

Cryptographic Techniques for Network Security

By: Valdir Santos Fonseca - W1669150

Table of Contents

1. Introduction
2. Decryption Methodology
 - a. Encrypted File
 - b. Base64 Decoding
 - c. Decryption of the file using Java Language
 - i. Encoder Java File
 - ii. Testing the Encryption Algorithm
 - iii. Discovering the Key
3. Results
 - a. Decoding
 - b. Key used to break the File
 - c. Recovery of Plaintext
4. Conclusion
5. References
6. Appendix

Introduction

Securing our files has become a challenge since the growth of technology became a threat to our privacy, protecting our information has been the top priority for us due to the high volume of hackers nowadays, because no one wants his private information exposed to the world or been used to cause material or human damage for example, so encrypting our information in this case, files, is one of many techniques used to secure privacy from third parties, hackers for example. Who wants information to be secure and confidential? I think everyone, so in this report my aim is to show one of those techniques that has been used for many people to keep their information visible just for them, avoiding access from outside and the risk to have that data exposed.

Decryption Methodology

Encrypted File

To apply the decryption code, has been given base64 encoded file with our ID, where the key and the plain text is different. So that's mean the original final has been already encrypted from text to binary and using base64 format to encoded, where it gave you a code, an ASCII code.

Base64 Decoding

Because we need to decode the file already in base64 format, I use base64 decoding online tool as mention on the reference, to decode the encoded file instead of doing it character by character.

Decryption of the file using Java Language

With my acknowledge in Java, I use this language to solve the problem instead using the other one, C Language, so I change the main given code to my own, and get the results by asking the user to introduce his key.

i. Encoder Java File

1. The Encoder file (Encoder1.java) helped me to understand the encryption algorithm, so using the decoded file as a source and creating a new file where the content of the Plaintext will be stored after being decoded using our key.

ii. Testing the Encryption Algorithm

1. The Encryption Algorithm will make the use of the given key to decrypt the decode file, so reading key it's going to be turned into a binary number.

iii. Discovering the Key

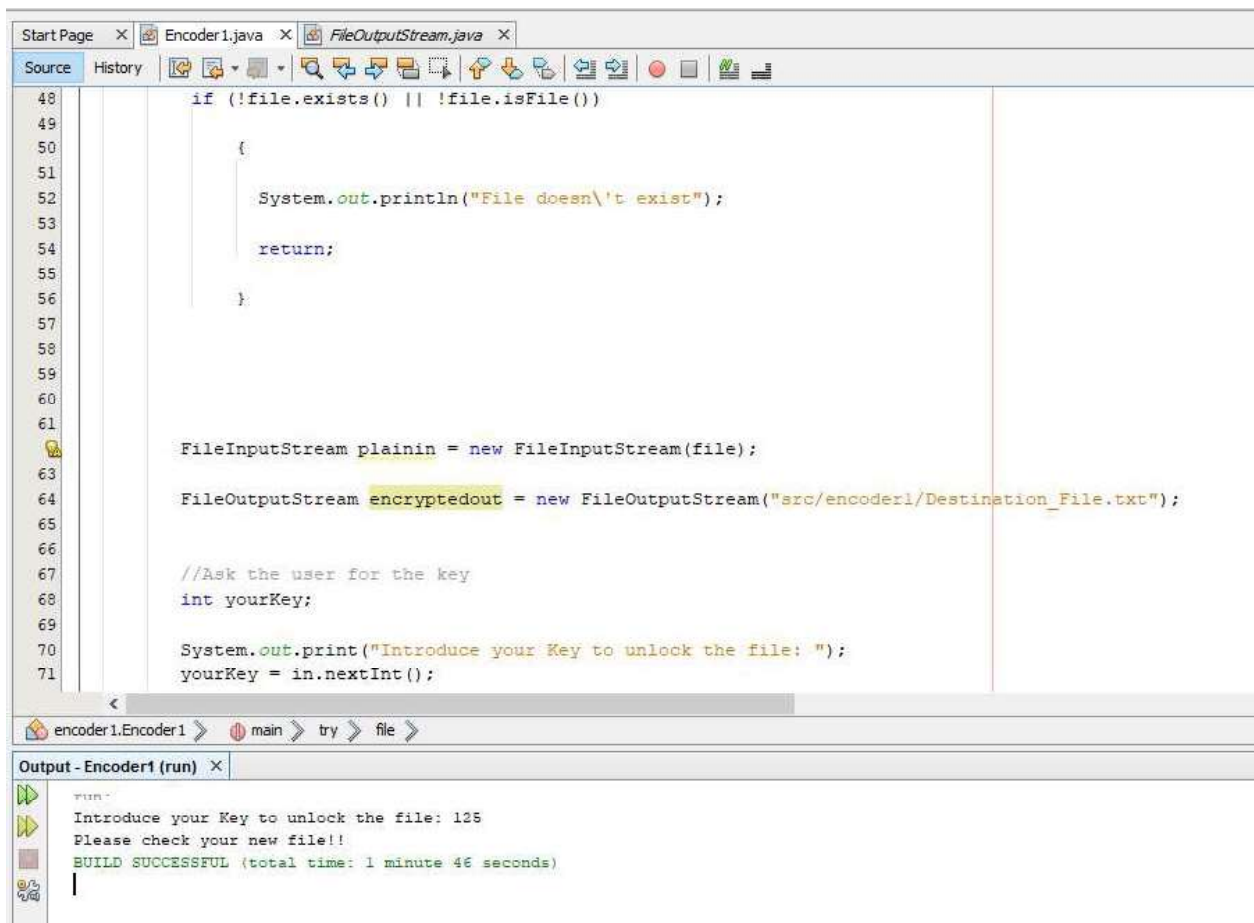
The key given was an integer number between 0 – 255 that we must introduce to get our file decoded, not knowing the code I have tried most of the numbers until get the right one and get the code decrypted. Founded two keys, one giving you a Text without space and disorganised and the second one which was the right one where the Text is organised and included spaces, so those sentences can make sense.

Results

Decoding

[illegible]

Key used to break the code: 125



```
48     if (!file.exists() || !file.isFile())
49     {
50
51         System.out.println("File doesn't exist");
52
53         return;
54     }
55
56
57
58
59
60
61
62     FileInputStream plainin = new FileInputStream(file);
63
64     FileOutputStream encryptedout = new FileOutputStream("src/encoder1/Destination_File.txt");
65
66
67     //Ask the user for the key
68     int yourKey;
69
70     System.out.print("Introduce your Key to unlock the file: ");
71     yourKey = in.nextInt();
```

encoder1.Encoder1 > main > try > file >

Output - Encoder1 (run) X

```
run -
Introduce your Key to unlock the file: 125
Please check your new file!!
BUILD SUCCESSFUL (total time: 1 minute 46 seconds)
```

Recovery of the Plaintext

Destination_File - Notepad

File Edit Format View Help

bit permanent, if of
advantage to the species, and if the insect whose nest and stored food
are thus feloniously appropriated, be not thus exterminated.

SLAVE-MAKING INSTINCT.

This remarkable instinct was first discovered in the *Formica* (*Polyerges*) *rufescens* by Pierre Huber, a better observer even than his celebrated father. This ant is absolutely dependent on its slaves; without their aid, the species would certainly become extinct in a single year. The males and fertile females do no work. The workers or sterile females, though most energetic and courageous in capturing slaves, do no other work. They are incapable of making their own nests, or of feeding their own larvae. When the old nest is found inconvenient, and they have to migrate, it is the slaves which determine the migration, and actually carry their masters in their jaws. So utterly helpless are the masters, that when Huber shut up thirty of them without a slave, but with plenty of the food which they like

Conclusion

To conclude what I can say is this report was important to me, made me understand a little bit about decoding, using base64, so you can encode your file with it to make it secure and decoded it using the same format. I have learned that the security of our files is in our hands so that's mean if you want security and privacy, encrypting our files is one of the best way to keep ourselves secure.

References

<https://www.base64decode.org/>

Lectures from Blackboard and files to download

Appendix

- Because I change the main code of Encoder, so below I'm showing the changed code where the user can introduce his key and visualise the right file.

```
package encoder1;

import java.io.*;
import java.util.Scanner

/**
 * Encrypts a file using a permutation cipher.
 * XOR each byte against a key.
 *
 *
 * @author Valdir Fonseca
 */

public class Encoder
{

    public static void main(String[] args)
    {
        Scanner in = new Scanner(System.in);

        if (args.length !=0)
        {
            System.out.println("USAGE: java Encoder1 plaintextfile encryptedfile key\n");

            System.out.println(" Where plaintextfile is the file you wish to encrypt");

            System.out.println(" and the encrypted version is placed in encryptedfile.");
        }
    }
}
```

```
System.out.println(" key takes a value in the range (0-255).");
```

```
    return;
```

```
}
```

```
try
```

```
{
```

```
    File file = new File("src/encoder1/decodedFile.txt");
```

```
    if (!file.exists() || !file.isFile())
```

```
    {
```

```
        System.out.println("File doesn't exist");
```

```
        return;
```

```
    }
```

```
// Reading the file from the computer
```

```
FileInputStream plainin = new FileInputStream(file);
```

```
// Writing on the destination file after doing de decode
```

```
FileOutputStream encryptedout = new  
FileOutputStream("src/encoder1/Destination_File.txt");
```

```
//Ask the user for the key
```

```
int yourKey;
```

```
System.out.print("Introduce your Key to unlock the file: ");
```

```
yourKey = in.nextInt();
```

```
byte key = (byte)yourKey;
```

```
System.out.println("Please check your new file!!");
```

```
// Should check key is in the right range
```

```
while((yourKey = plainin.read()) >= 0 && (yourKey = plainin.read()) <=255)
{

    // The following line is the encryption algorithm

    byte ec = (byte) (yourKey^key);

    encryptedout.write(ec);

}

plainin.close();

encryptedout.close();

}

catch (Exception exception)

{

    exception.printStackTrace();

}

}
```