אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

The Department of Information Systems Engineering

Security of Computers and Communication Networks

(372-1-4601)

Assignment #2

Due date: 21.04.2016 23:59

Submission guidelines

- **Goal**: The goal of this assignment is to understand and get hands-on experience with the AES encryption algorithm.

- **Answering all the questions is mandatory.**

- Before submitting make sure your code complies and follows the command line interface defined in this assignment. Make sure it is running in reasonable time duration (not more than 2 minutes per operation).

- It is allowed to submit the assignment in pairs.

- You are required to submit a ZIP file named Ex2_ID1_ID2.zip including the following files:
    - A pdf/word file with answers to Part 1
    - All your source files and the aes.py/jar/exe file for Part2

- An automatic script will test your programs. Programs that will fail to run for whatever reason will be graded 0.

- Your code will be tested using automatic code analysis tools to detect copied code. Avoid any discomforts by simply not cheating or copying others work.

- If you need more information **Google it!** Still have questions? Use the course forum on Moodle.

- Submit your answers to 'Assignment 2' task on the Moodle website. Last submission date is: 21.04.2016 23:59.

Part 1 – Theoretical Questions

Question 1.1:

In this question you need to break a simplified version of the AES encryption algorithm, marked as AES_1. This version of AES uses only one key K as is (meaning, no manipulations are performed on the key) and performs only one iteration of the AES algorithm.

Notations:

- $\boldsymbol{M}$ – Message (128 bit)
- $\boldsymbol{C}$ – Cipher (128 bit)
- $\boldsymbol{K}$– Key (128 bit)
- $\boldsymbol{C = AES\{M\}_K}$ – The cipher C is the encryption of message M using the encryption algorithm AES with the key K

Description of $\boldsymbol{AES_1}$:

- AES_1 - a single round implementation of AES
- $AES_1\{M\}_K = AddRoundKey\left(MixColumns\left(ShiftRows\big(SubBytes(M)\big)\right), K\right) = C$
- For a given message M, the AES_1 will perform: (1) $SubBytes$, (2) $ShiftRows$, (3) $MixColumns$ and (4) $AddRoundKey$ using a given K.
- $AES_1^{-1}\{C\}_K = SubBytes^{-1}\left(ShiftRows^{-1}\big(MixColumns^{-1}(AddRoundKey(C, K))\big)\right) = M$

Given a message M and its cipher $C = AES_1\{M\}_K$ you need to find an **efficient** (practical) method for finding the key K. Describe your method and its time complexity.

Question 1.2:

In this question you need to break another simplified version of AES encryption algorithm, marked as AES_3^*. This version of AES used three different keys: K_1, K_2, K_3 as given (meaning no manipulations are performed on the keys) and performs 3 iterations of the AES algorithm, **without** performing the $MixColumns$ and $SubBytes$ operation in each round.

Definition of $\boldsymbol{AES_1^*}$:

- AES_1^* is a single round implementation of AES, without the $MixColumns$ and $SubBytes$ functions
- $AES_1^*\{M\}_K = AddRoundKey(ShiftRows(M), K) = C$
- $AES_1^{*-1}\{C\}_K = ShiftRows^{-1}(AddRoundKey(C, K)) = M$
-

Definition of $\boldsymbol{AES_3^*}$:

- AES_3^* is the application of AES_1^* three times with three different keys: K_1, K_2, K_3
- $AES_3^*\{M\}_{K_1,K_2,K_3} = AES_1^*\left\{AES_1^*\{AES_1^*\{M\}_{K_1}\}_{K_2}\right\}_{K_3} = C$
- $AES_3^{*-1}\{C\}_{K_1,K_2,K_3} = AES_1^{*-1}\left\{AES_1^{*-1}\{AES_1^{*-1}\{C\}_{K_3}\}_{K_2}\right\}_{K_1} = M$

Given a message M, and its cipher C, where: $C = AES_3^*\{M\}_{K_1,K_2,K_3}$ You need to find an **efficient** method for finding 3 keys K_1, K_2, K_3 that holds: $C = AES_3^*\{M_1\}_{K_1,K_2,K_3}$ Describe your method and it's time complexity.

Part 2 – Practical Exercise

Exercise 2.1:

a. Implement AES_1 as described in question 1.1 of the theoretical questions part.

b. Given a message M and it's cipher $= AES_1\{M\}_K$, implement the method you proposed in your answer to question 1.1 for finding the key K.

Exercise 2.2:

a. Implement AES_3^* as described in question 1.2 (Part 1 - theoretical questions).

b. Given a message M and its cipher C in which: $C = AES_3^*\{M\}_{K_1,K_2,K_3}$ implement the method you proposed in your answer to question 1.2 for finding the keys: K_1, K_2, K_3.

Implementation Notes:

- You are free to implement your code in any of the following programming languages: Java, C#, Python (2.7).
- A message M might be longer than 128 bits. You should consider this fact in your implementation and enable handling longer messages accordingly: break M into 128 bit blocks, and apply the algorithm on each block separately. You can assume that: $M.length \ \% \ 128 \ == \ 0$.
- Note that you are provided with java code containing definitions of the following tables: $SubBytes$, $SubBytes^{-1}$ and $MixColumns$ multiplication results.
- You must submit one of the following file:
  - Java - "aes.jar" (an executable Jar)
  - C# - "aes.exe"
  - Python " aes.py"
- Your submitted program file should implement the following command line interfaces:
- Encryption/Decryption interface:
  - –a <AES1/AES3> : denotes the algorithm to use: "AES1" for AES_1 and "AES3" for AES_3^*
  - –e : instruction to encrypt the input file
  - –d: instruction to decrypt the input file
  - –k <path>: path to the key(s) , the key should be 128 bit for AES_1 or 384 bit (128*3) for AES_3^*. The latter should be divided into 3 separate keys.
  - –i <input file path>: a path to a file we want to encrypt/decrypt
  - –o <output file path>: a path to the output file
  - Usage:
    aes –a <AES1 or AES3> -e/-d –k <path-to-key-file > -i <path-to-input-file> -o <path-to-output-file>
  - Example: AES_3^* encryption test for a Jar submission will be executed using:
    Java –jar aes.jar –a AES3 –e –k key.txt –i message.txt –o cypther.txt
- Hacking (breaking) interface:
  - –a <AES1/AES3> : denotes the algorithm to break: "AES1" for AES_1 and "AES3" for AES_3^*
  - –b: instruction to break the encryption algorithm
  - –m <path>: denotes the path to the plain-text message
  - –c <path>: denotes the path to the cipher-text message
  - –o <path>: a path to the output file with the key(s) found.
  - Usage:
    aes –a <AES1 or AES3> -b –m <path-to-message> –c <path-to-cipher> -o < output-path>
  - Example: Breaking AES_1 test for a Python submission will be executed using:
    python aes.py –a AES1 –b –m message.txt –c cypther.txt –o keyFound.txt
- Input and output formats:
  - You should read and write from/to the input/outputs files in **bytes** and not text.
  - When you read the inputs and write the outputs, make sure you order the bytes correctly according to the state structure (lecture – Cryptographic Algorithms slide 16).
- You will be provided with inputs and outputs examples, use them for testing your code.

Good Luck!