selection kernel consists of Part A only. The next step is to monitor the fluctuation of the output parameter Z, for a specific period of time and recording the outgoing sequence of hops. Then, the value of the input parameter A is calculated by applying the following equations:

$$Z'_{3-0} = Z_{3-0} \oplus B, \text{ and } Z'_4 = Z_4 \tag{4.1}$$

And then,

$$A = Z' - X_i, \ 0 \leq X_i \leq 31 \tag{4.2}$$

Note that by revealing the value of the input A, the value of the clock portion CLK_{25-21} can be calculated by reversing equation 3.2 as follows:

$$CLK_{25-21} = A \oplus MAC_{27-23} \tag{4.3}$$

The value of this portion of the clock is important in order to follow the hopping sequence for a long time period.

Because the Hop Selection kernel has two more parts, namely Part B and C, the previous assumption of considering Part A only is not the practical one. We gradually progress by adding one more part, i.e. Part B, and see what we can do.

4.2 Possible Method for Attacking Part B

In chapter 3, we discussed the function of Part B. The control word P of this part is derived by XORing the clock portion CLK_{20-7} with some portions of the device MAC address. Since the clock portion CLK_{20-7} is naturally counting from 0 up to 16383, then the possible values of the control word P lies in the range [0,16383]. The sequence supplied by Part A, was also discussed in chapter 3. This sequence is a shifted permuted version of the natural counting supplied by the input parameter X. Moreover, the supplied sequence by Part A does not change unless the clock portion CLK_{20-7} counts from 0 up to 16383 and reset again to 0. One important fact about Part B is that the sequence of 32 input elements is considered as a sequence of 32 pairs; where the elements of each pair are identical. The role of Part B is to break the bounded pairs and redistribute them over the set of 64 elements at the output R. The elements of each pair would be scattered in a pseudo random fashion depending on the value of the control word P.

By attacking Part B of the Hop Selection kernel, we mainly aim to reveal the value of the control word P and then extract the clock portion CLK_{20-7} of the targeted device. The proposed method for attacking Part B can be described as a pattern recognition procedure. We first consider all possible sequences that can be presented by the output parameter R. This can be conducted by considering all possible input values of Part B, the parameter Z and the control word P, and construct a table of all possible hopping sequences at the output R. We denote the constructed table as T. This scheme is similar to the idea of building a truth table that holds all possible outcomes against all possible incomes of a specific logical expression. Second we collect a sequence of hops which is considered as a "pattern". The word "pattern" refers to a hopping sequence of 64 elements presented by the output R when Part B is fed by 32 elements from Part A. Third we look up the table T and find the matching pattern(s). At this point, the features of the table T are clear: it has 16384 rows and each row holds a pattern of 64 elements. Each row reflects a specific value of the clock portion

CLK_{20-7}. The table is illustrated in Fig. 4.1:

P\X	0	0	1	1	2	2	3	....	29	29	30	30	31	31
0	.	.	.	.	.	.	.		.	.	.	.	.	.
1	.	.	.	.	.	.	.		.	.	.	.	.	.
2	0	0	1	4	2	16	20	....	29	15	30	27	31	31
3	0	0	1	2	4	16	18	....	27	15	30	29	31	31
4	0	0	2	16	8	10	17	....	23	30	29	15	31	31
5	0	0	1	4	2	16	20	....	29	15	30	27	31	31
.	.	.	.	.	.	.	.		.	.	.	.	.	.
.	.	.	.	.	.	.	.		.	.	.	.	.	.
.	.	.	.	.	.	.	.		.	.	.	.	.	.
16381	0	0	1	2	4	16	18	....	27	15	30	29	31	31
16382	0	0	2	16	8	10	17	....	23	30	29	15	31	31
16383	0	0	1	4	2	16	20	....	29	15	30	27	31	31

Figure 4.1: All Possible Outcomes of Part B: the table T.

For attacking Part B, we reassemble Part A and B together and follow the steps of:

(1) Building the table T, as shown in Fig. 4.1.

(2) Monitoring the output parameter R and collecting some hopping patterns. This is illustrated in Fig. 4.2.

(3) Comparing each collected pattern against the table T and reporting a list of candidate values for the control word P. This is illustrated in Fig. 4.3.

(4) Filtering the reported lists by dropping false values. This is illustrated in Fig. 4.4.

	Collected patterns													
Pattern1	0	0	1	2	4	16	18	....	27	15	30	29	31	31
Pattern2	0	0	2	16	8	10	17	....	23	30	29	15	31	31

Figure 4.2: Two hopping patterns are collected online.

hit P=	Comparing pattern1 against the table													
4	0	0	2	16	8	10	17	. . . .	23	30	29	15	31	31
.	.	.	.	.	.	.	.	. . . .	.	.	.	.	.	.
16382	0	0	2	16	8	10	17	. . . .	23	30	29	15	31	31

Lists of candidate values for P control word

L1	4	.	. . . .	16382	L1 is collected by comparing pattern1
L2	3	.	. . . .	16381	L2 is collected by comparing pattern2

Figure 4.3: Comparing each pattern with the table T and listing candidates.

	C1	C2	C3	C4	C5	C6	C7	. . . .	Cn
L1	4	21	49	73	87	92	98	. . . .	16382
L2	~~3~~	~~20~~	50	74	88	93	99	. . . .	16~~38~~1
Filtered list	50	74	88	93	99				

Figure 4.4: Filtering candidate lists by dropping false values.

By repeating the steps 2-4, the number of candidates shrinks into one element that is the real value of CLK_{20-7}. The list of candidates is reduced by dropping false values from the candidates list. The criteria that we adopt for dropping false values bases on the following fact: consecutive hopping patterns of the Bluetooth device are produced by consecutive values of the clock portion CLK_{20-7}. This is why the values marked with red cross are dropped, as shown in Fig. 4.4.

4.2.1 Discussion

Attacking Part B is not as simple as it looks. However, we have to consider the following issues:

- What is the suitable sequence for the parameter Z? In order to construct the table T, a specific sequence should be considered for the parameter Z.
- How can we recognize a complete pattern of 64 elements? Assuming a sequence of two or three hundreds of hops is just collected, this does not ensure knowing the starting

point of each hopping patterns.

In the proposed setup of the selection kernel, Part A and Part B are reassembled together. This means that only the parameter R can be monitored. Note hopping pattern elements presented by the parameter R are all affected by the mapping function of Part B except two elements: 0 and 31. Hence, the value of the input A can be extracted because the value of the parameter X, that is associated with each hop, is given. Moreover, the device MAC address is also given. Up to this point, we can proceed with one of the following two options:
(1) Construct the table T by calculating the parameter Z. This can be conducted by emulating the operation of Part A. This option requires the table T to be constructed online before comparing any collected patterns along the table. Online construction of the table T is a time consuming job.
(2) The other option is to neglect the effects of Part A and consider a natural counting sequence instead of the parameter Z. In this case, the table T can be constructed offline. However, the collected patterns should to revised by removing the effects of Part A before they can be compared with the table. Removing the effects of the ADD and XOR blocks of Part A in the execution time is much easier than constructing the table T.

It is obvious that (2) is the option when the table T is constructed. By considering option (2), Part A would implicitly be cracked first. The process of revising collected patterns is illustrated in Fig. 4.5.

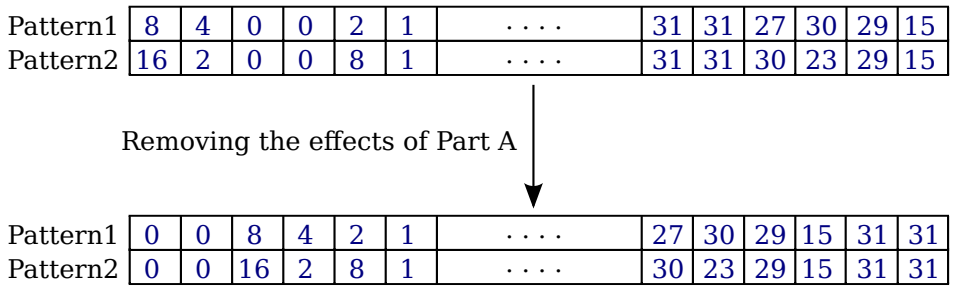

Figure 4.5: Revising two collected patterns by removing the effects of Part A.

With respect to the issue of recognizing 64-element patterns when several hundreds of hops are collected, the given values of the parameter X to recognize 64-element patterns. When the value of the input X equals 0, this means the starting point of the pattern; while the value 31 refers to the end point of the pattern.

The proposed steps to crack Part B are modified to the steps of:

(1) Building the table T using natural counting sequence instead of the parameter Z.

(2) Monitoring the output parameter R and collecting some hopping patterns.

(3) Revising the collected patterns by removing the effects of Part A.

(4) Comparing each of the revised patterns against the table T and reporting a list of candidate values for the control word P.

(5) Filtering the reported lists by dropping false values.

4.3 Possible Method for Attacking Part C

In chapter 3, we discussed the function of Part C of the selection kernel. This part contains single adder block ADD. The inputs to the ADD block are: E, F, Y_2, and R; while the output paramter is denoted as I. The role of Part C is to span the 64 elements of each hopping pattern over 64 MHz. Moreover, Part C redistribute the incoming elements over the defined space of 79 channels.

The main goal of attacking Part C is to expose the hopping patterns supplied by the input parameter R. The parameter R is basically the output of Part B. By extracting the pseudo random numbers supplied by the parameter R, we can apply the proposed methods for attacking Part A and Part B of the selection kernel. Cracking Part C is complicated because of the mod_N operation seeded in Part C of the kernel. The parameter N represents the number of "Good" channels that can be used by the Bluetooth. There are two mod_N

operations in Part C: the first one follows the ADD block; while the second one is involved in the calculation of the parameter F.

4.3.1 Discussion

The proposed method to to crack the mod_N operation seeded in Part C is to tune the parameter N value to be in the form of 2^m. In this case, the mod_N follows the principle of Fact 3.1. In the range of the parameter N, there two possible values operation that we can consider: 32 and 64. In this work, we adopt the value of 32. Wireless jamming is a useful tool that can be invested to tune the value of N down to 32. In the ideal case, jamming 47 channels of the 79 channels defined for the Bluetooth, the value of the parameter N is reduced from 79 down to 32. The wireless jamming is illustrated in Fig. 4.6.

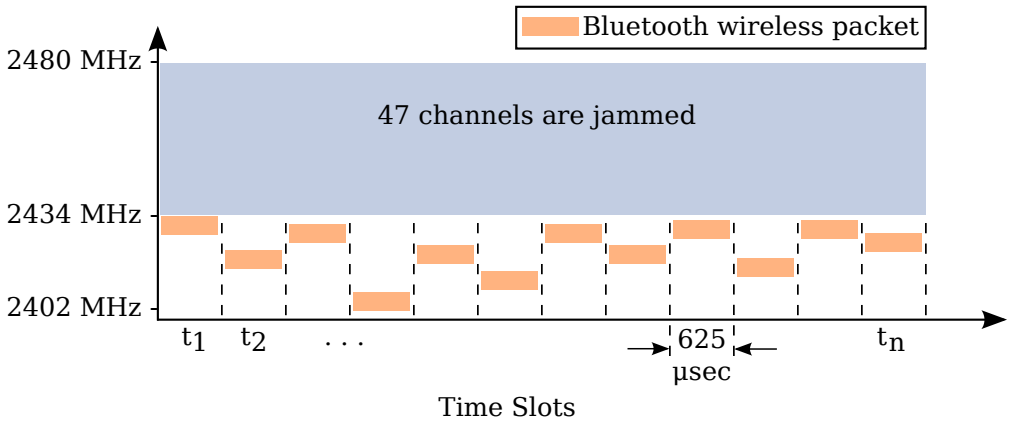

Figure 4.6: Frequency Hopping behaviour Bluetooth under jamming effect.

By tuning the value of the parameter N into 32, the inputs (E,F,Y_2) and the output I of Part C will be affected in the following ways:

(1) The input parameter E in Part C will not be affected by the value of N because it is a constant derived from the device MAC address.

(2) The least significant bits F_{4-0} of the parameter F will be considered. Recall that the mod_N operation involved in calculating the value of F follows the Fact 3.1 too. Moreover,

the parameter F will be restricted to two values: 0 and 16. This because the fifth bit F_4 of the parameter F is affected by the clock bit CLK_7, see equation 3.7. These two values of the parameter F are added to consecutive patterns alternatively.

(3) The parameter Y_2 is neglected because this parameter affects the sixth bit of the output parameter I.

(4) The output of the ADD block will be shrunk to 5 bits only. In other words, the least significant bits I_{4-0} of the parameter I will be considered.

Based on this argument, we can conclude that only two values are added to consecutive 64-element patterns of the parameter R. These two values are: $E+F(0)$ and $E+F(16)$. Despite of this addition, the unaffected values of Part B {0,31} can still be detected. Consecutive collected patterns will be in one of the following forms: A or B. Both forms are illustrated in Fig. 4.7.

Pattern1	..	..	E+0	E+0	....	31+0	31+0	..
Pattern2	..	..	E+16	E+16	....	31+E+16	31+E+16	..

(A)

Pattern1	..	..	E+16	E+16	....	31+E+16	31+E+16	..
Pattern2	..	..	E+0	E+0	....	31+0	31+0	..

(B)

Figure 4.7: Two possible form of the collected patterns from the output I.

By removing the effects of these two values, we get pure hopping patterns as they are presented by the output parameter R. This is illustrated in Fig. 4.8.

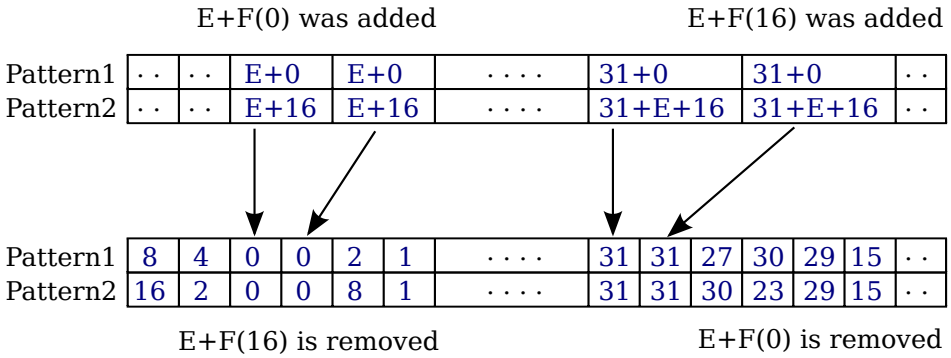

Figure 4.8: Removing the effects of Part C.

Up to this point, Part C can be cracked by following the steps of:

(1) Jamming 47 channels of the Bluetooth space; and this implies reducing the parameter N value down to 32.

(2) Monitoring the output parameter of the kernel, which is I, and reporting some hops of the Bluetooth system.

(3) Removing the effects of the parameters E and F from the collected patterns.

4.4 Simple Scenario for Attacking the Kernel

Assuming an ideal situation where two Bluetooth devices are paired and some files are being exchanged between them at the moment, as illustrated in Fig. 4.9.
The proposed attacking method follows the steps of:

(1) Constructing the table T as shown in Fig. 4.1. This is constructed offline.

(2) Starting the attack by jamming 47 channels and in order to leave 32 channels for the Bluetooth to hop over. This is shown in Fig. 4.6.

(3) Collecting some hopping patterns and removing the effects of Part C as shown in Fig. 4.7.

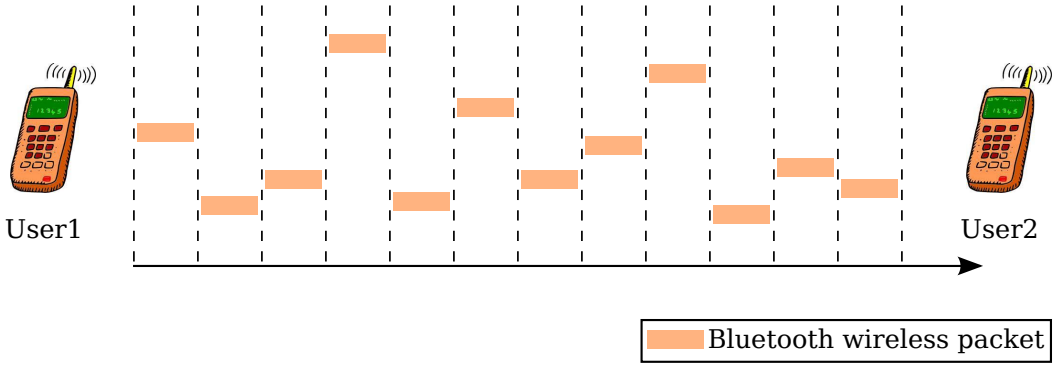

Figure 4.9: Two Bluetooth devices are exchanging wireless data.

(4) Calculating the values of the input parameters: A and B; and revising the patterns by removing the effects of Part A as shown in Fig. 4.5.

(5) Comparing each revised pattern with the table T and reporting lists of candidate values of the control word P as shown in Fig. 4.2 and Fig. 4.3 respectively.

(6) Dropping false values of the candidates as shown in Fig. 4.4.

(7) Repeating steps 5 and 6 until a unique value of the control word P is found.

(8) Removing the effect of jamming, stated in step 2.

(9) Emulating the Hop Selection kernel operation by using the revealed values of the MAC and the clock of master device; and this means hopping along with targeted Bluetooth and capturing its wireless packets is possible.

4.5 Simulation and Implementation Challenges

In this section, we discuss the simulation environment and the implementation challenges of the proposed method for cracking the selection kernel. The simulation program is important because it helps to:

(1) Have better understanding for the Hop Selection kernel operation.

(2) Validate the proposed method for cracking the selection kernel.

(3) Evaluate the performance of the proposed algorithm.

(4) Specify the implementation challenges of such algorithm.

The simulation software is constructed in Matlab R2010a environment. The program mainly consists of three components:

(a) The Hop Selection kernel: this software package imitates the operation of the selection kernel in terms of generating pseudo random hops. The inputs to this component are: the device MAC address as an integer, the device clock as an integer, the parameter N as an integer value, and finally an integer represents the number of hops to be generated. The output of this software component is a list of integers representing the hopping pattern of the kernel.

(b) The table T builder: this software component of the program is responsible for constructing the T table. It has no inputs; it just returns the table T.

(c) The Cracking Program: this software component represents the implementation code of the proposed attacking method as described in section 4.4.

4.5.1 Implementation Challenges

The discussed attacking method relies on some ideal considerations, such as: the availability of full CPU speed and large amount of cache memory for the proposed attacking method. Because this is not the practical situation, we mainly aim to reduce the required time to crack the hopping sequence of a targeted Bluetooth system. We also concern about the computational resources required for this kind of attack. This includes: the CPU speed and the required amount of memory. In general, the implementation challenges

(A) Constructing the table T is a time consuming issue. It is possible to constructed the table T offline?

(B) The size of the table T requires more than million of bytes; is it always possible to host such table on commercial hardware devices with limited amount of memory space?

(C) The proposed algorithm for searching the table is inefficient one; can we make it faster?

(D) In worst cases, how many patterns should be collected to extract a unique value for the control word P? Collecting large number of hops may not be always possible.

Regarding **(A)**, which is about the construction of the table T. This issue is discussed in section 4.2. We showed that the table T can be constructed offline by:

- Considering the natural counting sequence, such as the supplied sequence by the input parameter X, instead of the parameter Z.

- Revising collected patterns by removing the effects of Part A before they can be compared against the table T.

With respect to **(B)**, the size of the table T is $16384 \times 64 = 1048576$ bytes (1 byte per element). Because of memory limitations in most commercial hardware, the size of such table is a problem. We propose a solution that relies on some statistical analysing about the resident data in the table T. The statistics show that the table T contains many redundant patterns, or redundant rows. Hence, we compress the table T into two smaller tables that are: T_c and T_i. Figure 4.10 shows these two tables as (A) and (B) respectively.

T_c Table

R1	0	0	1	4	2	16	20	. . . .	29	15	30	27	31	31
R2	0	0	1	2	4	16	18	. . . .	27	15	30	29	31	31
R3	0	0	2	16	8	10	17	. . . .	23	30	29	15	31	31
. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . .	. . . .	. . .	. . .	. . .	. . .	. . .	. . .
R960	.	.	.	.	.	.	.	. . . .	.	.	.	.	.	.

(A)

T_i Table

R1	2	5	16383	.	.
R2	3	16382	1	.	.
R3	4	16383	2	.	.
. . .	. . .	. . .	. . .	. . .	. . .
R960	.	.	.	.	.

(B)

Figure 4.10: Compressing the table T into two smaller tables: T_c and T_i.

Each row in the table T_c represents a unique pattern of the original table T. On the other hand, the equivalent row in the table T_i represents a list of rows numbers where the pattern is located in the original table T. As an example, consider the third pattern in the table T_c. The equivalent row in the table T_i, i.e. the third row, holds a list of rows numbers where the third pattern is located in the original T table. The memory space required to host each of the tables T_c and T_i is 960×64 (61440) and $960 \times 30 \times 2$ (57600) respectively. By using such compression method, the following optimizations are gained:

(1) The required memory space to host the tables T_c and T_i is 61440 + 57600 = 119040 bytes. The compression ratio is 119040/1048576 $\times$ 100%= 11.35%. This much better than the space required to host the original table T.

(2) The search algorithm is optimized as well. The new compressed table T_c requires 960 pattern comparisons in worst case; and a list of rows numbers is directly obtained from the table T_i. While, the table T requires 16384 pattern comparisons in order to collect

a list of rows numbers.

(3) The table T_c and T_i are constructed offline. This means constructing these two table is not a time consuming issue.

Referring to **(C)**, which is about the search algorithm. The proposed method of compression introduced the table T_c, which is the source of patterns that should be searched for patterns matching. This table holds static data that does not change in the execution time. Hence, the *Perfect Hashing Algorithm* can be employed to optimize the search algorithm. The perfect hashing algorithm process is illustrated in Fig. 4.11.

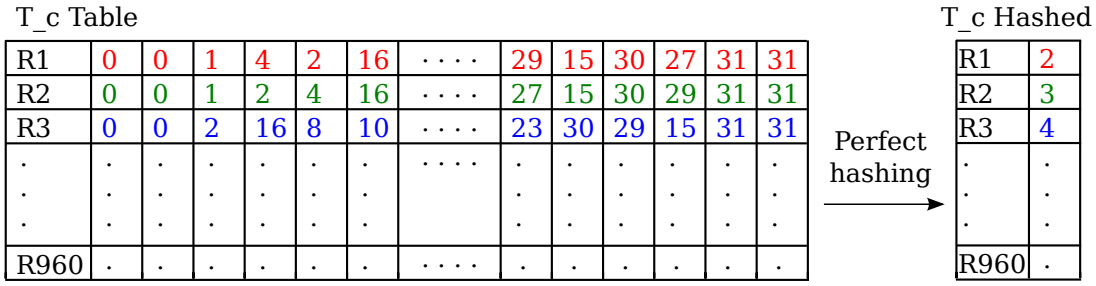

Figure 4.11: Calculating the hash value for each row in the table T_c.

In order to adopt the perfect hashing technique, we have to:

(1) Construct a new table, which we denote as $T_c - hashed$. This table is a hashed version of the table T_c.

(2) Each collected pattern should be hashed before it can be compared against the table $T_c - hashed$.

The table $T_c - hashed$ is constructed offline by calculating the hash value for each row in the table T_c. The hash value should be in the range of [1-960]. The size of the $T_c - hashed$ is 960 $\times$ 2=1920 bytes. The new memory space required is 119040 + 1920 = 120960 bytes. It is clear that the efficiency of the search algorithm is optimized because it requires $O(1)$

operation to locate a pattern in the table T and load the equivalents list held by the table T_i.

The required number of patterns to extract a unique value for the control word P is a challenge, as stated in **(D)**. It is important to know the max number of patterns required to extract a unique value for the P control word. The statistical analysis that we conduct using the table T_i shows that in worst cases, 32 patterns should be collected to extract one value for the control word P. In best cases, 2 patterns are good enough to achieve this. The curves plotted in Fig. 4.12 represent the relation between the number of patterns to be collected and the number of candidates for the control word P in best and worst cases.

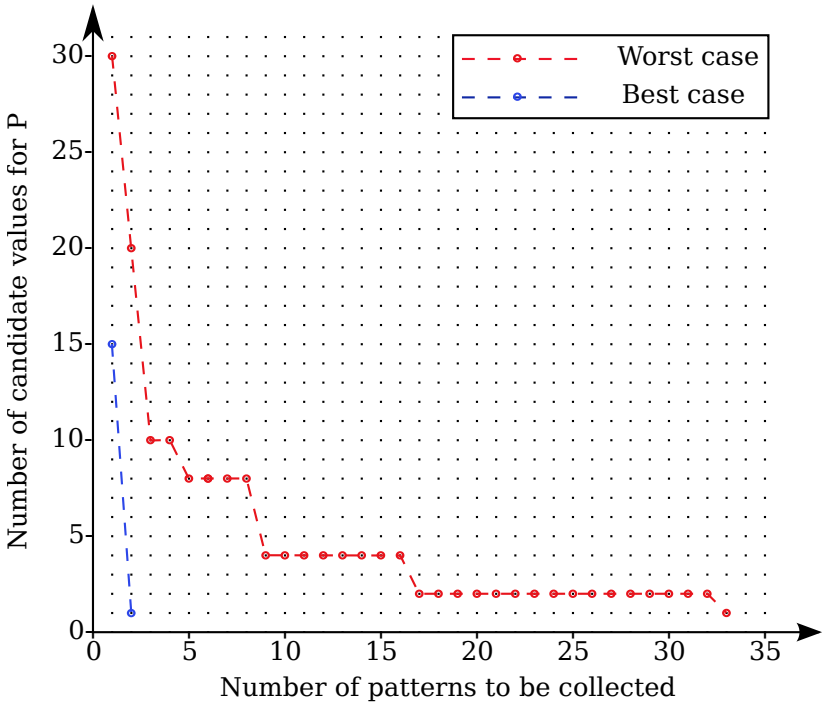

Figure 4.12: Maximum and minimum number of required patterns to find P.

It is clear that collecting 32 patterns (2048 hops) is impractical all the time. To deal with this problem, we propose a solution that relies on the curves plotted in the graph of Fig. 4.12. The red color curve represents the required number of patterns in worst case. Regarding the red curve, observe the period from 4 to 5 on the x-axis. Increasing the number

of patters to 5 helps by dropping 2 candidates values out of 10. This reduction does not exceed 50% of the total number of candidates. The same argument holds for the periods of: 8-9, 16-17, and 32-33. Moreover, the percentage of reduction does not exceed 50% over all the period from 3 up to 32 on the x-axis. We base on this fact to deal with the problem. Our proposed method is explained by the following example. Consider the case when the number of patterns is 4 and the number of candidates is 10. And we don't want to wait until a new pattern is collected and false values are dropped as illustrated by Fig. 4.13 (A). The following steps are applied to follow the Bluetooth hopping sequence by using three earliest collected patterns:

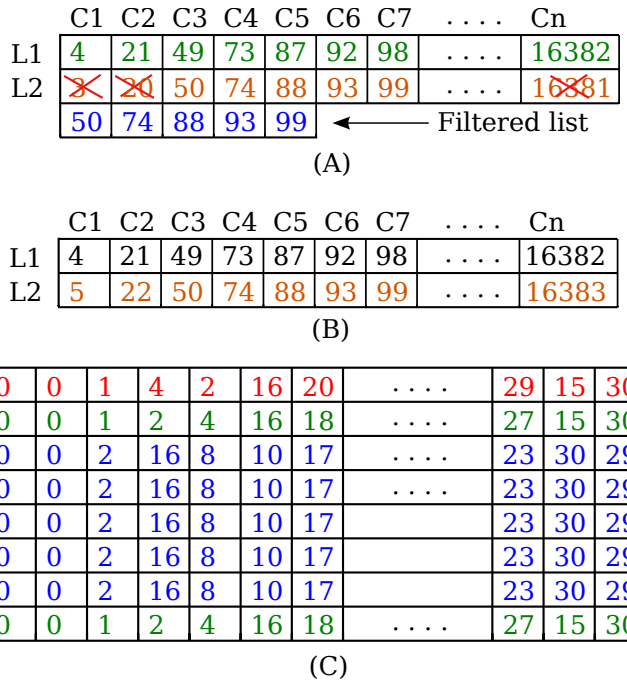

Figure 4.13: Using fewer patterns to hop with Bluetooth.

(1) Increment numerical values in the list of candidates by one, as illustrated by Fig. 4.13 (B).

(2) Load the equivalent patterns from the table T_c, as illustrated by Fig. 4.13 (C).

(3) Follow the dominating hopping pattern in blue color shown in the table of Fig. 4.13 (C).

We can conclude that 3 patterns are good enough to follow the frequency hopping of the targeted Bluetooth.

In this chapter, we discussed the proposed attacking method for for the hop selection kernel. Attacking each part of the kernel were explained individually. In sections 4.1, 4.2, and 4.3 the attacking methods of Part A, B, and C were discussed respectively. The discussion of the attacks methods include show the main parameters that are intended to be exposed from individual parts. In section 4.4, a simple attacking scenario was discussed to demonstrate the attacking process of the whole kernel. In section 4.5, the main implementation challenges were reviewed with some proposed solutions to overcome the challenges.

Chapter 5

Conclusion and Future Work

The Bluetooth technology is an emerging communication standard that is suitable for short range wireless connections. This wireless system is characterized by: low cost, low power consumption, and simplicity of use. Because of its attractive features, the Bluetooth wireless system becomes one of the popular and widely used technologies. Security and confidentiality of wireless data is one of the central issues that wireless systems developers concern about. The Bluetooth system relies on two techniques to maintain the confidentiality of wireless packets: first the encryption and second the Frequency Hopping technique. Bluetooth utilizes a stream cipher, called E_0, to encrypt wireless packets. The cryptanalysis of this cipher is studied thoroughly in the literature. On the other hand, the Frequency Hopping technique is rarely studied by researchers. The hopping technique does not add any kind of encryption to wireless packets; however, it enables the Bluetooth device to access wireless channels in a pseudo random fashion. For each wireless packet, a wireless channel is specified for transmission.This kind of random selection, makes it difficult for an attacker to predict the next channel to be utilized by Bluetooth. Hence, capturing wireless packets is not a trivial job.

In this work, we mainly focused on the Frequency Hopping mechanism and specifically examined the Hop Selection kernel that masters the hopping behaviour of Bluetooth. We

analysed the kernel hardware by partitioning it into three parts. This kind of modeling is essential to understand the selection kernel operation. Based on this modeling scheme, we propose an attacking method for the Hop Selection kernel. The propose method shows the possibility to expose the value of the Bluetooth clock that is used to predict its hopping sequence. Therefore, capturing wireless packets is possible.

In chapter 2, we presented a useful background and definitions about the Bluetooth wireless system. This background is necessary the analysis of the Hop Selection kernel that is presented in chapter 3. In chapter 2, we discussed the spectrum of the Bluetooth radio system, the frequency range, channels definition, and bandwidth spacing. The definition of the Bluetooth Hop Selection kernel is presented as well. The two essential parameters for the selection kernel are defined. The Bluetooth network topology is reviewed in chapter 2.

In chapter 3, we analysed the Hop Selection kernel hardware. We started by defining the internal components of the kernel and the input parameters for each constituting component. To understand the kernel function, we propose partitioning the kernel into three parts. This modeling scheme is useful for understanding the role of kernel parts individually. Furthermore, we mathematically expressed the operation of each part.

In chapter 4, we presented the proposed attack for the Hop Selection kernel. The proposed attacking method relies on the kernel analysis of chapter 3. After discussing the possible attack for each part of the kernel, we presented a simple scenario that showed how to crack the whole selection kernel. Cracking the selection kernel exposes the internal value of the Bluetooth clock. Which can be exploited to predict the Bluetooth hopping sequence and, hence, capturing wireless packets. In chapter 3, the simulation software is discussed. This software is constructed in Matlab environment. The implementation challenges and the proposed solutions were discussed at the end of chapter 4 as well.

5.1 Future Work

With respect to the future work, we aim to further investigate the following topics:

5.1.1 Implementing the proposed attacking on real hardware

In order to implement the proposed attacking method on real hardware, there are two main options: (1) Implementing the propose method using some experimental electronic devices that are available in the laboratory. An example of such device is the Universal Software Radio Peripheral (USRP) device. The available models are: USRP-1 and USRP-N200. The second one is capable of monitoring up to 50 MHz of channels bandwidth; and this makes it a good option for jamming and monitoring wireless channels. The operation of the USRP devices can efficiently mastered by the GNU radio software package.

(2) The second option is to use some commercial hardware such as the Ubertooth device. The primary analysis show that at least 4 Ubertooth devices are required to monitor part of the Bluetooth hopping sequence. Then, the monitored sequences can be invested by the proposed attacking method.

5.1.2 Investigating the possibility of jamming 15 channels of the Bluetooth space

The proposed attacking method suggests reducing the value of the input parameter to 32. This is conducted by jamming 47 channels of the defined 79 channels for Bluetooth. The 32 is preferred number for our attacking method, since it can be expressed in the form of 2^m. There is another useful number that can be expressed in the form of 2^m, which is 64. In the future work, we investigate the possibility of jamming 15 channels to reduce the value of N down to 64. This case is more efficient than the case of jamming 47 channels.

5.1.3 Exposing the MAC address and the first 6 bits of the clock

The proposed attacking method in this work relies on some useful results obtained by Spill [6]. The Spill proposed a method to expose the device MAC address and the first 6 bits of the clock. In the future work, we focus on the possibility of extracting these two essential parameters from the collected hopping pattern. In other words, there will be no need to adopt Spill methods any more.

5.1.4 Suggestions for avoiding the Hopping Sequence attack

In our future work, we concern to find a suitable solution for preventing the presented Frequency Hopping attack as well. Toward this goal, there are mainly two possible directions that we can go with:

(1) Modifying the hardware components consisting the Hope Selection kernel.

(2) Modifying the Bluetooth software is another solution that helps the Bluetooth system to avoid such security threat.

The solution of modifying some hardware components inside the kernel can be further studied to avoid such problem in next generations of the Bluetooth adapters. In other words, it is limited to the new generation of Bleutooth products.

With respect to the Bluetooth devices that are currently in use, the hardware solution is impractical one. However, modifying the Bluetooth software can be a more feasible solution. One of the sensitive components that can be modified by software is the clock of the Bluetooth device. The Bluetooth device frequently changes the value of the clock for synchronization purposes with the piconet master device. This capability of updating the clock value can be invested to prevent frequency hopping attack. To be more specific, the piconet master device can be programmed to change the value of its clock in a random fashion and inform the piconet slave device(s) to synchronize their with the master device. In this way, exposing the

master device clock value, by an attacker, is worthless since the clock value can be changed any time causing the attacker to lose the synchronization and failure to capture wireless packets.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] 802.15.1-2005 IEEE Standard Part 15.1: *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs).*

[2] Bluetooth Special Interest Group (SIG), See the "Our History": http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx [Accessed Oct 2011].

[3] Jaap C. Haartsen; *The Bluetooth Radio System*, IEEE Personal Communications, February 2000.

[4] Jaap C. Haartsen and Stefan Zurbes; *Frequency Hop Selection in the Bluetooth Radio System*, Spread Spectrum Techniques and Applications, 2002 IEEE Seventh International Symposium on, Czech Rep. 2002.

[5] Petar Popovski and Hiroyuki Yomo and Ramjee Prasad; *Strategies for adaptive frequency hopping in the unlicensed bands*, IEEE Wireless Communications Journal, Vol. 13, No. 6, Pages 60-67, ISSN 1536-1284, Dec 2006.

[6] Spill, Dominic and Bittau, Andrea; *BlueSniff: Eve meets Alice and Bluetooth*, Proceedings of the first USENIX workshop on Offensive Technologies, Boston, MA 2007, publisher: USENIX Association, Berkeley, CA, USA.

[7] Yaniv Shaked and Avishai Wool; *Cracking the Bluetooth PIN*, In Proceedings of the 3^{rd} international conference on Mobile systems, applications, and services (MobiSys '05). ACM, New York, NY, USA, 39-50.

[8] Morsi, K. and Xiong Huagang and Gao Qiang; *Performance estimation and evaluation of Bluetooth frequency hopping selection kernel*, Pervasive Computing (JCPC), 2009 Joint Conferences on, Pages 461-466, IEEE Dec 2009.

[9] Haataja, K.; Toivanen, P.; *Two practical man-in-the-middle attacks on Bluetooth secure simple pairing and countermeasures*, Wireless Communications, IEEE Transactions on, Vol. 9, No.1, pp.384-392, January 2010.

[10] Hypponen, K.; Haataja, K.M.J.; *Nino man-in-the-middle attack on bluetooth secure simple pairing* Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on , vol., no., pp.1-5, 26-28 Sept. 2007

[11] Haataja, K.; Toivanen, P.; *Practical Man-in-the-Middle Attacks Against Bluetooth Secure Simple Pairing* Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference on , vol., no., pp.1-5, 12-14 Oct. 2008

[13] Levy, O. and Wool, A.; *Uniform Framework for Cryptanalysis of the Bluetooth E_0 Cipher*, First International Conference on Security and Privacy for Emerging Areas in Communications Networks, 2005. SecureComm 2005.

[14] Yi Lu and Willi Meier and Serge Vaudenay; *The Conditional Correlation Attack: A Practical Attack on Bluetooth Encryption*, Advances in CryptologyCRYPTO 2005, volume 3621 of Lecture Notes in Computer Science, Springer-Verlag.