CHAPTER IV

SYSTEM DESIGN AND IMPLEMENTATION

4.1 INTRODUCTION

The previous chapter highlighted the system development model, materials and tools needed for the system construction, and the operating system needed to install the SDR software. This chapter explains in detail the process of the system design and implementation. The system's main design is to build an SDR system that acts a TX system, and another SDR system that acts as an RX system. Figure 4.1 shows the entire diagram for the proposal system.

Figure 4.1: The entire proposed system.

Figure 4.2 shows that the system can have another design, where every sensor node has two interfaces, a ZigBee and an RTL-SDR interfaces. After WSN nodes gain secret key by RTL-SDR dongle, the WSN nodes will shut down the RTL-SDR interfaces or put them to sleep and use ZigBee for data exchange. We will use the first proposal, though, as it is easy to expand the system by duplicating the WSN base station.

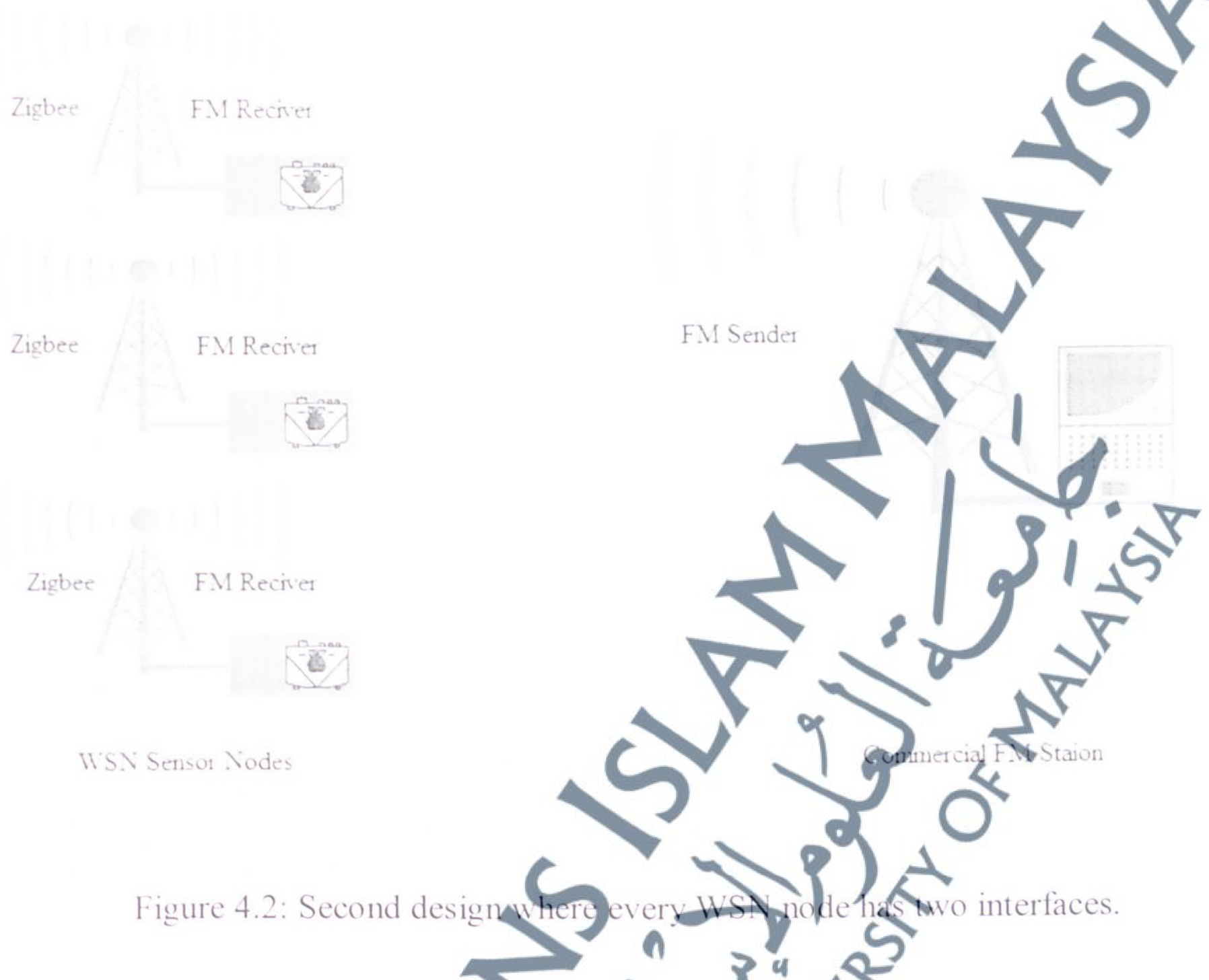

Figure 4.2: Second design where every WSN node has two interfaces.

4.2 SYSTEM DESIGN

The thesis' main contribution is to hide the secret key within the commercial FM band. The system design will simulate the commercial FM station so it can be used to send the key to remote WSN base station. Figure 4.1 showing the design on the FM Station that is acting as a TX, WSN Base Station that is acting as an RX.

There are three layers for each system, the first layer is a python code written to encrypt the symmetric key and spread the code using spread code. The python code was built over GNURadio blocks, XOR Encryption/Decryption, Packet Encoder/Decoder and GMSK Mod/Demod. The TX system will be sinked to a HackRF ONE, while RX

will receive the signal using the RTL-SDR dongle. Figure 4.3 shows a full design structure for the TX and RX systems.

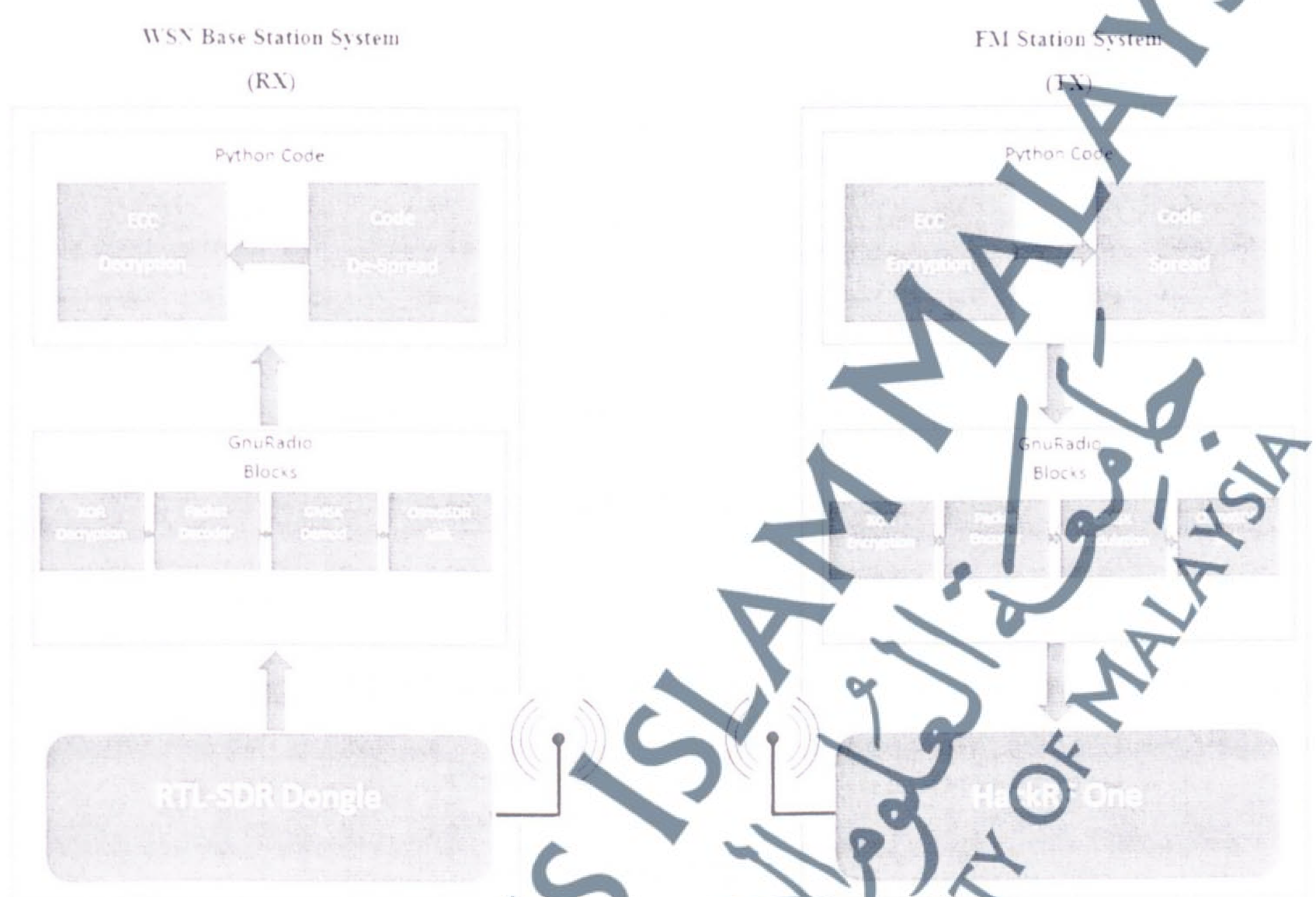

Figure 4.3: Full system design.

4.3 SYSTEM STRUCTURE AND DESIGN

The project design will be divided to two main modules:

i. Module 1 for the commercial FM system.

ii. Module 2 for the WSN base station.

4.3.1 MODULE 1: COMMERCIAL FM SYSTEM

This module will be acting as the commercial FM station (TX) that will be responsible for symmetric key broadcast using the algorithm as follows:

1. Initial phase: for the first time only, a temporary ECC private key will be generated from reading current hour and current user name and combine them together to create a private key.
2. From the private key, a public key will be generated.
3. WSN symmetric key will be encrypted using the generated ECC public key. As the ***First*** layer of key security.
4. The encrypted key will be spread using a spread code. This will be the ***Second*** layer of key security and will be a phase to the following sub-steps:
    a. There will be 4 different spread code saved within the TX host.
    b. The spread code is chosen depending on time.
    c. Python code will read the minutes' value for local time.
    d. Spread Code Number1 (SC#1) will be chosen if the minutes' value is between 00 and 15.
    e. SC#2 will be chosen if the minutes' value is between 16 and 30.
    f. SC#3 will be chosen if the minutes' value is between 31 and 45.

g. SC#4 will be chosen if the minutes' value is between 46 and 59.

Figure 4.4 shows spread code selection depends on time.

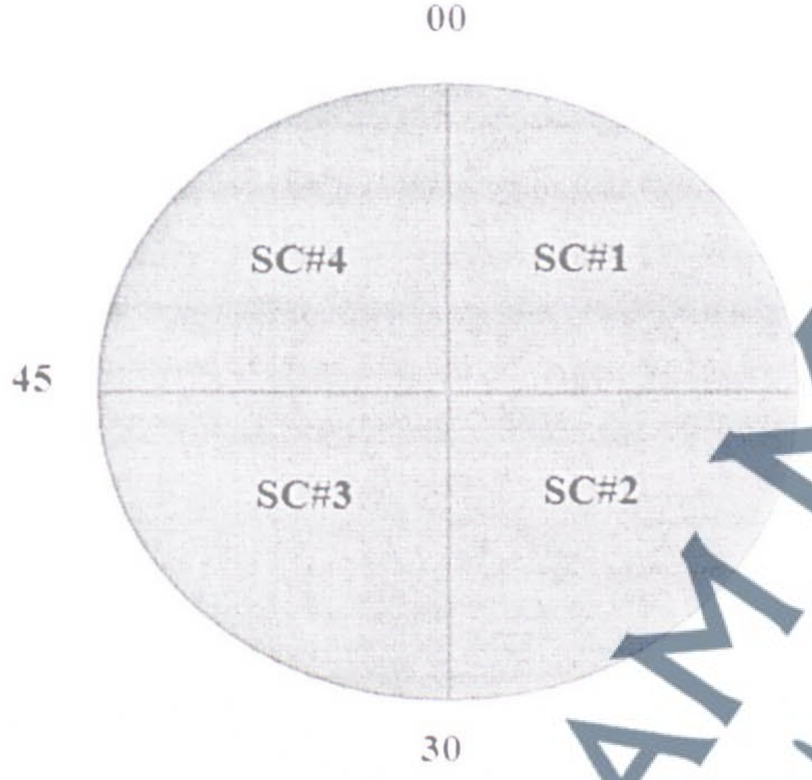

Figure 4.4: Spread code chosen depends on time.

5. The outcome of the spread phase will be encrypted using XOR encryption. This is the ***Third*** layer of security.

6. The outcome of the XOR phase will be encapsulated with a fake code. The fake code will be added at the start and at the end of the code. Each start and end code is different.

7. The encapsulated result will be injected within the fake code, this is the ***Fourth*** security layer.

8. The result will be modulated on the 80 KHZ, this is the ***Fifth*** security layer. Figure 4.5 shows the five layers of secrecy technique that the system provides.

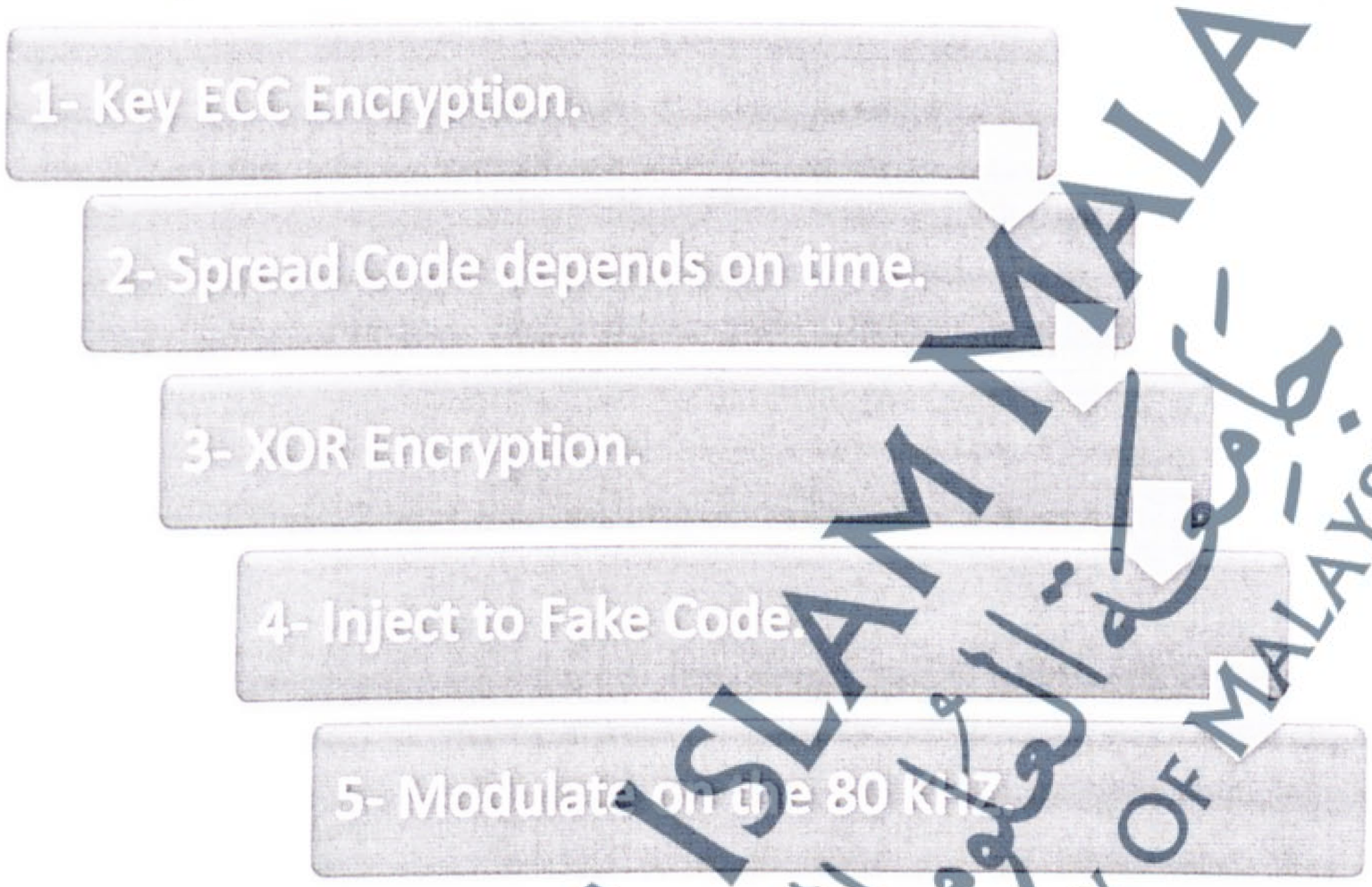

Figure 4.5: WSN Secrecy Technique Security Layers.

9. Then modulated within the commercial FM station band. Figure 4.6 shows the module for commercial FM station (TX system) and symmetric key broadcast method.

4.3.2 MODULE 2: WSN BASE STATION

In this phase WSN base station will recover the symmetric key by the following algorithm:

1- Initial phase: for the first time only, a temporary ECC private key will be generated from reading current hour and current user name and combine them together to create private key.

2- The WSN Base station will keep listening to the specified station.

3- 4th harmonic 80 KHz will be demodulated.

4- When data collected, fake code will be conducted from start and the end of the key.

5- XOR decryption.

6- Again time will be checked to get the spread code used at the time of send, to retrieve the key.

7- The key will be decrypted using ECC private key that was gained in the initial phase.

8- The Key will be sent by ZigBee to all near WSN nodes. Figure 4.7 shows the module two for WSN Base Station (RX system) key gain method.

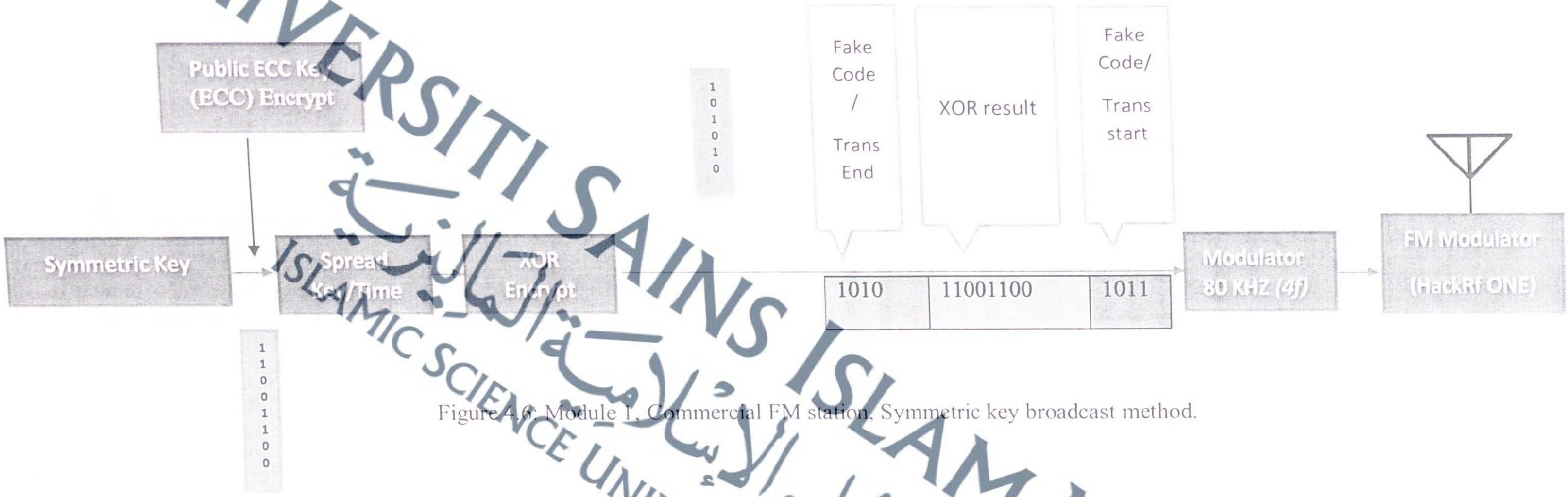

Figure 4.6: Module 1, Commercial FM station, Symmetric key broadcast method.

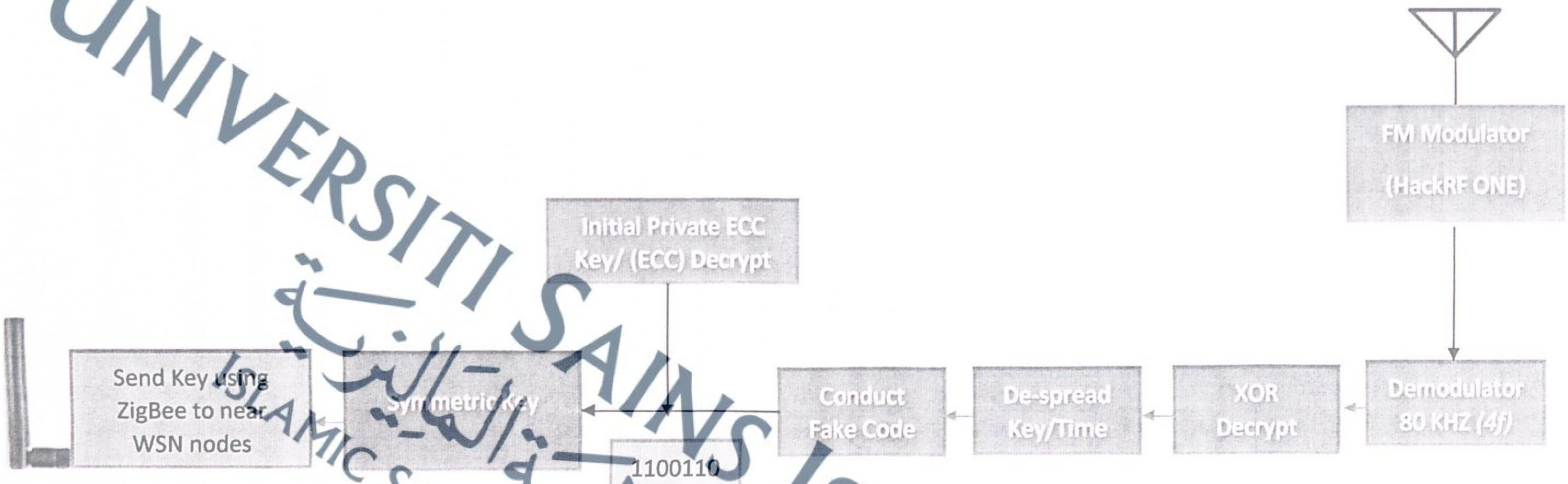

Figure 4.7: Module 2, WSN Base Station, Keygain method.

4.4 SYSTEM IMPLEMENTATION

The implementation starts using GNURadio live 3.7.8 live distribution, as it has the most updated GNURadio and its dependencies. Next step was to install it on a USB drive using "LiLi USB Creator", as it is the most stable software to create a bootable GNURadio live 3.7.8 USB drive. We have used a USB 3 thumb drive with a 32GB, and we have created a 4GB of persistent space so changes will be saved in case of restart or shutting down the system. Both TX and RX systems are created in the same way, but we have made another WSN Base station using Raspberry Pi 2.

4.4.1 WSN TX MODULE

First an initial private key will be created, the private key will be a combination of the current Linux username and the current date (Day, Month, Year), from initial key a public key will be created using SECCURE function. WSN Public key will be generated using the WSN Private Key, the Private Key will be encrypted using the initial key.

The key will be converted to binary, and encapsulated with a Fake code. The encapsulated result will be spread using a spreading code that will be chosen from a file, the chosen code will be one out of four codes, the chosen code will be depend on time. The result will be XOR with a simple binary. The result will be encoded with a packet encoder that will be modulated later with a GSMK. The modulated signal will be sinked to HackRF One.

The TX module will be executed in sequence, the following pseudo code describes the process of transmitting the symmetric key:

Fake_Prefix = 01101010101010101010

Fake_Suffix = 10100010100101011010

XOR_Code = 10110110

Date = get current date ()

Symmetric_key = Syemmetrickey

Username = get current username () ## whoami UNIX command

Initial_Private_Key = username+date

Initial_key_Public = SECCURE (Initial_Private_Key)

Encrypt_Symmetric_Key = SECCURE.Initial_key_Public (Symmetric_Key)

Binary_Code = Convert_to_Binary (Encrypt_Symmetric_Key)

Get time (minutes)

If minutes <= 15

Read fake_code_1

Spread_Code = Binary_Code ⊕ fake_code_1

If minutes > 15 and minutes <= 30

Read fake_code_2

Spread_Code = Binary_Code ⊕ fake_code_2

If minutes > 30 and minutes <= 45

Read fake_code_3

Spread_Code = Binary_Code ⊕ fake_code_3

Else minutes > 45 and minutes <= 59

Read fake_code_4

Spread_Code = Binary_Code ⊕ fake_code_4

XOR_Encrypt = Spread_Code ⊕ XOR_Code

Fake_result = Fake_Prefix+XOR_Encrypt+Fake_Suffix

Encoder = Packet_Encoder (Fake_Result)

Add GMSK.Encoder () With Sine 80 KHz

Sink to HackRF

4.4.2 WSN RX MODULE

Like TX module, an initial private key will be created, the private key will be a combination of the current Linux username and the current date (Day, Month, Year). From the initial key, a public key will be created using SECCURE function. The WSN base station status will be on listening mode waiting for the transmission to start.

When transmission start is detected, gaining state will last for five to ten seconds only because of the repeating issue, after that the receiving will be stopped and the result will be saved in a file. The fake code suffix and prefix will be deleted and the encapsulated code will be conducted. Code will be de-coded using packet decoder, and then demodulated using the GMSK Demod block.

The result will be XOR'd with the simple XOR code, the current time will be read for the current minutes to choose the spread code that will be used to recover the binary key. The binary key will be converted to ASCII and decrypted using the initial private key.

The gained symmetric key can be sent to nearby WSN nodes using ZigBee. Figure 4.8 shows the flow chart between the TX and RX systems.

RX module will be executed in a reverse sequence to TX, the following pseudo code describes the process receiving the symmetric key:

Fake_Prefix = 01101010101010101010

Fake_Suffix = 10100010100101011010

XOR_Code = 10110110

Date = get current date ()

Username = get current username () *## whoami UNIX command*

Initial_Private_Key = username+date

Put BaseStation to Listen_State.Filter (80KHZ)

If receive_data = true

Timer_coutdown = 10 seconds

GMSK.Decoder ()

Packet_decoder (receive_data)

Save recieve_data to recieve_file

When timer_coutdown = 0

Stop receive_data

Open recieve_file

Get time (minutes)

XOR_Decrypt = open (recieve_file) ⊕ XOR_Code

Fake_Prefix = 01101010101010101010

Fake_Suffix = 10100010100101011010

XOR_Code = 10110110

Date = get current date ()

Username = get current username () ## whoami UNIX command

Initial_Private_Key = username+date

Put BaseStation to Listen_State.Filter (80KHZ)

If receive_data = true

Timer_coutdown = 10 seconds

GMSK.Decoder ()

Packet_decoder (receive_data)

Save recieve_data to recieve_file

When timer_coutdown = 0

Stop receive_data

Open recieve_file

Get time (minutes)

XOR_Decrypt = open (recieve_file) ⊕ XOR_Code

Sink Decrypt_Symmetric_Key to ZigBee ##Optional

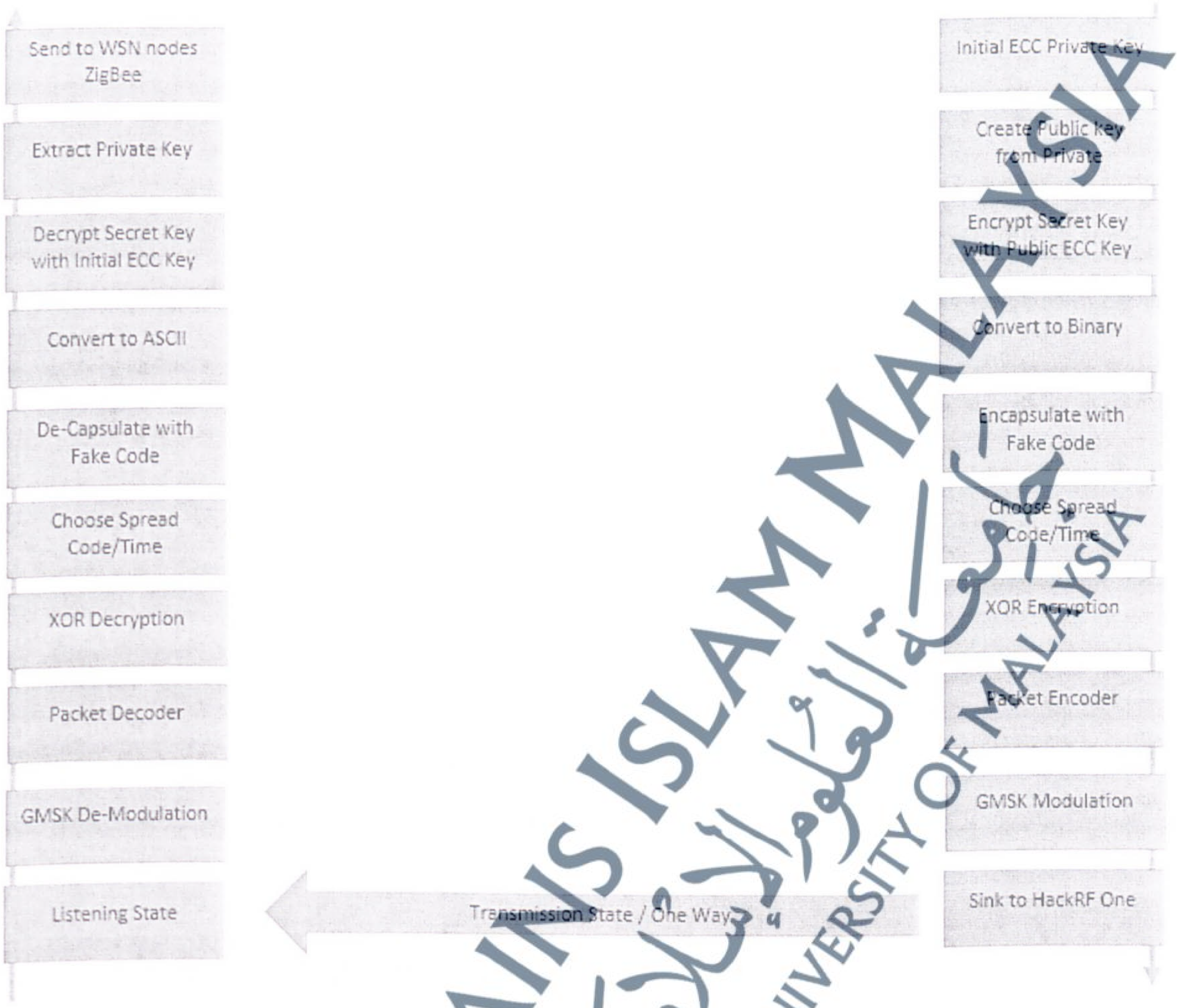

Figure 4.8 Flow chart between the TX and RX systems.