Πολυτεχνείο Κρήτης

Σχολή Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Security systems

Assignment No.1

Ομάδα LAB41446304

Τρίμας Χρήστος	***2016030054***

Purpose of Exercise:

Implement a few basic crypto algorithms in C, using linux environment.

Files:

In the folder, there are four files.

simple_crypto.h : Contains three functions that implement the three encryption and decryptions algorithms(One Time Pad, Caesar cipher, Vigenere cipher). It also contains every useful library that one needs to compile and run the algorithms. Finally, there are three helper functions to make life easier for the demo program.

-randomGenerator function, returns a random string equal to the size of the plaintext, that is needed for the OTP algorithm. This function utilizes the /dev/urandom file of the linux system to create that randomness.

-nonBufferedOvfInput function, returns the user input. In order to avoid buffer overflow attacks, and make my program take as input, unknown size strings, I use malloc and realloc to change the buffer size. The nonBufferedOvfInputUpper function, does the same, except that takes the input and transforms it in uppercase letters. This function is vital for the vigenere algorithm, if the user gives a lowercase symbol in his input.

simple_crypto.c: This is a c file, that implements the functions of the above library.

demo.c: Contains the main function that one can compile and run to check the algorithms.

Makefile: a makefile that compiles and runs the demo file.

Implementation:

Otp_encrypt_decrypt function, finds the length of the user input text, and for every symbol in the random generated key, performs XOR with every symbol of the plaintext respectively.

Caesars_cipher function, first creates a new Alphabet, or a buffer that contains every possible character, from 0-9A-Za-z. Then for the encryption, the program searches for a match, between the input characters and the new Alphabet. If there is a match then the character is incremented by the key and the modulo operation is applied in order to remain in the boundaries of the buffer. In other words, I have created a circular array, that if for one reason the symbol gets out of bounds, it starts over. For the encryption, things are pretty much the same, except that I subtract the key, from the encrypted character and then add the key times 62(the number of characters in the new Alphabet). This ensures, that no matter how big the key the modulo operation will be successful.

Vigenere function, a lot similar to the Caesar's cipher, first I ensure that the encryption key is the same size as the plaintext, and then we add each key to each symbol. The whole process is similar to above, but this time I apply the modulo operator with 26, that are the symbols of the Alphabet(A-Z).