# BIBLIOGRAPHY


Ahmed, N., Kanhere, S. S., & Jha, S. (2005). The holes problem in wireless sensor networks: a survey. *ACM SIGMOBILE Mobile Computing and Communications Review*, *9*(2), 4-18.

Barca, C., Neamtu, C., Popescu, H., Dumitrescu, S., & Sandu, A. S. (2013, June). Implementation of RDS platform solutions for an emergency system. In Electronics, Computers and Artificial Intelligence (ECAI), 2013 International Conference on (pp. 1-4). IEEE.

Bos, J. W., Halderman, J. A., Heninger, N., Moore, J., Naehrig, M., & Wustrow, E. (2014). Elliptic curve cryptography in practice. In *Financial Cryptography and Data Security* (pp. 157-175). Springer Berlin Heidelberg.

Brumley, D., & Boneh, D. (2005). Remote timing attacks are practical. *Computer Networks*, *48*(5), 701-716.

Campos, R., Duarte, R., Sousa, F., Ricardo, M., & Ruela, J. WiFIX: A New Solution for 802.11-based Wireless Mesh Networks.

Cellular Networks Benefits and Shortfalls For Emergency Communications. (2005). Retrieved December 20, 2015, from http://www.cpaccarolinas.org/docs/cellular_communications.pdf

Chasmai, M., Das, R., Garg, R., & Kaur, M. Wireless Sensor Network for snow-met data monitoring in Indian Himalayas-A Review.

Chen, Y., Lymberopoulos, D., Liu, J., & Priyantha, B. (2012, June). FM-based indoor localization. In *Proceedings of the 10th international conference on Mobile systems, applications, and services* (pp. 169-182). ACM.

Dimitrievski, A., Pejovska, V., & Davcev, D. (2011). Security Issues and Approaches in WSN. *Department of computer science, Faculty of Electrical Engineering and Information Technology*.

Fisher, R., Ledwaba, L., Hancke, G., & Kruger, C. (2015). Open hardware: a role to play in wireless sensor networks?. *Sensors*, 15(3), 6818-6844.

Kang, W., Stankovic, J. A., & Son, S. H. (2008). On Using Weather Information for Efficient Remote Data Collection in WSN. In *Workshop on Embedded Networked Sensors*.

Kim, Y. H., Lee, H., Lee, D. H., & Lim, J. (2006, January). A key management scheme for large scale distributed sensor networks. In *Personal Wireless Communications* (pp. 437-446). Springer Berlin Heidelberg.

Kong, X., Zhang, D., & Ahmed, M. (2010). A Software-Defined Radio System for Intravehicular Wireless Sensor Networks. *International Journal of Digital Multimedia Broadcasting, 2010*.

Lan, K., & Li, M. (2010). Feasibility study of using FM radio for data transmission in a vehicular network. *2010 International Computer Symposium (ICS2010)*.

Laskar, T., & Jena, D. (2012). A Survey on Key Management Issues in WSN. *International Journal of Engineering and Innovative Technology (IJEIT), 1*.

Lenstra, A. K., Kleinjung, T., & Thomé, E. (2013). Universal security. In *Number theory and cryptography* (pp. 121-124). Springer Berlin Heidelberg.

Li, Y., He, Z., Voigt, T., & Leidelöf, S. (2009). A software radio-empowered sensor network. In *9th Scandinavian Workshop on Wireless Adhoc Networks (Adhoc'09)*.

Qin, Y., & He, Z. (2011). A communication Monitor for Wireless Sensor Networks based on Software Defined Radio.

Radio Broadcast Data System • CBC/Radio-Canada. (n.d.). Retrieved December 20, 2015, from http://www.cbc.radio-canada.ca/en/reporting-to-canadians/sync/sync-issue-2-2012/radio-broadcast-data-system/

RDS Basics. (n.d.). Retrieved December 20, 2015, from http://www.2wcom.com/fileadmin/redaktion/dokumente/Company/RDS_Basics.pdf

Sayin, A. (2013). VHF/UHF Uplink Solutions for Remote Wireless Sensor Networks.

The Case for Elliptic Curve Cryptography. (2009, January 1). Retrieved from https://www.nsa.gov/business/programs/elliptic_curve.shtml

Torrieri, D. (2011). Code-Division Multiple Access. In *Principles of spread-spectrum communication systems* (pp. 365-463). Springer New York.

Vannucci, G., Bletsas, A., & Leigh, D. (2008). A software-defined radio system for backscatter sensor networks. *Wireless Communications, IEEE Transactions on*, *7*(6), 2170-2179.

Walters, J. P., Liang, Z., Shi, W., & Chaudhary, V. (2007). Wireless sensor network security: A survey. *Security in distributed, grid, mobile, and pervasive computing*, *1*, 367.

Wang, Y., Zhang, Y., Liu, J., & Bhandari, R. (2015). Coverage, Connectivity, and Deployment in Wireless Sensor Networks. In *Recent Development in Wireless Sensor and Ad-hoc Networks* (pp. 25-44). Springer India.

Xu, G., Shen, W., & Wang, X. (n.d.). Applications of Wireless Sensor Networks in Marine Environment Monitoring: A Survey. *Sensors*, 16932-16954.

Youssef, A., Krumm, J., Miller, E., Cermak, G., & Horvitz, E. (2005, March). Computing location from ambient FM radio signals [commercial radio station signals]. In *Wireless Communications and Networking Conference, 2005 IEEE* (Vol. 2, pp. 824-829). IEEE.

Zhang, D., Heo, U., You, K. S., & Choi, J. H. (2010). Resilient Security Protocol for Combating Replay Attacks in Wireless Sensor Networks. *The Journal of the Korea Contents Association*, *10*(7), 70-80.

# APPENDIX A

# TX SYSTEM PYTHON CODE

This appendix includes the python code that was developed to build the TX system, depending on the sequence of the proposed algorithm. Figure 1 shows the TX system built with GNURadio, the python code generated from it used to build the full TX system:

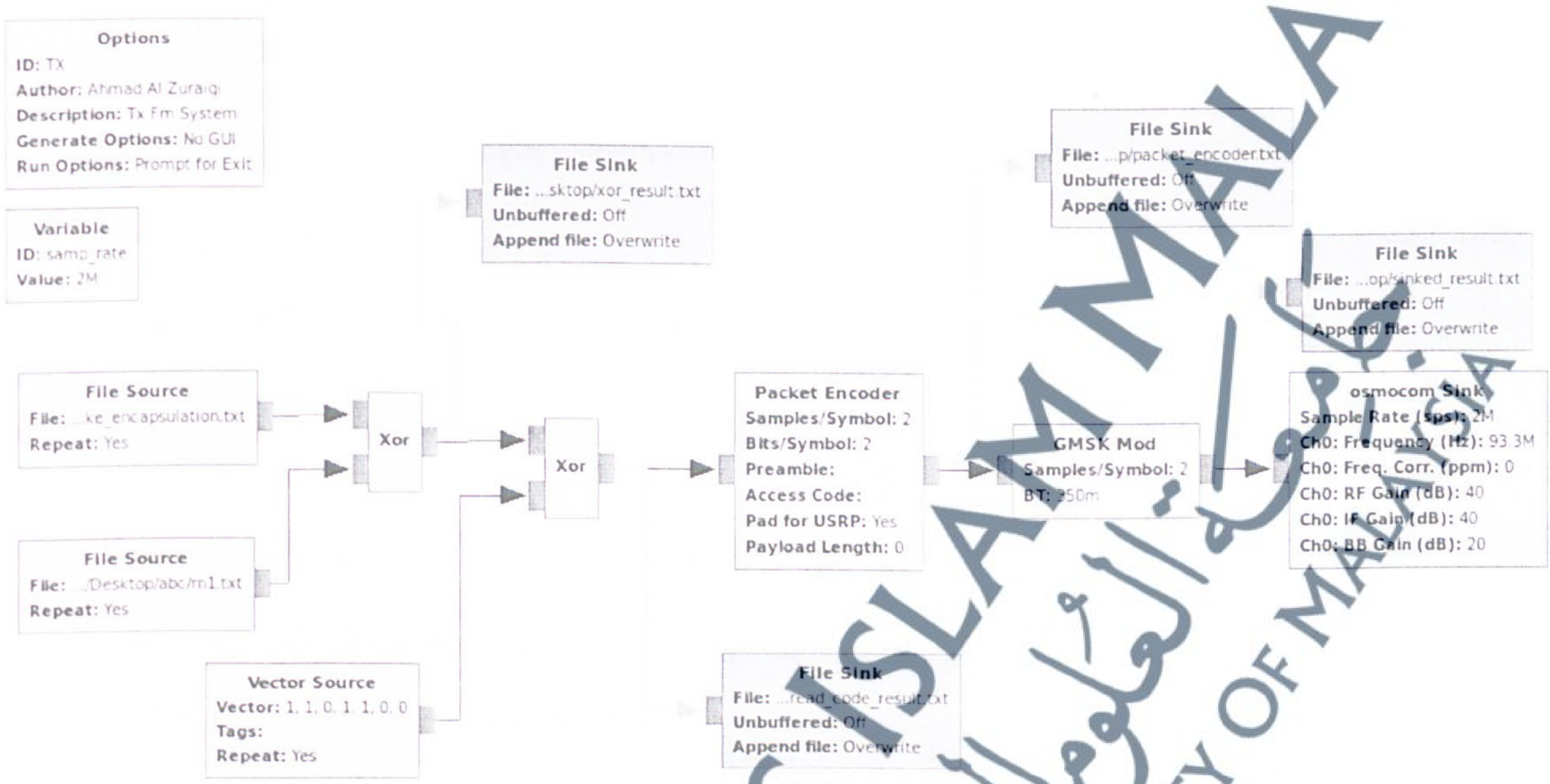


Figure A.1: GNURadio TX system blocks

## 1- GENERATE ECC INITIAL PRIVATE KEY

```
from time import localtime, strftime
time = strftime("%d-%Y-%M", localtime())
user = getpass.getuser()
private_initial_ecc_key = user + time
private_initial = 'private_initial_ecc_key.txt'
file = open( private_initial, 'w')
file.write(private_initial_ecc_key)
```

## 2- CREATE ECC PUBLIC KEY FROM THE INITIAL KEY

```
Private_file = open(private_initial)
i = Private_file.read()
```

```
public_key = str(seccure.passphrase_to_pubkey(private_initial_ecc_key))
Public_key_file = 'public_ecc.txt'
File2 = open(Public_key_file, 'w')
File2.write(ciphertext)
```

### 3- ENCRYPT SYMMETRIC KEY WITH ECC PUBLIC KEY

```
script, symmetric_file = argv, 'symmetric.txt'
txt2 = open(symmetric_file)
s = txt2.read()
script, public_file = argv, 'public_ecc.txt'
txt3 = open(public_file)
p = txt3.read()
encrypt_symmetric = 'encrypt_symmetric.txt'
encrypt= seccure.encrypt(s, p)
e = open(encrypt_symmetric, 'w')
e.write(encrypt)
e.close()
```

### 4- CONVERT FROM ASCII TO BINARY

```
script, l = argv, encrypt_symmetric
txt = open(l)
w = txt.read()
a = bin(int(binascii.hexlify(w), 16))
txt.close()
Ascii_to_binary_file= 'Ascii_to_Binary.txt'
d = open(j, 'w')
d.write(a)
```

### 5- ADD FAKE CODE (ENCAPSULATION)

```
prefix =
'0b1000000001000000001100000011000001110011011110100011000010111001001110100
0011000000110000010000000001000000'
```

```
suffix =
'0b1000000010000000011000000110000001100101011011100110010000011000000011000000
01000000001000000'

x = Ascii_to_binary_file

Fake_encapsulation_file = 'fake_encapsulation.txt'
with open(x, 'r') as src:
    with open(Fake_encapsulation_file, 'w') as dest:
        for line in src:
            dest.write('%s%s%s\n' % (prefix, line.rstrip('\n'), suffix))
```

## 6- SPREAD CODE SELECTION

```
t= strftime("%-M", localtime()) # read minute value from current time
time = int(t)
if 00<=time<15:
        script, spread_code = argv, '/spread_code_1.txt'
        txt = open(spread_code)
        s = txt.read()
        selected_spread_code = 'selected_spread_code.txt'
        d = open(selected_spread_code, 'w')
        d.write(s)

elif 15<= time <30:
        script, spread_code = argv, '/spread_code_2.txt'
        txt = open(spread_code)
        s = txt.read()
        selected_spread_code = 'selected_spread_code.txt'
        d = open(selected_spread_code, 'w')
        d.write(s)

elif 30<= time <45:
        script, spread_code = argv, '/spread_code_3.txt'
        txt = open(spread_code)
        s = txt.read()
        selected_spread_code = 'selected_spread_code.txt'
        d = open(selected_spread_code, 'w')
        d.write(s)

else:
        script, spread_code = argv, '/spread_code_3.txt'
        txt = open(spread_code)
        s = txt.read()
        selected_spread_code = 'selected_spread_code.txt'
```

```
d = open(selected_spread_code, 'w')
d.write(s)
```

# APPENDIX B

# RX SYSTEM PYTHON CODE

This appendix includes the python code that was developed to build the RX system, depends on the sequence of the proposed algorithm. Figure 1 shows the RX system built with GNURadio, the python code generated from it used to build the full RX system:

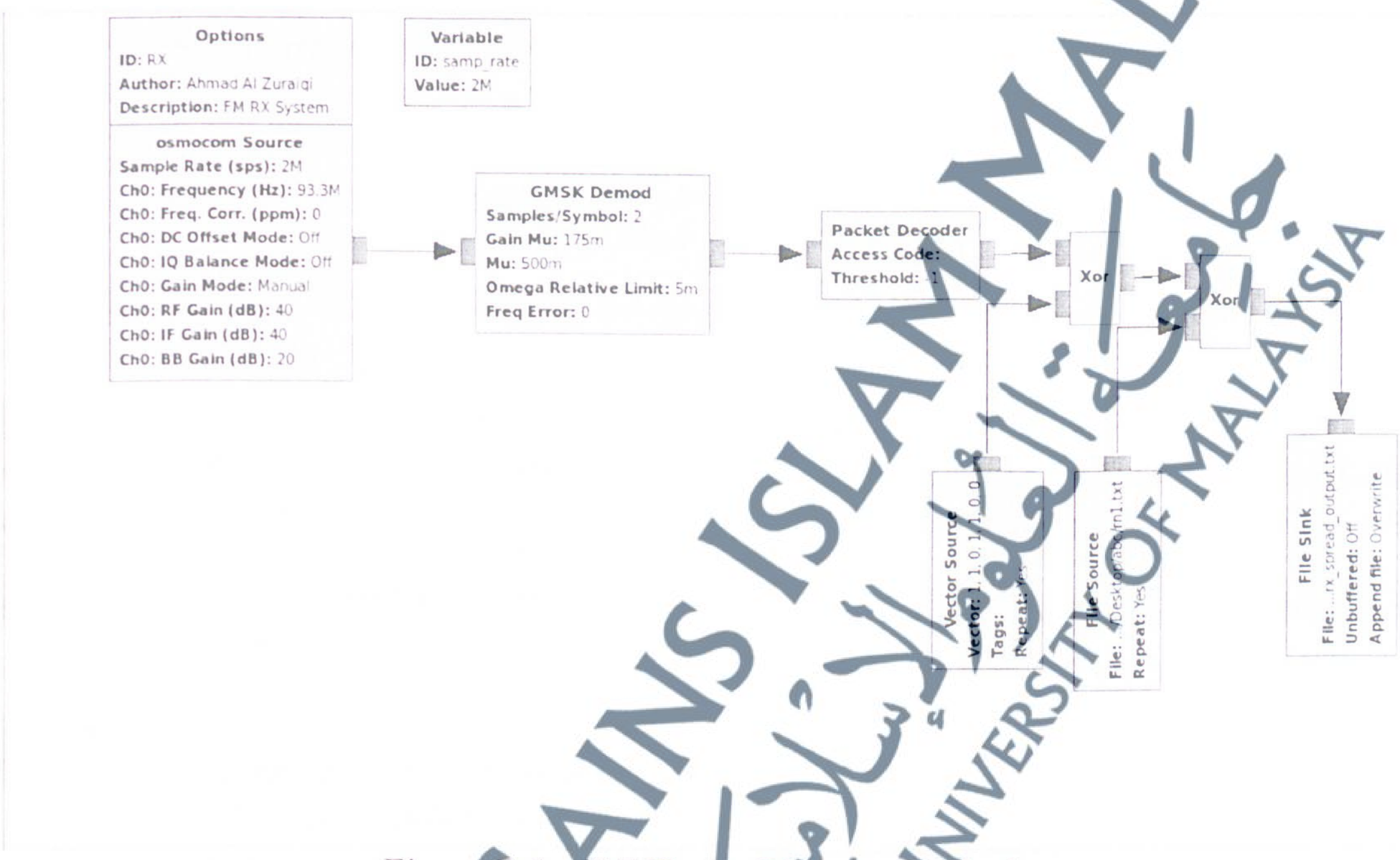


Figure B.1: GNURadio RX system Blocks

## 1- GENERATE ECC INITIAL PRIVATE KEY

```
from time import localtime, strftime

time = strftime("%d-%Y-%M", localtime())

user = getpass.getuser()

private_initial_ecc_key = user + time

private_initial =   'private_initial_ecc_key.txt'

file = open( private_initial, 'w')

file.write(private_initial_ecc_key)
```

## 2- SPREAD CODE SELECTION

```
time = strftime("%M", localtime()) # read minute value from current time
```

```
if 00<=time<15:
        script, spread_code = argv, '/spread_code_1.txt'
        txt = open(spread_code)
        s = txt.read()
        selected_spread_code = 'selected_spread_code.txt'
        d = open(selected_spread_code, 'w')
        d.write(s)

elif 15<= time <30:
        script, spread_code = argv, '/spread_code_2.txt'
        txt = open(spread_code)
        s = txt.read()
        selected_spread_code = 'selected_spread_code.txt'
        d = open(selected_spread_code, 'w')
        d.write(s)

elif 30<= time <45:
        script, spread_code = argv, '/spread_code_3.txt'
        txt = open(spread_code)
        s = txt.read()
        selected_spread_code = 'selected_spread_code.txt'
        d = open(selected_spread_code, 'w')
        d.write(s)

else:
        script, spread_code = argv, '/spread_code_3.txt'
        txt = open(spread_code)
        s = txt.read()
        selected_spread_code = 'selected_spread_code.txt'
        d = open(selected_spread_code, 'w')
        d.write(s)
```

### 3- FAKE CODE DECAPSULATION

```
prefix =
'0b1000000010000000011000000110000011100110111010001100001011100100111010000011000000110000010000000001000000'
suffix =
'0b1000000010000000011000000110000011001010110111001100100001100000011000001000000'
script, despread_result = argv, 'despread_result_file.txt'
txt = open(despread_result)
s = txt.read()
result = re.search('%s(.*)%s' % (start, end), s).group(1)
print(result)
binary_result= 'Binary_Result_file.txt'
```

```
f = open( k, 'w')
f.write(result)
```

## 4- CONVERT FROM BINARY TO ASCII

```
script, convert_file = argv, binary_result

txt = open(convert_file)

s = txt.read()

n = int(s, 2)

a = binascii.unhexlify('%x' % n)

k = 'Encrypted_key.txt'

f = open( k, 'w')

f.write(a)
```

## 5- ENCRYPT SYMMETRIC KEY WITH ECC PUBLIC KEY

```
script, private_key_file = argv, private_initial

txt2 = open(private_key_file)

Initial_Key = txt2.read()

script, k = argv, 'Encrypted_key.txt'

txt3 = open(k)

Encrypted_Key= txt3.read()

Secret_Key = seccure.decrypt(Encrypted_key, Initial_Key)

Print Secret_Key
```