

The Project Report

Section 3

ID 3

Mohamed Ayman 900182267

AbdEl-Rahman Fawzy 900183004

Shamel Radwan 900183844

The Project Report

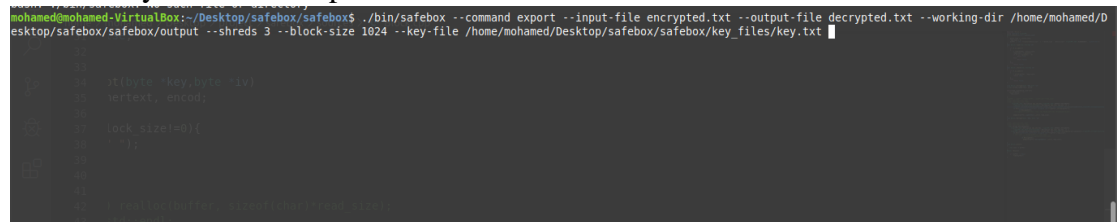
How to run?

1. You need to put the data.txt in the input file.
2. Know the place you are in, and then put it in makefile.vars
3. You need to make sure the parameters are right as the screenshot mentions



```
mohamed@mohamed-VirtualBox: ~/Desktop/safebox/safebox
File Edit View Search Terminal Help
mohamed@mohamed-VirtualBox:~/Desktop/safebox/safebox$ ./bin/safebox --command import --input-file /home/mohamed/Desktop/safebox/safebox/data/data.txt --output-file encrypted.txt --working-dir /home/mohamed/Desktop/safebox/safebox/output --shreds 3 --block-size 1024 --key-file /home/mohamed/Desktop/safebox/safebox/key_files/key.txt
```

4. Make sure to write the right path in input and working directory.
5. Write what number of shreds you need
6. Type the size of each block
7. Run it in the terminal
8. After that you need to export: the screenshot elaborates on what I mean.



```
mohamed@mohamed-VirtualBox:~/Desktop/safebox/safebox$ ./bin/safebox --command export --input-file encrypted.txt --output-file decrypted.txt --working-dir /home/mohamed/Desktop/safebox/safebox/output --shreds 3 --block-size 1024 --key-file /home/mohamed/Desktop/safebox/safebox/key_files/key.txt
```

9. Input file path is the output of the import file. Number of shreds, block size and path of key must be the same
10. Open the decrypted.txt, you will see it is decrypted.

How do I encrypt a plain Text?

Simply, what we have understood is that we need to cipher a plain text using simple techniques and decrypt it: **utilizing AES and CBC_Mode. First.** We have been thinking about calling the class constructor of CBC_Mode as in the block encrypt function elaborates. *The parameters of CBC_Mode* needs the key and iv, so we have gone to the **Shred Manager class**, reading what is inside the key file to a dynamic array of byte, and generating a random iv as required in the conventional world in CBC_mode. Back to the **encrypt block Function**, the function needs to be ciphered by put it in a StringSource function; then StreamTransformationFilter will pipe everything to StringSink. Afterwards, I need to put the Cipher_Text inside the buffer, but before so, I need to use MessageEnd to finalize everything. Return to the shred Manager encrypt function, I need to perform a "for loop" to encrypt the all blocks. So I need to perform the function of **Spooler for this loop**. This function says that if this block does not load anything, return null, otherwise return the block which is loaded. Come back to **shredManager** encrypt function, this loop will do as the following: Taking each block and put it to the array of shred! I need to complete then the **constructor** of **shredManager**. When we have been thinking about a convenient solution, we can make `Shred[i] = newShred(parameters)`. But what about decryption? What is the problem of decryption? Let me explain then the problem of padding! When encrypting the block, an extension happens in the size of ciphertext. So I need to make if condition. If we are encrypting, take the size of block. But to put the ciphered, extended text, I will use the assignment of `(blocksize+16)~15` to extend the size of decrypted, so we can fill the decrypted shreds to fill it in the file without any problem. Back to the for loop, I need to to encrypt each block and append it.

How do I decrypt the cipheredText?

We have made two ways to solve this problem. The first one is using the assignment I have mentioned in the first Paragraph. **A second way is that** (it is under the folder project) we need to handle the padding by ourselves, so we need to make a "while loop" in encryption under the condition that $(\text{read_size} \% \text{block_size} \neq 0)$ reallocate the read_size by one. And extend the buffer to be spaces. Not only this, but we also use the ZEROS_PADDING to stop the padding. Anyway, let's begin with how we deciphered the text. Decrypt Block function is simple as the encrypted block function except decrypt instead of encrypt. But in shred Manager, it will be slightly different. I started to take the shred[0]->appendblock(), decrypt it and then with the same loop till b pointing to null.

What about SafeBox?

Simply, it is just to calling functions and passing the path of each one of them.