20CYS205 – MODERN CRYPTOGRAPHY

Hight Cipher Implementation and Cryptanalysis

Submitted by

AMAL RITESSH A P	– CB.EN.U4CYS22005
ANANTH R	– CB.EN.U4CYS22006
ANASWARA SURESH M K	– CB.EN.U4CYS22007
ARUL SUJITH S	– CB.EN.U4CYS22008

Under the guidance of

Dr. Lakshmy KV and **Saurabh Shrivastava**
Assistant Professors
Amrita Vishwa
VidyapeethamCoimbatore

TIFAC-CORE IN CYBER SECURITY

AMRITA SCHOOL OF

ENGINEERING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

2023

Acknowledgement

First of all, we would like to express our gratitude to our Mentors, **Dr. Lakshmy KV and Saurabh Shrivastava** , Assistant Professors, TIFAC-CORE in Cyber Security, Amrita Vishwa Vidyapeetham, Coimbatore, for their valuable suggestions and timely feedbacks during the course of this major project. It was indeed a great support from them that helped me successfully fulfill this work

I would like to thank **Dr.M.Sethumadhavan**, Professor and Head of Department, TIFAC-CORE in Cyber Security, for his constant encouragement and guidance through-out the progress of this major project.

I convey special thanks to our friends for listening to our ideas and contributing their thoughts concerning the project. All those simple doubts from their part have also made us think deeper and understand about this work.

In particular, we would like also like to extend our gratitude to all the other faculties of TIFAC-CORE in Cyber Security and all those people who have helped us in many ways for the successful completion of this project.

TABLE OF CONTENTS

Introduction:

HIGHT is a 64-bit symmetric key light-weight block cipher suitable for low-resource device. HIGHT stands for 'HIGh security and light weigHT' and is developed by Korea 2005. HIGHT is an ISO/IEC international standard block cipher algorithm included in ISO/IEC 18033-3:2010 [ISO-HIGHT]. It has simple structure with use of basic arithmetic operation - XOR, addition/subtraction in modular 2**8, and circular shift rotation, and also without S-Box.

Notation and Terminology:

The following notation is used in the description of HIGHT encryption algorithm:

[^]	bitwise XOR
[+]	addition in modular 2**8
[-]	subtraction in modular 2**8
||	concatenation
<<<n	left circular shift rotation by n-bit in 8-bit value
P	plaintext
C	ciphertext
K	master key
WK	whitening key
SK	subkey
F0	round function 0
F1	round function 1
Xi	i-th byte of X
Xj,i	i-th byte of X in round j
di	intermediate status value in subkey scheduling

Key Generation:

HIGHT generates 128 subkeys and a whitening key, all derived from the 128-bit master key using a dedicated key scheduling algorithm. The algorithm is as follows.

Master key:

K= K15 || K14 || K13 || K12 || K11 || K10 || ... || K0

Whitening key algorithm:

This following algorithm is used to generate whitening keys

```
        for i = 0, 1, 2, 3:
                WKi = K(i+12)
        for i = 4, 5, 6, 7:
                WKi = K(i-4)
```

Subkey generation algorithm:

To generate 128 subkeys it is required to find 128 7bit ‘d’ values using the initial ‘s’ value.

Initial ‘s’ value:
s0 = 0, s1 = 1, s2 = 0, s3 = 1, s4 = 1, s5 = 0, s6 = 1

The following is the algorithm used to find 128 7bit ‘d’ values:

```
d0 = s6 || s5 || s4 || s3 || s2 || s1 || s0
for i = 1 to 127:
        s(i+6) = s(i+2) [^] s(i-1)
        di = s(i+6)||s(i+5)||s(i+4)||s(i+3)||s(i+2)||s(i+1)||si
```

Once all d values are generated we can proceed with the subkey generation

Subkey generation:

```
for i = 0 to 7:
        for j = 0 to 7:
                SK(16*i+j) = K(j-i mod 8) [+] d(16*i+j)
        for j = 0 to 7:
                SK(16*i+j+8) = K((j-i mod 8)+8) [+] d(16*i+j+8)
```

Encryption:

The encryption operation is as shown in Figure 1. The transformation
of a 64-bit block P into a 64-bit block C is defined as follows

```
(1) P = P7 || P6 || P5 || P4 || P3 || P2 || P1 || P0

(2) X0,0 = P0 [+] WK0,        X0,1 = P1,
    X0,2 = P2 [^] WK1,        X0,3 = P3,
    X0,4 = P4 [+] WK2,        X0,5 = P5,
    X0,6 = P6 [^] WK3,        X0,7 = P7.
```

```
(3) for i = 0 to 30:
      X(i+1),0 = Xi,7 [^] (F0(Xi,6)[+]SK(4*i+3)),  X(i+1),1 = Xi,0,
      X(i+1),2 = Xi,1 [+] (F1(Xi,0)[^]SK(4*i+2)),    X(i+1),3 = Xi,2,
      X(i+1),4 = Xi,3 [^] (F0(Xi,2)[+]SK(4*i+1)),  X(i+1),5 = Xi,4,
      X(i+1),6 = Xi,5 [+] (F1(Xi,4)[^]SK(4*i)),  X(i+1),7 = Xi,6.
    for i = 31:
      X(i+1),0 = Xi,0,     X(i+1),1 = Xi,1 [+] (F1(Xi,0)[^]SK124),
      X(i+1),2 = Xi,2,     X(i+1),3 = Xi,3 [^] (F0(Xi,2)[+]SK125),
      X(i+1),4 = Xi,4,     X(i+1),5 = Xi,5 [+] (F1(Xi,4)[^]SK126),
      X(i+1),6 = Xi,6,     X(i+1),7 = Xi,7 [^] (F0(Xi,6)[+]SK127).

(4) C0 = X32,0 [+] WK4,        C1 = X32,1,
    C2 = X32,2 [^] WK5,        C3 = X32,3,
    C4 = X32,4 [+] WK6,        C5 = X32,5,
    C6 = X32,6 [^] WK7,        C7 = X32,7.

(5) C = C7 || C6 || C5 || C4 || C3 || C2 || C1 || C0
```

Decryption:

The Decryption is the is reverse process of Encryption.
Transformation of a 64-bit block CipherText into 64-bit block PlainText.

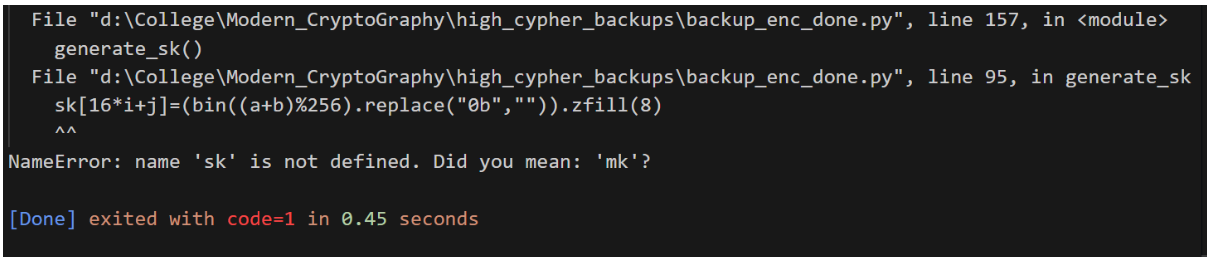
```
(1) C = C7 || C6 || C5 || C4 || C3 || C2 || C1 || C0

(2) C0 = X32,0 [-] WK4,        C1 = X32,1,
      C2 = X32,2 [^] WK5,        C3 = X32,3,
      C4 = X32,4 [-] WK6,        C5 = X32,5,
      C6 = X32,6 [^] WK7,        C7 = X32,7.
(3) for i = 31:
        X(i+1),0 = Xi,0,     X(i+1),1 = Xi,1 [-] (F1(Xi,0)[^]SK124),
        X(i+1),2 = Xi,2,     X(i+1),3 = Xi,3 [^] (F0(Xi,2)[+]SK125),
        X(i+1),4 = Xi,4,     X(i+1),5 = Xi,5 [-] (F1(Xi,4)[^]SK126),
        X(i+1),6 = Xi,6,     X(i+1),7 = Xi,7 [^] (F0(Xi,6)[+]SK127).
    for i = 30 to 0:
        X(i+1),1 = Xi,2 [-] (F0(Xi,0) [^] SK(4*i+2)),   X(i+1),0 = Xi,1,
        X(i+1),3 = Xi,4 [^] (F1(Xi,2) [+] SK(4*i+1)),   X(i+1),2 = Xi,3,
        X(i+1),5 = Xi,6 [-] (F0(Xi,4) [^] SK(4*i)),      X(i+1),4 = Xi,5,
        X(i+1),7 = Xi,0 [^] (F1(Xi,6) [+] SK(4*i+3)),   X(i+1),6 = Xi,7.
```

(4) P0 = X0,0 [+] WK0, P1 = X0,1,
P2 = X0,2 [^] WK1, P3 = X0,3,
P4 = X0,4 [+] WK2, P5 = X0,5,
P6 = X0,6 [^] WK3, P7 = X0,7.

(5) P = P7 || P6 || P5 || P4 || P3 || P2 || P1 || P0

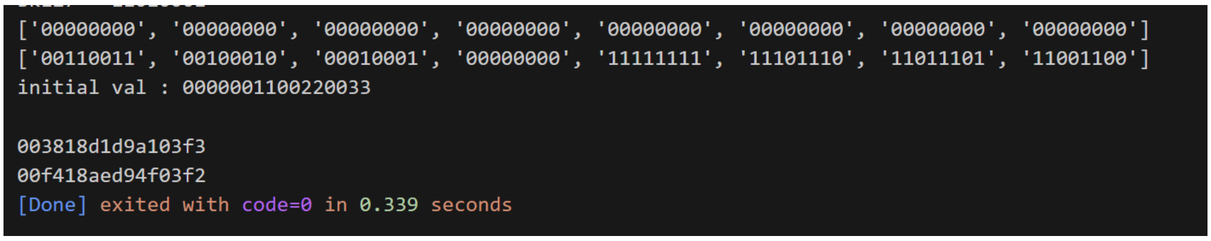
Error-Handling:

1) Since ‘sk’ was not defined it showed an error, that is subkey in short



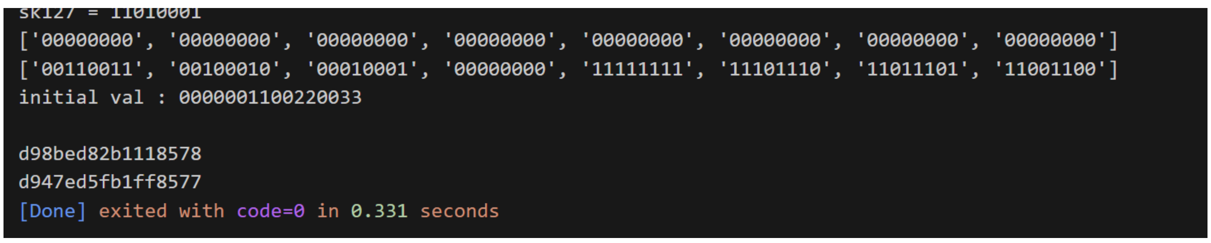
```
  File "d:\College\Modern_CryptoGraphy\high_cypher_backups\backup_enc_done.py", line 157, in <module>
    generate_sk()
  File "d:\College\Modern_CryptoGraphy\high_cypher_backups\backup_enc_done.py", line 95, in generate_sk
    sk[16*i+j]=(bin((a+b)%256).replace("0b","")).zfill(8)
    ^^
NameError: name 'sk' is not defined. Did you mean: 'mk'?

[Done] exited with code=1 in 0.45 seconds
```

Once subkey is defined it worked fine.



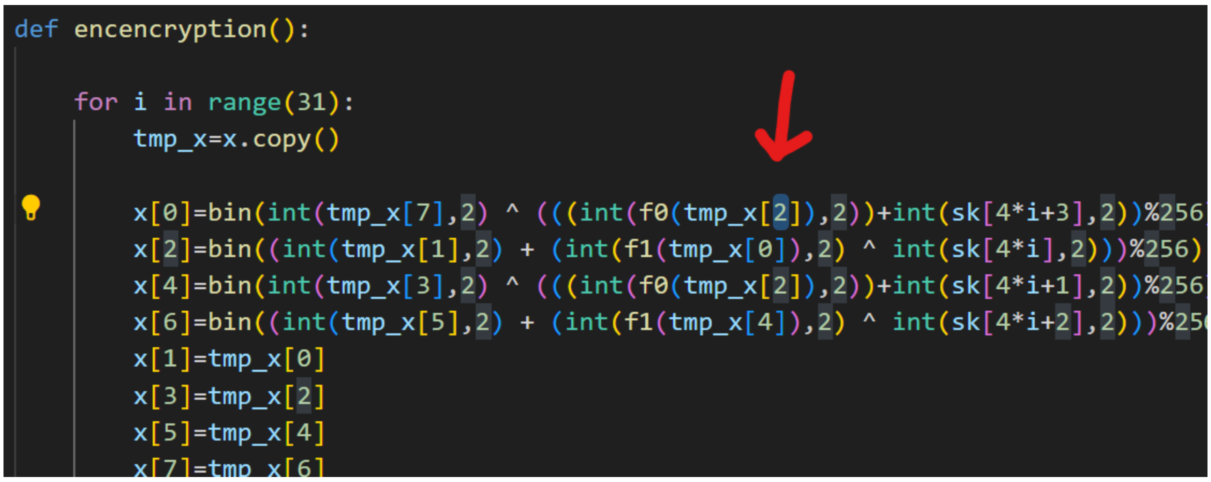
```
['00000000', '00000000', '00000000', '00000000', '00000000', '00000000', '00000000', '00000000']
['00110011', '00100010', '00010001', '00000000', '11111111', '11101110', '11011101', '11001100']
initial val : 0000001100220033

003818d1d9a103f3
00f418aed94f03f2
[Done] exited with code=0 in 0.339 seconds
```

2) The output for the input is wrong it should be ‘00f418aed94f03f2’
But it is ‘d947ed5fb1ff8577’



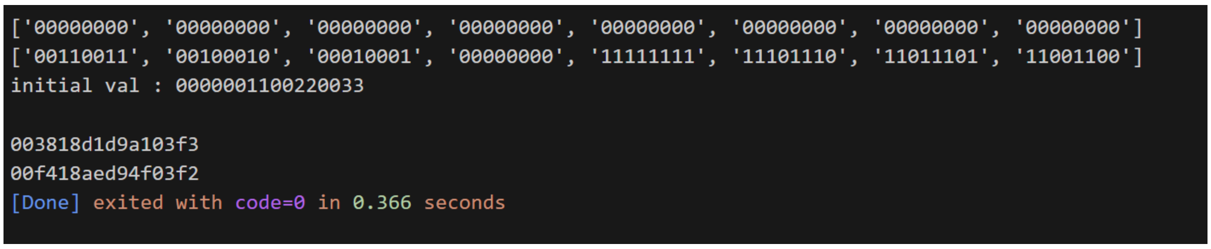
```
sk127 = 11010001
['00000000', '00000000', '00000000', '00000000', '00000000', '00000000', '00000000', '00000000']
['00110011', '00100010', '00010001', '00000000', '11111111', '11101110', '11011101', '11001100']
initial val : 0000001100220033

d98bed82b1118578
d947ed5fb1ff8577
[Done] exited with code=0 in 0.331 seconds
```

There was an integer change in first encryption round it should be 6 instead it was 2



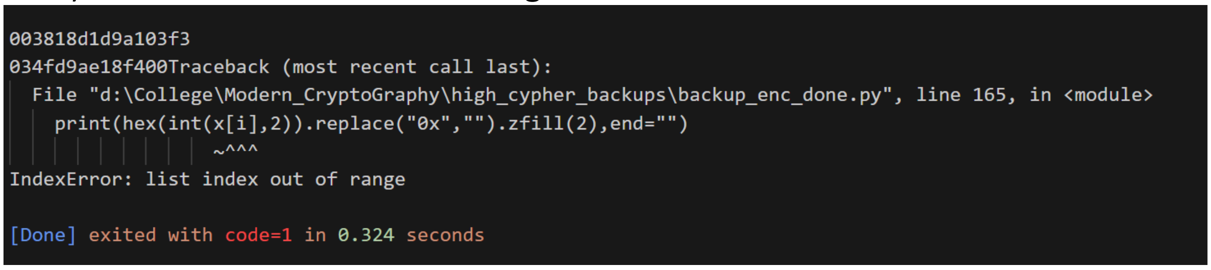
```
def encencryption():

    for i in range(31):
        tmp_x=x.copy()

        x[0]=bin(int(tmp_x[7],2) ^ (((int(f0(tmp_x[2]),2))+int(sk[4*i+3],2))%256
        x[2]=bin((int(tmp_x[1],2) + (int(f1(tmp_x[0]),2) ^ int(sk[4*i],2)))%256)
        x[4]=bin(int(tmp_x[3],2) ^ (((int(f0(tmp_x[2]),2))+int(sk[4*i+1],2))%256
        x[6]=bin((int(tmp_x[5],2) + (int(f1(tmp_x[4]),2) ^ int(sk[4*i+2],2)))%25
        x[1]=tmp_x[0]
        x[3]=tmp_x[2]
        x[5]=tmp_x[4]
        x[7]=tmp_x[6]
```

Once that was fixed, the correct output came.



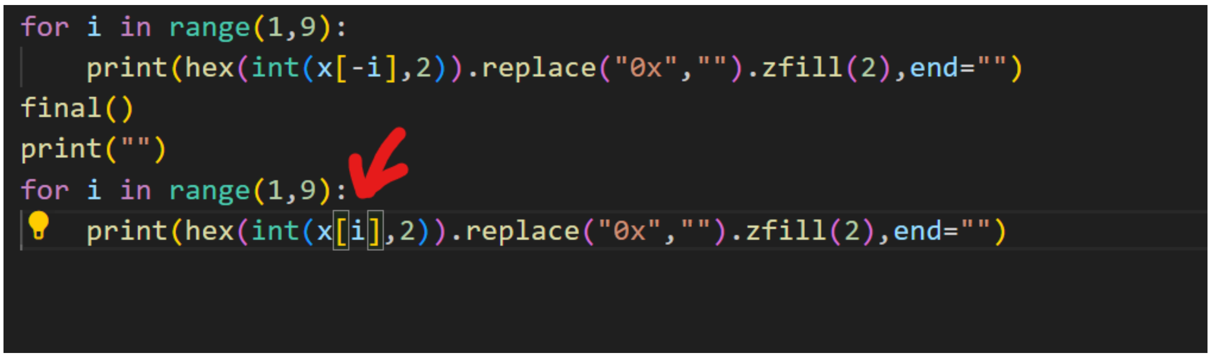
```
['00000000', '00000000', '00000000', '00000000', '00000000', '00000000', '00000000', '00000000']
['00110011', '00100010', '00010001', '00000000', '11111111', '11101110', '11011101', '11001100']
initial val : 0000001100220033

003818d1d9a103f3
00f418aed94f03f2
[Done] exited with code=0 in 0.366 seconds
```

3) The index value was out range.



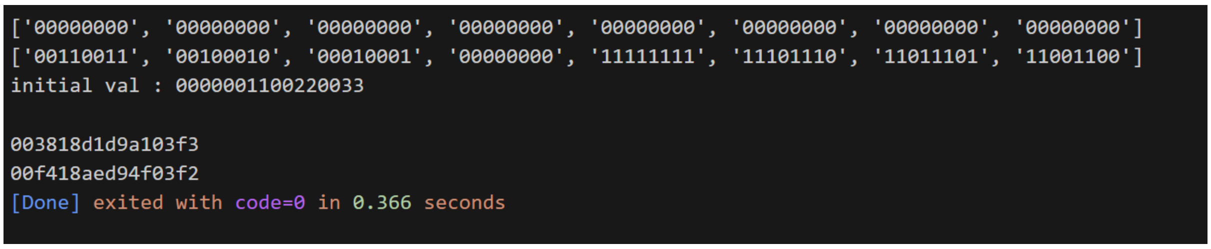
```
003818d1d9a103f3
034fd9ae18f400Traceback (most recent call last):
  File "d:\College\Modern_CryptoGraphy\high_cypher_backups\backup_enc_done.py", line 165, in <module>
    print(hex(int(x[i],2)).replace("0x","").zfill(2),end="")
                  ~^^^
IndexError: list index out of range

[Done] exited with code=1 in 0.324 seconds
```

it should have been ‘-i’ instead it was ‘i’.



```
for i in range(1,9):
    print(hex(int(x[-i],2)).replace("0x","").zfill(2),end="")
final()
print("")
for i in range(1,9):
    print(hex(int(x[i],2)).replace("0x","").zfill(2),end="")
```

Once that was fixed, correct output came.



```
['00000000', '00000000', '00000000', '00000000', '00000000', '00000000', '00000000', '00000000']
['00110011', '00100010', '00010001', '00000000', '11111111', '11101110', '11011101', '11001100']
initial val : 0000001100220033

003818d1d9a103f3
00f418aed94f03f2
[Done] exited with code=0 in 0.366 seconds
```

Conclusion:

In conclusion, our cryptography project successfully constructed the HIGHT cipher, a robust block cipher suitable for low-resource devices. This accomplishment is further highlighted by the successful deployment of HIGHT in two unique modes, CBC (Cipher Block Chaining) and ECB (Electronic Codebook), as well as a mode that accepts plain text input rather than hexadecimal data. Our research involved a thorough examination of the work titled 'HIGHT: A New Block Cypher Suitable for Low-Resource Devices,' which allowed us to understand the complexities of key generation, encryption, and decryption.

This project combines theoretical understanding with actual coding skills, giving existence to the HIGHT cipher. Aside from strengthening our understanding of cryptographic principles, this implementation demonstrated our ability to transfer theoretical information into a practical, functional codebase. The project's scope went beyond the algorithm itself, including the creation of a user-friendly interface.

As a unit, this project offered us with a unique opportunity to navigate the complexity of cryptography, starting with a deep dive into a research article and finishing in a fully usable cryptographic scheme.