**FACULTY OF INFORMATICS**

**COURSEWORK COVERSHEET**

|  |  |  |  |
| --- | --- | --- | --- |
| **SUBJECT’S INFORMATION:** | | | |
| Subject: | CSCI361 Cryptography and Secure Applications | | |
| Session: | July 2019 | | |
| Programme / Section: | BCS (CS1) | | |
| Lecturer: | **Mohamad Faizal Alias** | | |
| Coursework Type  *(tick appropriate box)* | ❑ Individual Assessment | | |
| Coursework Title: | Assessment 2 | Coursework Percentage: | 10% |
| Hand-out Date: | Week 6 | Received By :  (signature) |  |
| Due Date: | Week 9 | Received Date : |  |
| **STUDENT’S INFORMATION:** | | | |
| Student’s Name & ID: | **TEH WIN SAM** | **J16021533** | **6306196** |
| Contact Number / Email: | **0133696298** | [**me@tehwinsam.com**](mailto:me@tehwinsam.com) |  |
| **STUDENT’S DECLARATION** | | | |
| By signing this, I / We declare that:   1. This assignment meets all the requirements for the subject as detailed in the relevant Subject Outline, which I/ we have read. 2. It is my / our own work and I / we did not collaborate with or copy from others. 3. I / we have read and understand my responsibilities under the University of Wollongong’s policy on plagiarism. 4. I / we have not plagiarised from published work (including the internet). Where I have used the work from others, I / we have referenced it in the text and provided a reference list at the end of the assignment.   I am / we are aware that late submission without an authorised extension from the subject co-ordinator may incur a penalty. *(See your subject outline for further information).* | | | |
| Name & Signature: | TEH WIN SAM |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Subject: | CSCI361 Cryptography and Secure Application | | Session: | July 2019 |
| Programme / Section: | BCS (CS1) | | Lecturer: | **Mohamad Faizal Alias** |
| Coursework Type:  (Tick appropriate box) | ❑ Individual Assessment | | | |
| Coursework Title: | Assessment 2 | Coursework Percentage: | | 10% |
| Hand-out Date: | Week 6 | Received By:  (Signature) | |  |
| Due date: | Week 9 | Received Date: | |  |
| **STUDENT’S INFORMATION:** | | | | |
| Student’s Name & ID: | TEH WIN SAM J16021533 |  | |  |
| Contact Number / Email: | 0133696298 |  | |  |

**COURSEWORK SUBMISSION RECEIPT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Assessment Criteria** | | **Total Marks** | **Given Marks** |
|  | Part 1 - Modes of Block cipher (Group of 2 persons) | 3 |  |
|  |  |  |  |
|  | Part 2: Complete program with SEED encryption | 3 |  |
|  | Implementation of Cipher Feedback Mode (CFB) in program | 2 |  |
|  | Overall report | 2 |  |
|  |  |  |  |
|  |  |  |  |
|  | | **10** |  |
|  | | **Penalty** |  |
| Marked by: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Date: \_\_\_\_\_\_\_\_\_\_\_\_\_ | | **Final Mark (10 %)** |  |
| **Lecturer’s Comments** | | | |
|  | | | |
| **Penalty for late submission:** | | | |
| 1 day – minus 20% of total mark awarded  2 days – minus 50% of total mark awarded  3 days – 0 mark for this piece of coursework | | | |

**University of Wollongong**

**CSCI361: Cryptography and Secure Applications**

**July 2019 Session**

**Individual Assessment 2**

**Aims**

This assessment aims to establish full understanding of Block Ciphers and Modes of block ciphers.

**Objectives**

The assessment includes the following objectives:

1. Explain and apply fundamental cryptographic principles of block ciphers in particular SEED encryption
2. Classify and distinguish cryptographic algorithms in terms of their cryptographic characteristics and services provided.
3. Understand cipher modes, in particular CFB mode
4. Able to apply theoretical knowledge into a working C++ program in promoting Block cipher and Mode of block cipher

**Assessment 2 Part 1: (3%) - Group**

You will be group into 2 persons in one group. Your group will be assigned any ONE of the following Mode of block cipher: (Will be announce during F2F class)

* + - 1. Cipher Block Chaining (CBC)
      2. Cipher Feedback Mode (CFB)
      3. Output Feedback Mode (OFB)
      4. Counter Mode

Within your group discuss the operation of the mode assigned to your group. Create a video/screencast with proper annotation and voice over to explain the working of this mode.

Make sure your video covers sample block of Data (can be simplified version) with encryption and decryption process. You may choose any Block ciphers exists, but preferable DES or AES Video/Animation/Screencast must have appropriate diagram/figure and complete with necessary labels for clear understanding. Diagram used should not be a copied of the textbook figure or from class slides.

The dynamic working of encryption and decryption of the Mode should be properly animated and explained step-by-step.

**Submission for Part 1:**

Once completed, upload your video to YouTube. Submit the hyperlink to you video over Video submission link on UOW Moodle. Make sure the video does **NOT** exceed **10 minutes**.

Share the video link over our CSCI361 communication channel for others to view.

**Assessment 2 Part 2 Specifications: (7%) - Individual**

**Specifications:**

**Background of SEED Encryption:** SEED is a block cipher developed by the Korea Internet & Security Agency (KISA). It is used broadly throughout South Korean industry, but seldom found elsewhere. (Wikipedia.org).

SEED is a 16-round Feistel network with 128-bit blocks and a 128-bit key. It uses two 8 × 8 S-boxes. The 128-bit data block enters a Feistel network with an F-function operating on 64-bit halves, while the F-function itself is another Feistel network composed of a G-function operating on 32-bit halves.

However the recursion does not extend further because the G-function is not a Feistel network. In the G-function, the 32-bit word is considered as four 8-bit bytes, each of which is passed through one or the other of the S-boxes, then combined in a moderately complex set of boolean functions such that each output bit depends on 3 of the 4 input bytes.

**Task:** Create your own SEED encryption-decryption program implemented with **Cipher Feedback mode (CFB)**. Follow the exact specification of SEED on Block Size, Key Size and Rounds selected above. SEED encryption is also part of Crypto++ library.

The program allows the user to choose encryption process or decryption process at a time.

The usage of your SEED with CFB mode should be:

seed\_cfb -e key.txt plaintext.txt ciphertext.txt

seed\_cfb -d key.txt ciphertext.txt plaintext.txt

where the -e option is for encryption, the -d option is for decryption. and key.txt is the file containing a key.

The key.txt file stores the 128 bits key for the SEED key (i.e. password of 16 bytes long) and IV (Initialization Vector) should be placed on the second line also with 16 bytes size. Both key and IV should be human readable.

If the user does not follow the correct format of input, a clear instruction should be displayed to instruct the user how to use the program. As for the file name of ciphertext.txt and plaintext.txt could be enter as the user wish.

Provide a complete C++ codes/project files; this include source file, make file, \*.exe file and etc. for compilation. You must provide the compilation instruction in your report.

To assists your development you may use available Crypto++ library.

All codes should be properly commented. Any re-used parts should also be commented of where the original comes from.

Report all your work as follows:

* What have you done, references, study on SEED and suitable diagram to explain SEED (provide proper referencing)
* Combined your SEED with CFB and produce an appropriate conceptual diagram for both encryption and decryption where this diagram can explain the working of your program. (i.e. looks like Mode of Block cipher diagram)
* Explain the processes done by your codes (methods available) and library that you use
* Explain Key size, IV (Initialization Vector) format used, data encoding etc.
* Explain how to compile your code, include the library in used etc. and overall code development
* An example of executing your program with screen captures, sample input and output files

**Submission requirement for Part 2:**

You need to submit the following files:

* All source codes including \*.exe file.
* key.txt, plaintext.txt and ciphertext.txt samples
* report document

All files should be zipped to form a zip file named CSCI361-A2-<yourName>.zip. Remember to put your name and student number in all source codes, report cover and etc.

Assignments must be submitted electronically via Moodle submission link available.

**Plagiarism**

A plagiarised assignment will receive a zero mark (and penalised according to the university rules). Plagiarism detection software will be used.

Table of Contents

[**Quick overview of Cipher FeedBack using SEED block cipher encryption** 9](#_Toc20586292)

[Cipher Feedback Mode Encryption 9](#_Toc20586293)

[Cipher Feedback Mode Decryption 10](#_Toc20586294)

[REVISION ON STREAM CIPHER 10](#_Toc20586295)

[Citation 11](#_Toc20586296)

[**INTERNAL STURCTURE OF SEED ALGORITHM** 12](#_Toc20586297)

[F function 13](#_Toc20586298)

[G Function 14](#_Toc20586299)

[Key Scheduling 15](#_Toc20586300)

[Citation 16](#_Toc20586301)

[**INCLUDE LIBRARY EXPLANATION** 17](#_Toc20586302)

[Code required to display color in terminal 17](#_Toc20586303)

[Include Standard library 17](#_Toc20586304)

[**ARGUMENT CHECKING** 17](#_Toc20586305)

[**READ FILE** 19](#_Toc20586306)

[KEY AND IV FROM FILE (3ND ARGUMENT) 19](#_Toc20586307)

[READ INPUT FILE (4TH ARGUMENT) 20](#_Toc20586308)

[READ 1ST ARGUMENT (MODE = ENCRYPT || DECRYPT) 20](#_Toc20586309)

[**MODE OF ALGORITHM** 22](#_Toc20586310)

[ENCRYPTION 22](#_Toc20586311)

[STORE INTO 5th ARGUMENT OUTPUT FILE 24](#_Toc20586312)

[DECRYPTION 25](#_Toc20586313)

[STORE INTO 5th ARGUMENT OUTPUT FILE 27](#_Toc20586314)

[**USER MANUAL (GITHUB)** 28](#_Toc20586315)

[README.MD 28](#_Toc20586316)

[GIT CLONE FROM GITHUB 28](#_Toc20586317)

[INSTALLATION 29](#_Toc20586318)

[COMMAND TO COMPILE/BUILD 29](#_Toc20586319)

[How to issue help command from the program? 29](#_Toc20586320)

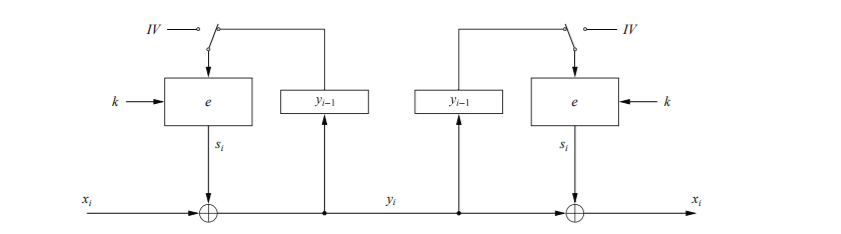
[**USAGE DEMONSTRATION** 30](#_Toc20586321)

[DEMO ENCRYPT 30](#_Toc20586322)

[DEMO DECRYPT 31](#_Toc20586323)

# **Quick overview of Cipher FeedBack using SEED block cipher encryption**

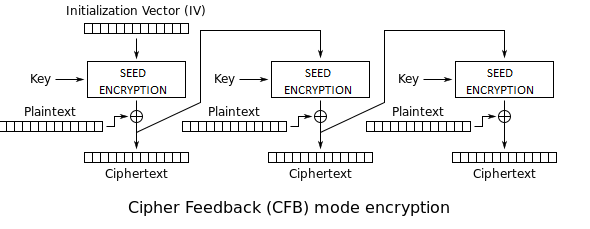
**ENCRYPTION MODE | DECRYPTION MODE**



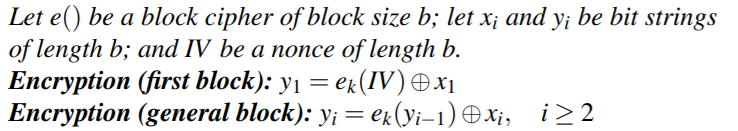
It might be curious that why the decryption process/mode in CFB is ‘e’ and not ‘e-1’. Let discuss in CFB decryption mode.

CFB is part of a Block Cipher Mode, Cipher Feedback (CFB) mode uses a block cipher as a building block for a stream cipher.

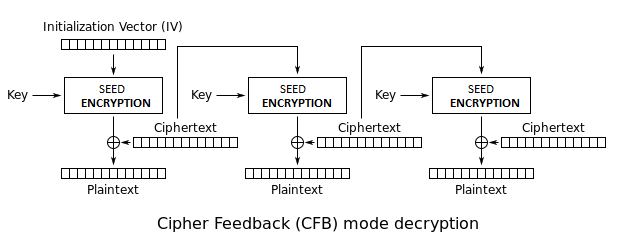
Cipher Feedback Mode Encryption



First, it generate an random IV(nonce) using algorithm “CSPRNG” or “PRNG” to generate and it uses KEY to encrypt the “IV” for first round, and after finish the encryption using SEED algorithm; the output is with our plaintext(X1) which generate the Ciphertext(Y1), and it take the previous Y1 as feedback to encrypt X2 to generate Y2



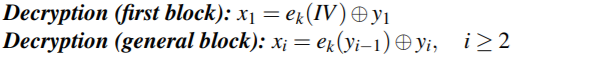
## Cipher Feedback Mode Decryption



For CFB Decryption mode, it’s still using SEED ENCRYPTION reason is Cipher Feedback (CFB) mode uses a block cipher as a building block for a stream cipher, and the keyword is **“stream cipher”**. In order to generate the same output we have to use back the **same encryption mode?**

Decryption part in CFB it uses “ENCRYPTION MODE” to perform “DECRYPTION”

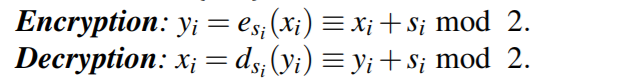




We got all the ciphertext blocks but we don’t have the plaintext block , so when it came to decryption we can use the same equation to perform the decryption task.

### REVISION ON STREAM CIPHER

Let quickly do a very quick revision for encryption and decryption using Stream Cipher



Why uses the same equation in Steam Cipher to perform encrypt and decrypt

Quick math

(Encryption)Yi = Xi + Si mod 2

(Decryption)Xi =Yi + Si mod 2

**Decryption PART**

Xi =Yi + Si mod 2 , Given “ Yi= Xi + Si mod 2 ”

Therefore , Xi = (Xi + Si ) + Si mod 2

Xi = Xi + 2Si mod 2

When Si + Si = 2Si , and it mod2 which mean = 0

Meaning , Xi = Xi;

Easy math Q.E.D(end of prove)

THEREFORE , Decryption part in CFB it uses “ENCRYPTION MODE” to perform “DECRYPTION”

## Citation

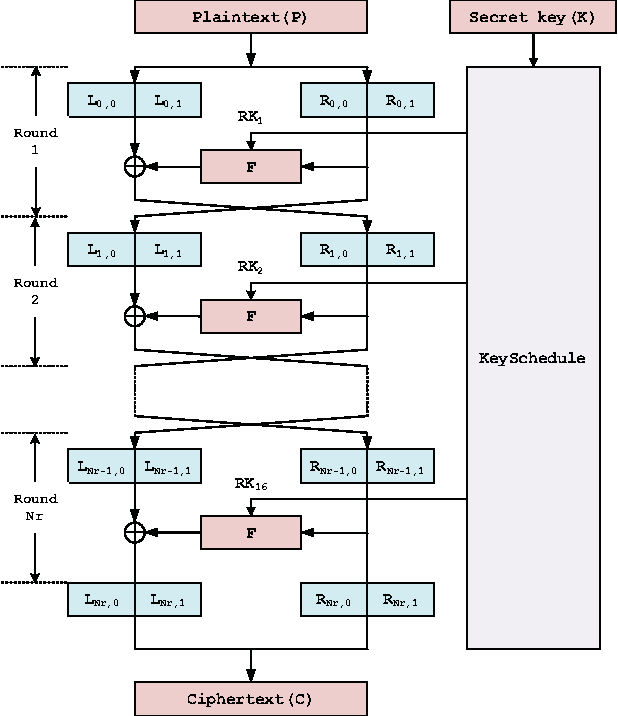
Paar, C., Preneel, B. and Pelzl, J. (2014). *Understanding Cryptography*. New York: Springer, pp.30 , 132.

# **INTERNAL STURCTURE OF SEED ALGORITHM**

A quick overall of SEED Encryption, it’s a BLOCK CIPHER similar to DES because it’s a “Feistel network” with 16 rounds meaning that it separate the plaintext(128bits) into “left” and “right” each contain 64bits then processes with 16 rounds of process to encrypt plaintext to ciphertext same goes to decryption 16 rounds from ciphertext to plaintext.

SEED is a better version than DES because DES have input/key(Initial permutation)/output of 64bits of key whereby SEED support 128bits of input-output and key length.

For SEED it have 2 8x8 S-Boxes

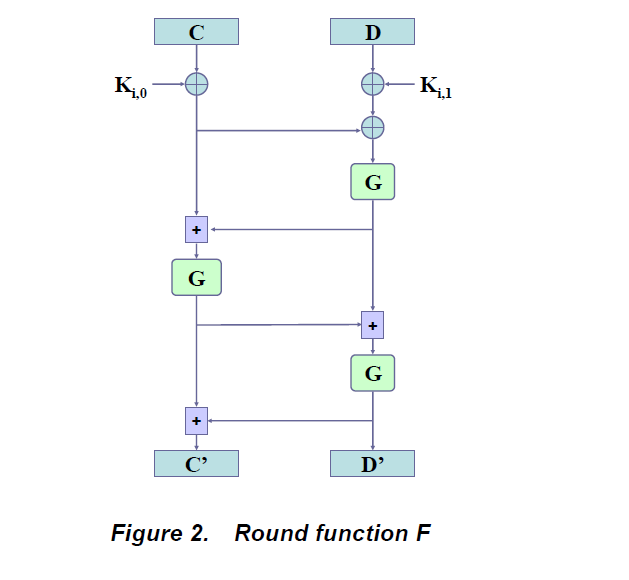


Do notice that the left and right separate into half and only RIGHT will go into F function to generate the subkey “RKn” from the Key Scheduling(which 128bits of input key separate into 4 blocks and each block is 32bits, which will discuss later) and R0,0 | R0,1 which later will denote/name it as C & D.

## F function

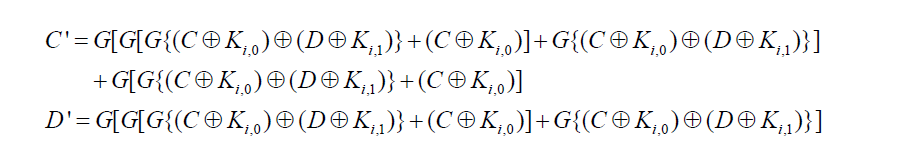
The F function do contain G function and XOR to generate the C’ and D’. C and D which is Ri,0 | Ri,1

Ki,0 is 32bits and Ki,1 is 32bits which came from Key Scheduling separate into half from the SubKeyi



So the following equation to generate C prime and D prime is

Variable i represent as rounds and assume the first round starting from ‘0’ .



And the result of C’ and D’ will (XOR) with Li+1,0 and Li+1,1

## G Function

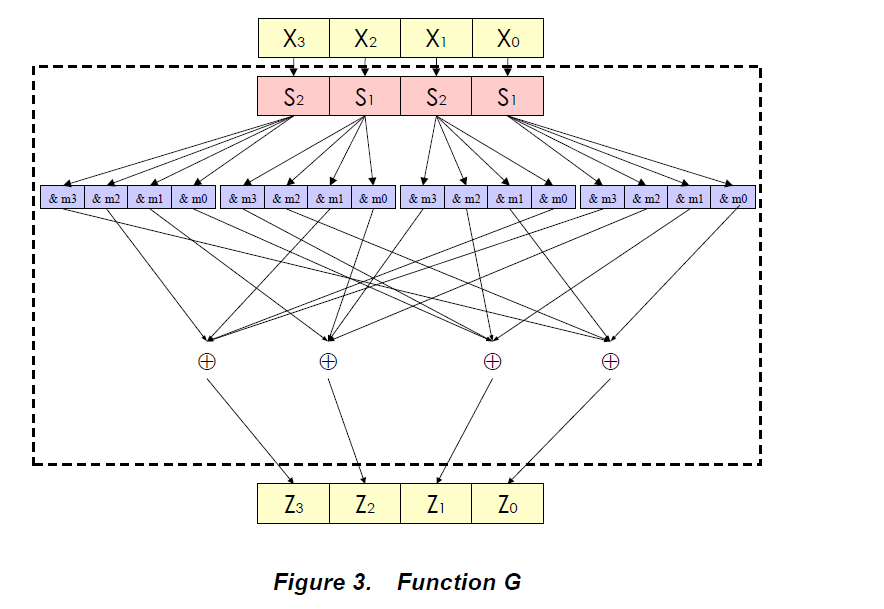
Now we going to discuss the internal structure of ‘G’ Function.

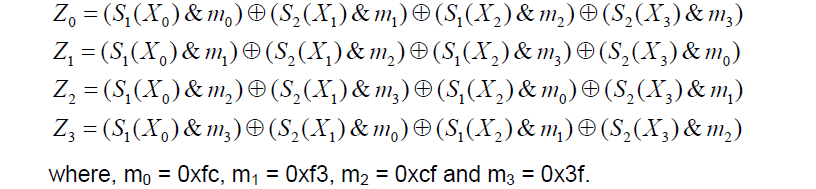
The G function have ‘2’ layers.

The two layer consist of TWO 8 x 8 S Boxes and Layer of BLOCK PERMUTATION of ‘16’ 8 BIT – SUB BLOCK

1st layer is TWO 8 x 8 S Boxes which generated from the Boolean function x247 and x251.

2nd layer is set of permutation in each S-Box





S1 represent as S1-BOX and S2 represent as S2-BOX

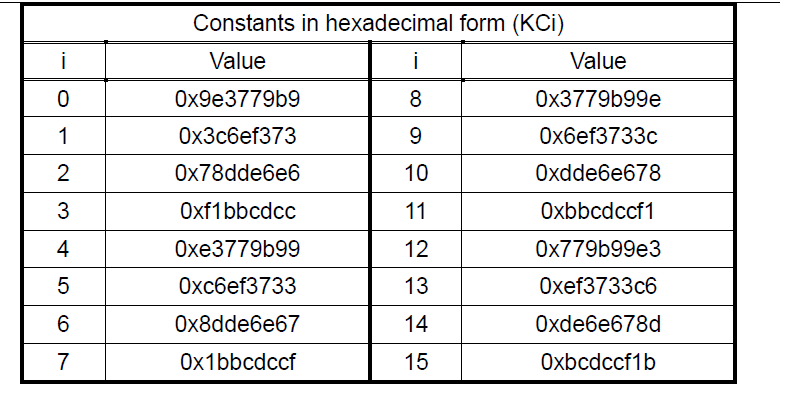
## Key Scheduling

How it generate subkeys Ki,0 and Ki,1 , which ‘i’ represent round.

As we mentioned the key length input is 128bits and it separate into 4 blocks which called as A,B,C,D and each block is 32bits.

The subkey is generate in Key Schedule, it’s very obvious. It uses G function, and some additional + subtraction and some left/right rotation in order to generate the Ki,0 and Ki,1.

Before we proceed to generating key algorithm there’s a **constant value** that I have to show before proceed to the algorithm that generate the subkeys.



Assume we generating the first subkey 

We notice that to generate

K1,0 use Gfunction((A)1st + (C)3rd block – KC0)

K1,1 use Gfunction((B)2nd – (D)4th block + KC0)

Now the second subkey



K2,0 use Gfunction((A)1st + (C)3rd block – KC1)

K2,1 use Gfunction((B)2nd + KC1 -(D)4th block)

Third subkey is



K3,0 use Gfunction((A)1st + (C)3rd block – KC2)

K3,1 use Gfunction((B)2nd - (D)4th block + KC2)

\*Do notice that\*

Will be 4th subkey which i will be replace to ‘4’

Ki,1 G((B)2nd **+ (4th)BLOCK – KCi-1**)

Notice that **red color and green color** both of them will swap every each round, till round 16.

As below code



## Citation

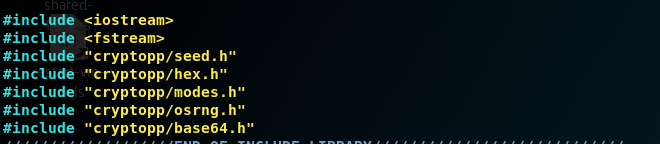
Seed.kisa.or.kr. (2019). *KISA 암호이용활성화 - SEED*. [online] Available at: https://seed.kisa.or.kr/kisa/algorithm/EgovSeedInfo.do [Accessed 28 Sep. 2019].

Seed.kisa.or.kr. (2019). [online] Available at: https://seed.kisa.or.kr/async/MultiFile/download.do?FS\_KEYNO=FS\_0000000009 [Accessed 28 Sep. 2019].

# **INCLUDE LIBRARY EXPLANATION**

Code required to display color in terminal

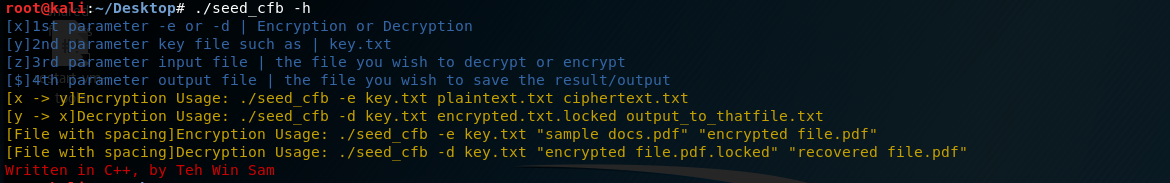
## Include Standard library



# **ARGUMENT CHECKING**

Firstly, if first parameter is not null then it will check is the mode is “-h”. **If (yes)**



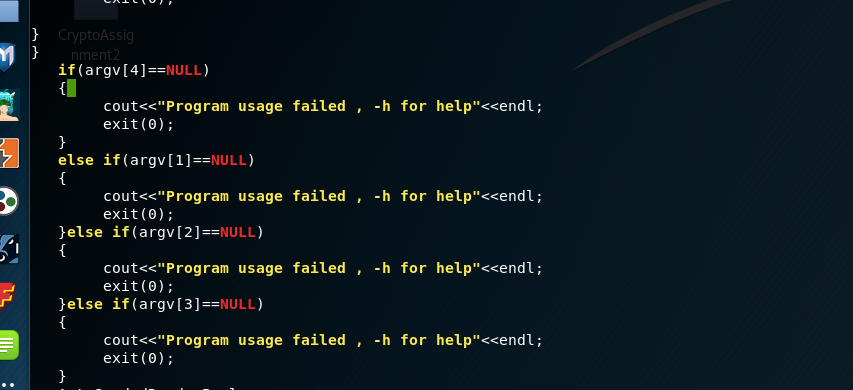


Second for color code cout<<FBLU(“text”)<<endl;

FYEL= yellow , FBLU=blue , FRED=red



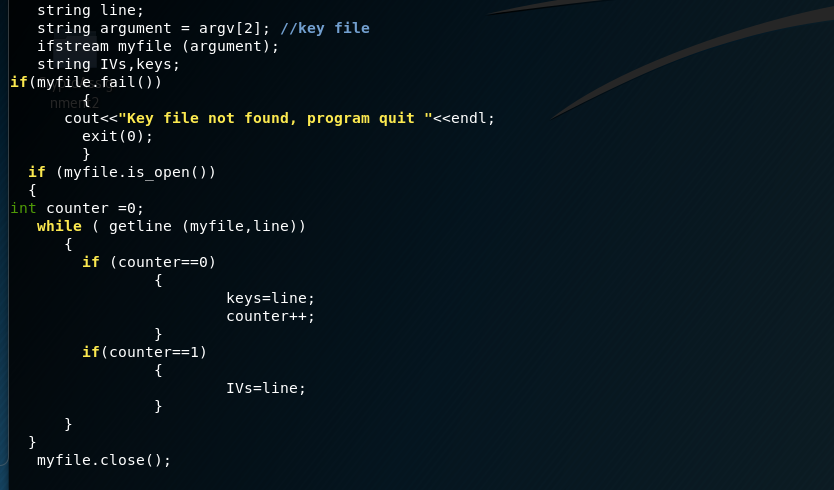
Next to check the parameter making sure they input all 4 parameters.



If argument failed to input properly it will output “program usage failed , -h for help”

# **READ FILE**

## KEY AND IV FROM FILE (3ND ARGUMENT)



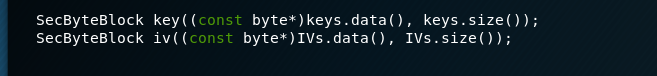
Since the “Key file” contain “KEY” and “IV”.

The first line is “IAMTEHWINSAMHAHA” which is the value of KEY

Second line is “TEHWINSAMISGOODS” which is the value of IV

Therefore it’s a counter if counter==0 mean it’s first line mean it’s representing the key value, and if counter != 0 which mean it’s value of IV.

The value of KEY and IV definitely in 16 bytes; no more or less.



The following code is to manually assign the value of key and iv into **block variable** “Key” and “IV” value.



## READ INPUT FILE (4TH ARGUMENT)

We now look for third parameter which is the input file which user wish to “encrypt/decrypt”.

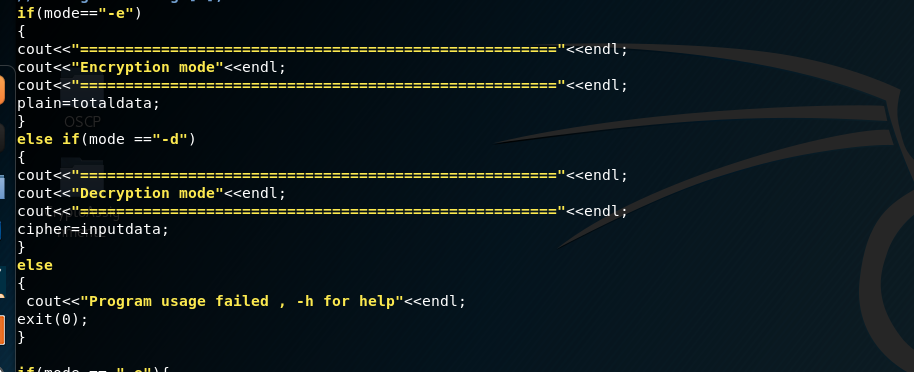
If the file failed to open “File not found” then it will quit the program with display sentence “your input file not found , program quit ” and exit(0); function to quit the program.

## READ 1ST ARGUMENT (MODE = ENCRYPT || DECRYPT)

Next notice that the argument[1] which is the mode **-e** or **-d** which is use to tell the program to check it’s encryption or decryption mode .

Assume it’s “**-d**” parameter; mode of decryption then it will check the last 7 word in the input file making sure that the file with “.locked” extension, if it’s not with the “.locked” file extension which mean the usage of program is wrong because using this program to encrypt it will add an extra extension file name “.locked” to the encrypted file after the encryption.

After the 3rd parameter able to open successfully it will start to grab sentence by sentence until EOF(End of File) every first line of sentence will add “\n” which represent the “new line/next line”



If mode is “-e”, if the mode is then the output is display “Encryption mode”

Else if mode is “-d”, the mode is then output the “Decryption mode”

If not “-e” or “-h” then it will display “Program usage failed , -h for help”

# **MODE OF ALGORITHM**

## ENCRYPTION

First, for “-e” encryption

try{

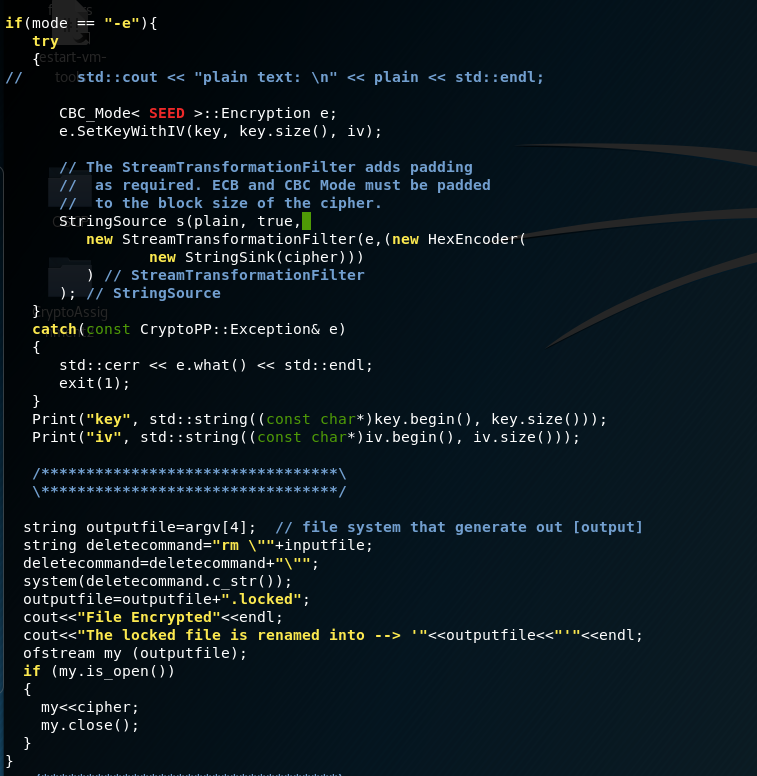
}

catch

{

}

Above try {}catch{} is exception handling, and which given in the crypto++ website, but I did some modification in order to accept “larger number of plaintext”



CFB\_Mode< SEED >::Encryption e;

e.SetKeyWithIV(key, key.size(), iv);

We create an OBJECT first and we make use the key and iv that we created from string into “BYTES”

And set the encryption key with the IV into object ‘e’

BEFORE

StringSource s(plain, true,

new StreamTransformationFilter(e,

new StringSink(cipher)

) // StreamTransformationFilter

); // StringSource

AFTER

StringSource s(plain, true,

new StreamTransformationFilter(e,(new HexEncoder(

new StringSink(cipher)))

) // StreamTransformationFilter

); // StringSource

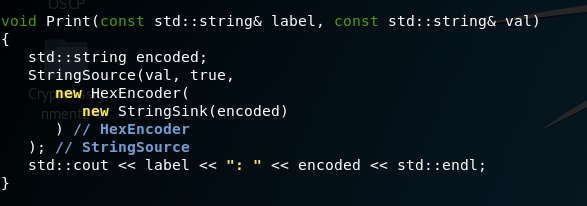
Which is used to encode(new HexEncoder) the plaintexts( variable plain ) into ‘hexadecimal’ and save into “variable cipher” which is the ciphertext value in HEX format using ‘e’ object which is the “key and IV” in our scenario.

StringSource s(plain, true,

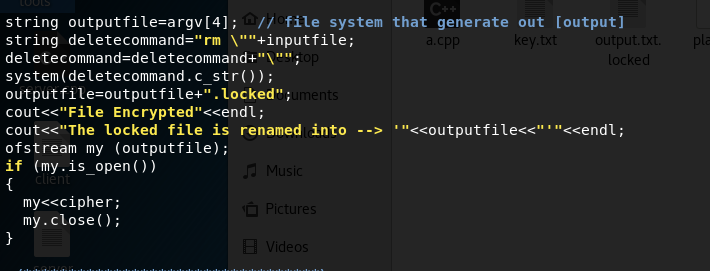
new StreamTransformationFilter(e,(new HexEncoder(

new StringSink(cipher)))

After that it’s it will print out the value of KEY and IV in format hexadecimal, with another function.



### STORE INTO 5th ARGUMENT OUTPUT FILE

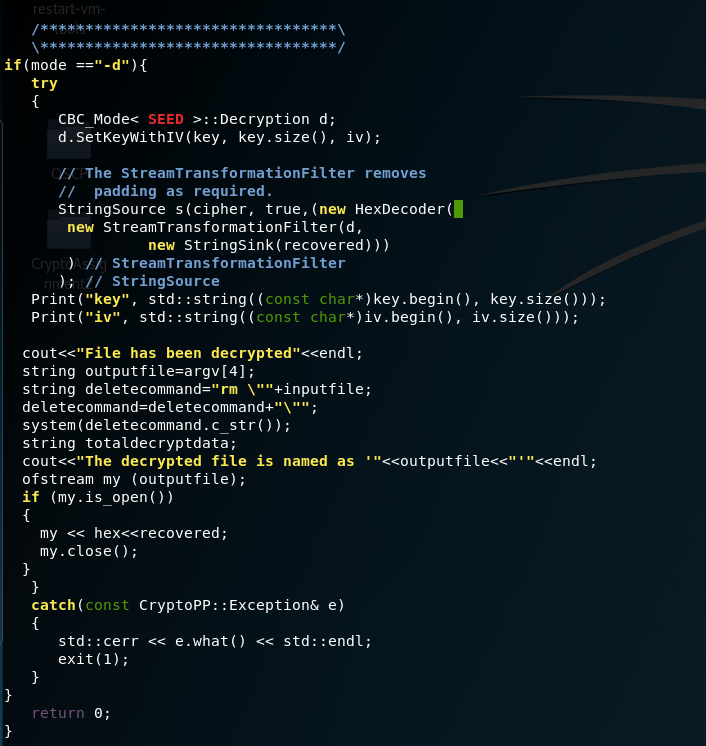


String output=argv[4] // which is the output file name

Then the program itself will add an extra file extension “.locked”. Then it will remove the original plaintext with command “rm name\_of\_file.txt”. After it have finished the process of encryption and deletion on the plaintext file it will start write into a new file; assume the 4th parameter is “output.txt” then will generate as “output.txt.locked”.

## DECRYPTION

For parameter “-d” decryption mode



try{

}

catch

{

}

Above try {}catch{} is exception handling, and which given in the crypto++ website, same I did some modification in order to accept “larger number of cipher”/ since we have made modification on encryption algorithm therefore, we have to made modification on decryption algorithm.

CFB\_Mode< SEED >::Decryption d;

d.SetKeyWithIV(key, key.size(), iv);

Now, the decryption part we create an object for decryption. We name the object ‘d’ and we set the decryption key using ( key and iv in BYTES format) into the object ‘d’.

BEFORE

// The StreamTransformationFilter removes

// padding as required.

StringSource s(cipher, true,

new StreamTransformationFilter(d,

new StringSink(recovered)

) // StreamTransformationFilter

); // StringSource

AFTER

// The StreamTransformationFilter removes

// padding as required.

StringSource s(cipher, true,(new HexDecoder(

new StreamTransformationFilter(d,

new StringSink(recovered)))

) // StreamTransformationFilter

); // StringSource

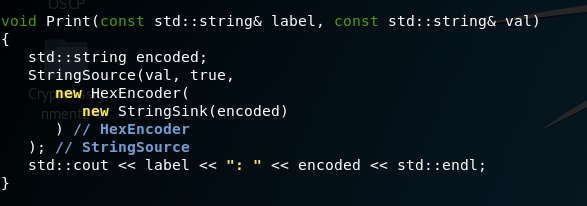
Which is take the ciphertext(variable cipher) decode from “hex” to “raw” and with StreamTransformationFilter import the object ‘d’ to decrypt and store the plaintext into (recovered).

StringSource s(cipher, true,(new HexDecoder(

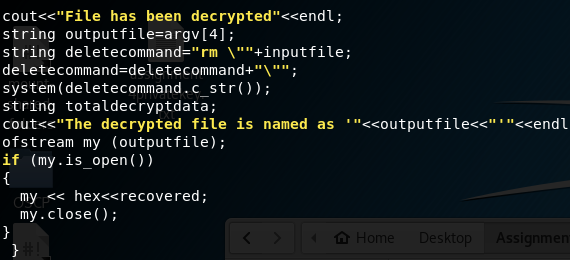
new StreamTransformationFilter(d,

new StringSink(recovered)))

After that it’s it will print out the value of KEY and IV in format hexadecimal, from another function



### STORE INTO 5th ARGUMENT OUTPUT FILE



String output=argv[4] // which is the output file name

Then the program itself will remove the extra file extension “.locked”. Then it will remove the ciphertext/encrypted file with command “rm encrypted.pdf.locked”. After it have finished the process of decryption and deletion on the ciphertext/cipher file, then it will start write into a new file; assume the 4th parameter is “output.txt” then will generate as “output.txt”.

Then return 0; end of the C++ program

# **USER MANUAL (GITHUB)**

## README.MD

README.MD

//Assume user is **in the /bin/sh** environment and **not /bin/bash**

## GIT CLONE FROM GITHUB

To download the program in Ubuntu or Linux environment

# git clone <https://github.com/Applebois/CryptoAssignment2>

Cloning into 'CryptoAssignment2'...

remote: Enumerating objects: 103, done.

remote: Counting objects: 100% (103/103), done.

remote: Compressing objects: 100% (75/75), done.

remote: Total 103 (delta 47), reused 68 (delta 20), pack-reused 0

Receiving objects: 100% (103/103), 16.33 MiB | 3.41 MiB/s, done.

Resolving deltas: 100% (47/47), done.

# cd CryptoAssignment2/

# ls

\_config.yml Output README.md sample1.gif sample2.gif sample.gif Source

Change program privilege to allow execution after issued “git clone” command

# cd CryptoAssignment2

# cd Output

# chmod u+x seed\_cfb

## INSTALLATION

Install the crypto++ library or else might not able to build/compile in later.

#sudo apt-get update

Then, issue next command to install crypto++

#sudo apt-get install libcrypto++-dev libcrypto++-doc libcrypto++-utils

### COMMAND TO COMPILE/BUILD

How to compile the source code after issued “git clone” command, and will save as seed\_cfb using parameter -o

# cd CryptoAssignment2

# cd Source

# g++ -g3 -ggdb -O0 -Wall -Wextra -Wno-unused -o seed\_cfb source.cpp -lcryptopp

## How to issue help command from the program?

# cd CryptoAssignment2

# cd Output

# chmod u+x seed\_cfb

# ./seed\_cfb -h

[x]1st parameter -e or -d | Encryption or Decryption

[y]2nd parameter key file such as | key.txt

[z]3rd parameter input file | the file you wish to decrypt or encrypt

[$]4th parameter output file | the file you wish to save the result/output

[x -> y]Encryption Usage: ./seed\_cfb -e key.txt plaintext.txt ciphertext.txt

[y -> x]Decryption Usage: ./seed\_cfb -d key.txt encrypted.txt.locked output\_to\_thatfile.txt

[File with spacing]Encryption Usage: ./seed\_cfb -e key.txt "sample docs.pdf" "encrypted file.pdf"

[File with spacing]Decryption Usage: ./seed\_cfb -d key.txt "encrypted file.pdf.locked" "recovered file.pdf"

Written in C++, by Teh Win Sam

# **USAGE DEMONSTRATION**

Content of “key.txt”

# cat key.txt

IAMTEHWINSAMHAHA

TEHWINSAMISGOODS

Content of “plaintext.txt”

# cat plaintext.txt

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB

## DEMO ENCRYPT

How encrypt the file?

# cd CryptoAssignment2

# cd Output

# chmod u+x seed\_cfb

# ./seed\_cfb -e "key.txt" "plaintext.txt" "output.txt"

=====================================================

Encryption mode

=====================================================

key: 49414D54454857494E53414D48414841

iv: 54454857494E53414D4953474F4F4453

File Encrypted

The locked file is renamed into --> 'output.txt.locked'

# cat output.txt.locked

E88AF9567F1E75B56E5B38586F2E33C2353E8DC8EF588636F139DEF711617AFB5A717030D0265057C00CEE3C3DE84639ECBF0C4A6FC005D9760F56B9D6D3

## DEMO DECRYPT

How decrypt the file?

# cd CryptoAssignment2

# cd Output

# chmod u+x seed\_cfb

# ./seed\_cfb -d "key.txt" "output.txt.locked" "plain message.txt"

=====================================================

Decryption mode

=====================================================

key: 49414D54454857494E53414D48414841

iv: 54454857494E53414D4953474F4F4453

File has been decrypted

The decrypted file is named as 'plain message.txt'

# cat plain\ message.txt

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAA

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBB