LOYOLA UNIVERSITY CHICAGO
COMP 447-001 - INTRUSION DETECTION - FALL 2014

Tiago de Almeida - tdealmeida@luc.edu
1394611

# Graduate Project 2
# Programming

Quick reference:
Implemented algorithm: Simplified AES (S-AES)
Programming language: C
Compiler: GCC 4.8.1
Operating System: OSX Yosemite Beta 10.10

Objective and method:

For this part of the project my goal was to implement a program that could encrypt any file based on any key provided by the user. The objective is to be able to secure any local file on a computer based on a key the user can choose, and be able to retrieve the information back with the same key. That is, a program which implements a secret key cryptographic algorithm.

The Simplified AES (S-AES) described in the Appendix 5B of (Stallings, 2013) was chosen for this goal. According to (Stallings, 2013), this algorithm is mainly focused on educational purposes and, since it is a simplified form of the AES algorithm, it does not provide the same level of security as AES does. However, it has similar properties and structure to AES, and a good grasp of S-AES makes it easier to appreciate the structure and workings of AES.

The S-AES algorithm works with a 16-bit key over a 16-bit block of data. In order to be able to work with bigger files, the algorithm was extended. Every 16-bit of data from the input plaintext is encrypted with 16-bit of the key. The following 16-bit of data is encrypted using the following 16-bit of the key. Once the end of the key is reached, the following 16-bit of data starts from the beginning of the key again. The data is encrypted in key-sized chunks, each one divided in 16-bits blocks. If the final data chunk is smaller than the key, it is encrypted using just the necessary part of the key. This way, padding is not necessary to complete the final chunk of data. Similarly, if the total data is smaller than the key, just the necessary part of the key is used.

The implementation works well and is able to cryptograph any kind of file. Text, image, audio and video files of varied sizes were tested and successfully encrypted and decrypted. The resulting algorithm let us secure any file, of any size, with any key. The key can use any type of character and be of any size. It gives the user freedom to use any key and encrypt any file.

The objective was successfully reached. However, a weakness on this strategy was identified. Since every 16-bit block of data is encrypted individually and is not related to a previous of following block, it is possible to partially decrypt the ciphertext when a small fraction of the key is incorrect. Therefore, the algorithm is safer when using bigger keys, where it's difficult to guess a big part of the key correctly.

The algorithm uses permutation and substitution techniques implemented through the functions described in Appendix 5B of (Stallings, 2013). Such functions are organized in three rounds.

Tools:

The program was developed using the programming language C, the compiler GCC 4.8.1 and tested on the operating system OS X Yosemite Beta 10.10. However, it must work in most Linux distributions as well as with previous versions of GCC. The Library galois.h, Fast Galois Field Arithmetic Library in C/C++, was used to calculate the single Galois Field multiplication necessary in the Mix Column function. This library is implemented by Professor James S. Plank from the University of Tennessee Knoxville, and it is free software under the terms of the GNU Lesser General Public License. (Plank, 2007)

Building:

```
./do
```

Encrypting:

```
./saes encrypt file_name key
```

Decrypting:

```
./saes decrypt file_name key
```

Obs1: The decryption method does not add the correct extension (.txt, .jpg. .mp3) of the original plaintext file to the final decrypted file. The data is the same, but the extension might be added manually if necessary.

Obs2: When running from the command line, be careful with characters in the key which may mean a command or something the shell can interpret, like the character &.

Obs3: In order to prevent users of writing to existing files, the program exits when the output file already exist. For example, if you try to decrypt a file you have already decrypted, with the same name, the program will tell you the output file name it is trying to create, and show the following error: "Error opening output file: File exists."

References:

Stallings, W. (2013). Chapter 5 Advanced Encryption Standard. In *Cryptography and network security: Principles and practice* (6th ed., pp. 184-193). Prentice Hall.

Plank, J. (2007). Fast Galois Field Arithmetic Library in C/C++. Retrieved October 1, 2014, from http://web.eecs.utk.edu/~plank/plank/papers/CS-07-593/