File Encryption with AES

Antonino Tan-Marcello
MSCS 630: Security Algorithms and Protocols
Professor Rivas

Abstract

This paper illustrates the implementation of the Advanced Encryption Standard, more specifically as outlined by this paper, Java programs designed to securely encrypt and decrypt computer files. The programs have shown to successfully encrypt several types of files, making the data within the files undecipherable and decoding them virtually impossible without the proper decryption procedure. When prompted, the program will also successfully decrypt the files back to their original state, as if they were never altered.

Introduction

Sensitive information must always be protected in one form or another. Securing information digitally, however, can actually be the most reliable, especially when AES comes into the picture. It is frequently seen in action movies where a computer chip/hard drive holds top secret information or programs that everyone wants to get their hands on. Most of the time, once it’s in possession, whatever data was on that device is already exposed to that individual. However, if the data on that storage device has been encrypted, then it would be virtually impossible to decipher that data. Actual applications today carry the same idea. File encryption can protect information on unattended or shared computers such as school, office or home computers. If someone has their device stolen or lost, they can be sure that their data is still safe if it is encrypted. Also, if a hacker gained remote access to your computer, they would not be able to retrieve any sensitive information, as they still have encrypted files to deal with. Encrypting files is also a good idea if a person utilizes databases or cloud storages, as there are people trying to intercept any valuable information. This leads up to the motivation behind this project, to directly encrypt computer files in order to protect information and recognize how much file encryption can substantially increase the security of information.

Background

A study conducted by Khati et al. titled Full Disk Encryption: Bridging Theory and Practice, have faced an issue of full disk encryption and storage limits. These researchers wanted to solve the issue of running out of storage space during the encryption process due to the additional storage of initialization vectors (IV) and/or message authentication codes (MAC). These researchers have proposed the notion of a diversifier, which does not use any additional storage space while allowing sectors of a hard drive to be encrypted. Khati et al. have shown that with a 2-bit diversifier, they can decrease the number of input/output operators. Perhaps a way Khati et al. could consider is to eliminate the authentication process, where a password is written to each file or each hard disk sector.

Methodology

The approach taken for this paper and project is to create user-friendly encryption/decryption programs for individuals who would like to encrypt specific files. Using Java on the Netbeans IDE, as well as implementing AES at 128-bits using a symmetric key, the goal is to efficiently and successfully write these programs. It was necessary to initially research published research papers, understand how AES works as well as getting familiar with designing a simple, user-friendly application. The main outline of the programs was to have one java file contain the jPanel, which holds the code for the UI and driver of the program. The second java file contains the AES core, in which the buttons of the UI are linked to. The AES core file contains all the necessary algorithms and tables needed to correctly execute the AES encryption and decryption procedures. At the end, I resulted in two similar versions of the program. The first version utilizes java's crypto libraries for AES and can encrypt/decrypt any file type. The second version implements AES from scratch and will encrypt/decrypt text files.

Experiments

After completing various tests of the two versions of the program, they were able to correctly encrypt and decrypt various types of files using AES at 128-bits. Files used to test the programs' encryption capabilities included several photo files varying in size, Word documents containing text and text files. The second version of the program (AES from scratch) used the text files for testing, and the first version of the program (AES using java's crypto libraries) tested the remaining file types. A total of 5 trials were conducted and recorded per file type. The resulting encrypted files were used to test the programs' decryption capabilities, in order to see if the encrypted files can be transformed back to their original state. Also, to encrypt an entire directory containing multiple files all at once, a test was conducted by compressing the folder to a .zip file and was followed by the encryption and decryption procedures.

Results

1.) Text file (Program version 2: AES implementation from scratch)

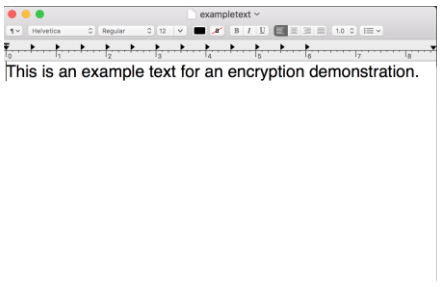
Table 1. Text file (.rtf) encryption and decryption processes elapsed times, file size is 394 in bytes.

Text file	Encryption time (ms)	Decryption time (ms)
Trial 1	47	17
Trial 2	11	6
Trial 3	9	4
Trial 4	25	3
Trial 5	6	3
Average	19.6	6.6

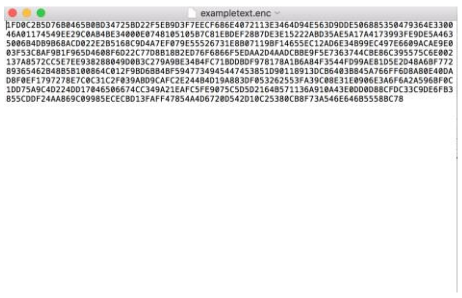
Original

Encrypted

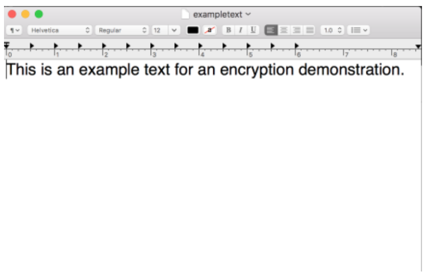
Decrypted



This is an example text for an encryption demonstration.



```
1FD0C2B5D76B0465B0BD34725BD22F5EB9D3F7EECF686E4072113E3464D94E563D9DDE506885350479364E3300
46A01174549EE29C0AB4BE34000E0748105105B7C81EBDEF28B7DE3E15222ABD35AE5A17A4173993FE9DE5A463
5006B4DB9B68ACD022E2B5168C9D4A7EF079E55526731E8B07119BF14655EC12AD6E34B99EC497E6609ACAE9E0
03F53C8AF9B1F965D4608F6D22C77D8B18B2ED76F6866F5EDAA2D4AADCBBE9F5E7363744CBE86C395575C6E002
137A8572CC5E7EE9382B8049D0B3C279A9BE34B4FC71BDD8DF978178A1B6A84F3544FD99AE81D5E2D48A6BF772
B9365462B48B5B100864C012F9BD6BB4BF594773494544745385 1D90118913DCB6403BB45A766FF6D8A80E40DA
D8F0EF1797278E7C0C31C2F039ABD9CAFC2E244B4D19A883DF053262553FA39C08E31E0906E3A6F6A2A596BF0C
1DD75A9C4D224DD17046506674CC349A21EAFC5FE9075C5D5D2164B571136A910A43E0DD0D88CFDC33C9DE6FB3
855CDDF24AA869C09985ECECBD13FAFF47854A4D6720D542D10C25380CB8F73A546E646B5558BC78
```



This is an example text for an encryption demonstration.

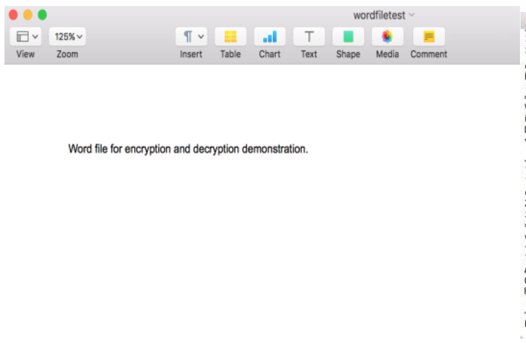
Figure 1. Text file (.rtf) screenshots of original, encrypted and decrypted.

2.) Word file (Program version 1: java crypto libraries with AES)

Table 2. Word file (.docx) encryption and decryption processes elapsed times, file size is 4,307 in bytes.

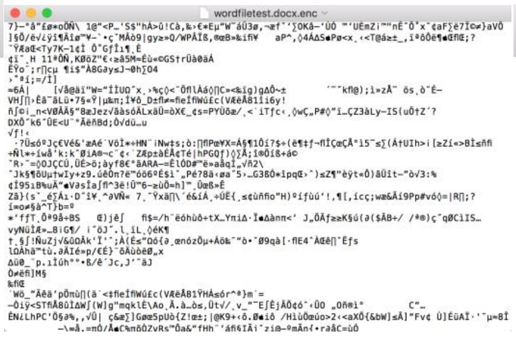
Word file	Encryption time (ms)	Decryption time (ms)
Trial 1	1	5
Trial 2	5	5
Trial 3	2	1
Trial 4	7	5
Trial 5	5	6
Average	4	4.4

Original



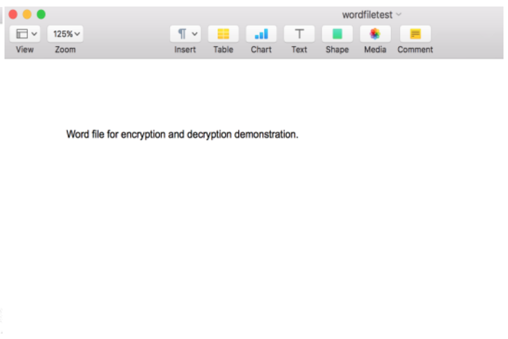
Word file for encryption and decryption demonstration.

Encrypted



[illegible]

Decrypted



Word file for encryption and decryption demonstration.

Figure 2. Word file (.docx) screenshots of original, encrypted and decrypted. Encrypted file could only be opened in a text editor.

3.) Image file (Program version 1: java crypto libraries with AES)

Table 3. Image file (.png) encryption and decryption processes elapsed times, file size is 37,075 in bytes.

Image file	Encryption time (ms)	Decryption time (ms)
Trial 1	7	2
Trial 2	1	4
Trial 3	1	2
Trial 4	1	1
Trial 5	1	1
Average	2.2	2

Original Encrypted Decrypted

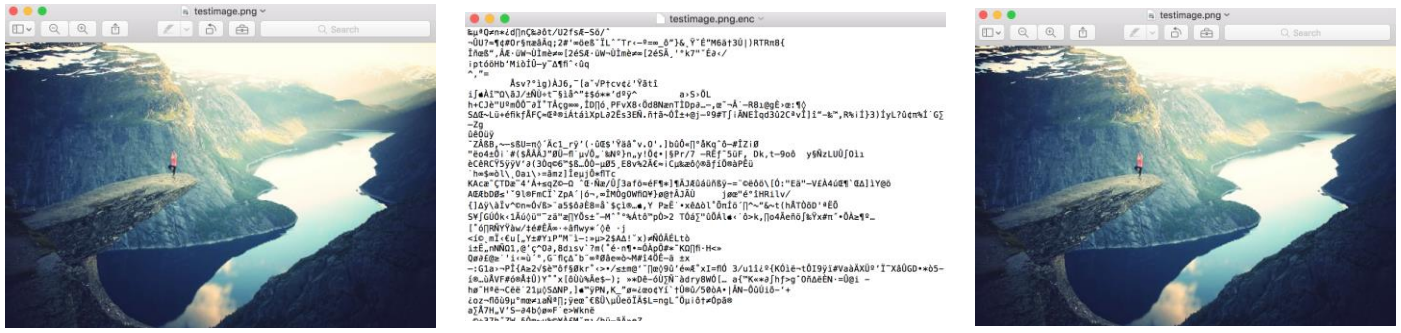


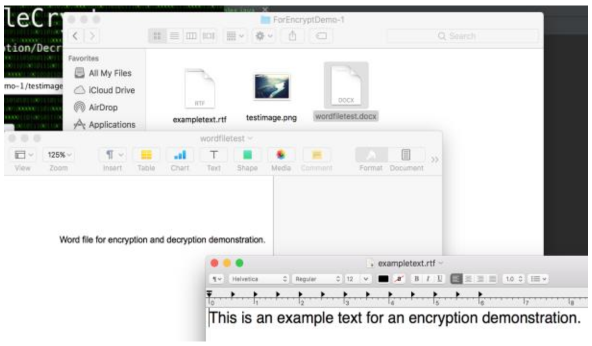
Figure 3. Image file (.png) screenshots of original, encrypted and decrypted. Encrypted file could only be opened in a text editor.

4.) Compressed files (Program version 1: java crypto libraries with AES).

Table 4. Compressed file (.zip) encryption and decryption processes elapsed times, file size is 42,780 in bytes.

Compressed file	Encryption time (ms)	Decryption time (ms)
Trial 1	1	1
Trial 2	1	1
Trial 3	1	1
Trial 4	1	2
Trial 5	1	1
Average	1	1.2

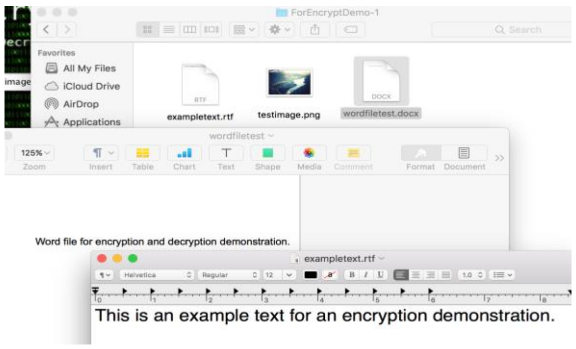
Original



Word file for encryption and decryption demonstration.

This is an example text for an encryption demonstration.

Decrypted



Word file for encryption and decryption demonstration.

This is an example text for an encryption demonstration.

Figure 4. Contents of directory before compressing/encryption and decryption/decompressing. Encrypted .zip file could not be decompressed.

Discussion

After performing some test cases, it is clear that the AES implementation towards the files has shown an obvious effect on their contents. For example, when the programs encrypted a picture file, the photo no longer represents an actual picture. Instead, when opening the file, a text editor containing many, meaningless characters appeared, leaving no trace of the original photo. When decrypting the encrypted picture, the file opens and displays the same exact original photo, as if it was untouched. The same expected results occurred when encrypting a word document. When encrypted, the file opens showing undecipherable characters. Once decrypted, the original document and its contents are shown. When decrypting the encrypted .zip files, the original contents all remained, exactly how they were originally. Version 1 of the program that utilizes java's crypto libraries for AES have shown to run the encryption/decryption processes a lot faster when comparing elapsed times in milliseconds.

Conclusion

From the results obtained by testing our programs, AES can be utilized to successfully secure computer files, as well as turning those files back to their original state. An application similar to the one applied to this project can be used to help people secure any type of document or personal data one may not want others to have access to. From our results, it is clear that encrypted files through AES can completely hide any data, and in order to get that information back, a decryption procedure with the correct key must be applied to those encrypted files. Otherwise, retrieving the encrypted information would be virtually impossible.

References

1. Bossi, Simone, and Andrea Visconti. "What users should know about Full Disk Encryption based on LUKS." *International Conference on Cryptology and Network Security*. Springer International Publishing, 2015.
2. Chakraborty, Debrup, Cuauhtemoc Mancillas López, and Palash Sarkar. "Disk encryption: do we need to preserve length?" *Journal of Cryptographic Engineering* (2015): 1-21.
3. Khati, Louiza, Nicky Mouha, and Damien Vergnaud. "Full Disk Encryption: Bridging Theory and Practice." *Cryptographers' Track at the RSA Conference*. Springer, Cham, 2017.
4. Thakur, Jawahar, and Nagesh Kumar. "DES, AES and Blowfish: Symmetric key cryptography algorithms simulation based performance analysis." *International journal of emerging technology and advanced engineering* 1.2 (2011): 6-12.
5. Singh, Gurpreet. "A study of encryption algorithms (RSA, DES, 3DES and AES) for information security." *International Journal of Computer Applications*67.19 (2013).