

Assignment 8

IDATT2503 - Security in programming and cryptography
Fall 2023

Francesca Grimaldi

1 Task 1 - AES a little history

1.1 The evolutionary history of AES differs from that of DES. Briefly describe the differences of the AES history in comparison to DES.

The **Data Encryption Standard (DES)** is a symmetric-key block cipher that uses a 64 bits key: it encrypts data 64 bits at a time. It was the result of a research project set up by IBM in the late 60s which resulted in a cipher known as LUCIFER. When it was decided to commercialize it (in the early 70s), IBM introduced significant changes to the algorithm after receiving technical advice from the **National Security Agency (NSA)**. This new version was published as an official Federal Information Processing Standard (FIPS) for the United States in 1977, but it was severely criticized for two main reasons:

1. The algorithm takes a 64 bit key input, but 8 bits are used for parity checking and are immediately discarded. The actual key is 56-bits long and is considered too short.
2. It was thought that S-boxes had some secret trapdoor that could enable the NSA to decrypt messages outside the requirement for the key.

Because DES was starting to become vulnerable to brute-force attacks, a new algorithm was required.

The **National Institute of Standards and Technology (NIST)** initiated a public, peer-reviewed competition to choose a new encryption standard. After a process that lasted from 1997 to 2000, the **Rijndael algorithm** was selected to become the **Advanced Encryption Standard (AES)**. Compared to its predecessor, this procedure was noticeably more transparent and open because NIST also asked for input from interested parties on how to choose the algorithm.

AES is a symmetric-key block cipher as well, but it supports key lengths of 128, 192, and 256 bits, offering more flexibility and security against modern cryptographic attacks.

1.2 What is the name of the algorithm that is known as AES?

The original name of the algorithm that is now known as AES is *Rijndael*.

1.3 Who developed this algorithm?

This algorithm was developed by *Vincent Rijmen* and *Joan Daemen*, two Belgian cryptographers.

1.4 Which block sizes and key lengths are supported by this algorithm?

Before being chosen as AES, the Rijndael algorithm supported all block lengths and key sizes multiples of 32 bits, between 128 and 256 bits. After the standardization process, the parameters were restricted to block size of 128 bits and key lengths of 128, 192 or 256 bits.

2 Task 2

The `task2.py` Python file contains the code used for One-time pad and Affine ciphers, whereas the website cryptool.org was used for AES.

2.1 Diffusion

The results of the encryption of both x1 and x2 using the key K1 for different ciphers are the following:

A) One-time pad (XOR)

x1 -> 0133456789abcdef0123456789abcdef

x2 -> 0103456789abcdef0123456789abcdef

Two bits change between the plaintexts, and the ciphertexts also differ by two bits. The *diffusion* is very low: there is much statistical correlation between plaintext and ciphertext.

B) Affine cipher

x1 -> e013456789abcdef0123456789abcdef

x2 -> bf03456789abcdef0123456789abcdef

Diffusion increases to a percentage of 5.46875% with the use of affine cipher, as 7 bits change. Although it is a better value than the previous one, it is still low.

C) One round of AES

x1 -> b3543dccec87235705c3aa65640fabdf

x2 -> 834c25e4ec87235705c3aa65640fabdf

Here we have a diffusion of 6.25% as 8 bits out of 128 change with the changing of 2 bits between the plaintexts. The value is low.

D) Full AES

x1 -> 0694267ba398480c6b2b9f649be476cb

x2 -> 282f7b11019800f8a978c6f750827ab5

The diffusion in this case is of 47.65625%. Approximately half of the bits of the ciphertext change, which means that the *diffusion* is good.

2.2 Confusion

The results of the encryption of x1 using both keys K1 and K2 for different ciphers are the following:

A) One-time pad (XOR)

x1 using K1 -> 0133456789abcdef0123456789abcdef

x1 using K2 -> 1133456789abcdef0123456789abcdef

Changing one bit in the key, only one bit changes between the ciphertexts, which is a very low value for the *confusion* property.

B) Affine cipher

x1 using K1 -> e013456789abcdef0123456789abcdef

x1 using K2 -> f013456789abcdef0123456789abcdef

Also in this case, *confusion* is very low because changing one bit in the key, only one bit changes between the two ciphertexts.

C) One round of AES

x1 using K1 -> b3543dccec87235705c3aa65640fabdf

x1 using K2 -> eafd942cfc87235715c3aa65740fabdf

In this case, 18 bits change (in percentage 14.0625%). *Confusion* is better compared to the previous ones but is still a moderate value.

D) Full AES

x1 using K1 -> 0694267ba398480c6b2b9f649be476cb

x1 using K2 -> 7af49a8defad94fa27cb03ac9f1c149a

The number of different bits between the two ciphertexts is 61, so nearly a half. AES is a cipher with a good level of *confusion*, in the sense that a change in the key affects the calculation of most of the ciphertext.

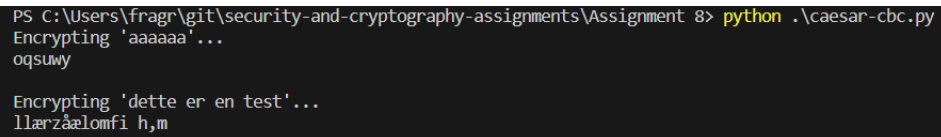
3 Task 3

The following subsections contain the outputs of the encryption and decryption of the given messages, obtained using the Caesar cipher at byte level ($E(x) = x + 3 \bmod 32$) and CBC mode using bitwise XOR, with initialization vector $IV = 13$.

The code used for this task is available in the `caesar-cbc.py` Python file.

3.1 Encryption

- Encryption of aaaaaa: oqsuwy
- Encryption of dette er en test: llærzåælomfi h,m

A terminal window with a black background and white text. The prompt is 'PS C:\Users\fragr\git\security-and-cryptography-assignments\Assignment 8>'. The user enters 'python .\caesar-cbc.py'. The output is 'Encrypting 'aaaaaa'...' followed by 'oqsuwy' on the next line. Then the user enters 'python .\caesar-cbc.py' again, and the output is 'Encrypting 'dette er en test'...' followed by 'llærzåælomfi h,m' on the next line.

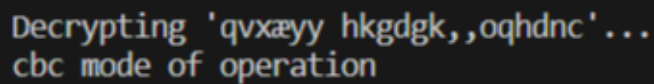
```
PS C:\Users\fragr\git\security-and-cryptography-assignments\Assignment 8> python .\caesar-cbc.py
Encrypting 'aaaaaa'...
oqsuwy

Encrypting 'dette er en test'...
llærzåælomfi h,m
```

Figure 1: Output of the encryption

3.2 Decryption

- Decryption of qvxæyy hkgdgk,,oqhdnc: cbc mode of operation

A terminal window with a black background and white text. The output is 'Decrypting 'qvxæyy hkgdgk,,oqhdnc'...' followed by 'cbc mode of operation' on the next line.

```
Decrypting 'qvxæyy hkgdgk,,oqhdnc'...
cbc mode of operation
```

Figure 2: Output of the decryption