Understandig the Galois Counter Mode (GCM)
HW3 - CNS Sapienza

Edoardo Puglisi 1649359

21/11/2019

Contents

1 Overview

The following experiment consists of evalutating GCM encryption and decryption speed on binary and text files.

2 Code

Code description.

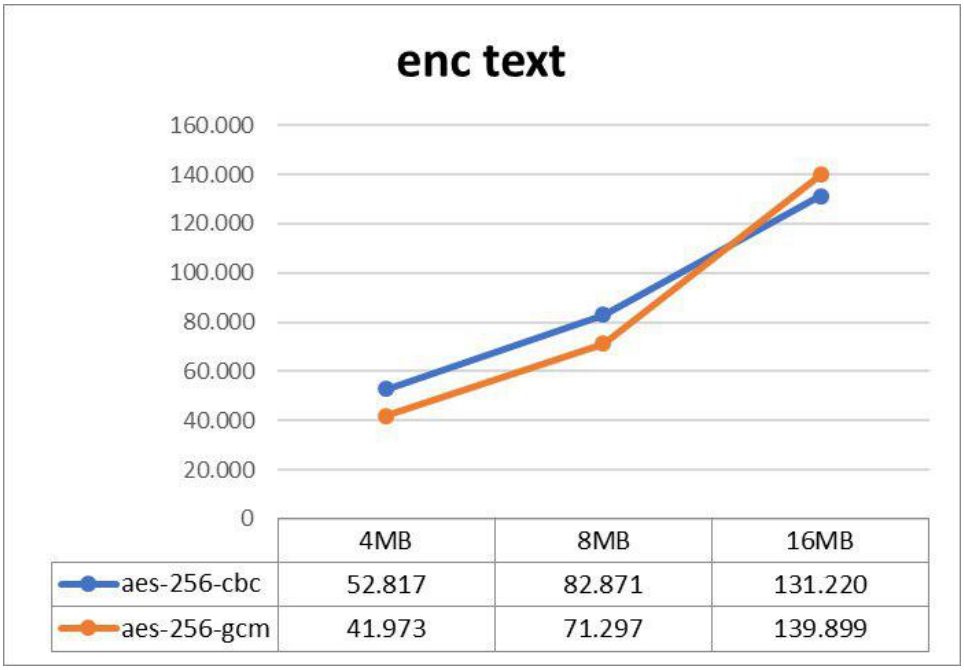
```
#!/bin/bash

PASSWORD="passwordmoltosicura"
OUTPUT="result.hw3-1649359.csv"
CICLI=10
truncate -s 0 $OUTPUT

echo ++++++ HOMEWORK 3 ++++++

for cipher in aes-256-cbc aes-256-gcm
do
    echo $cipher >>  $OUTPUT
    echo +++++ CIPHER $cipher +++++
    for text in text1 text2 text3
    do
        echo ===== ENCRYPTION =====
        echo $text >>  $OUTPUT
        runtime=0
        for i in {1..$CICLI}
        do
            start=$(date +%s%N)
            openssl $cipher -a -salt -pbkdf2 -in "
               $text-hw3-1649359.txt" -out "$text.
               enc" -pass pass:$PASSWORD
            runtime="$(($runtime+($(date +%s%N)-
               $start)))"

        done

        enc_runtime_scaled=$(bc <<< "scale=3;
           $runtime/$CICLI/1000000")
        echo ENC_SPEED $enc_runtime_scaled

        echo ===== DECRYPTION =====
```

```
        runtime=0
        for i in {1..$CICLI}
        do
            start=$(date +%s%N)
            openssl $cipher -d -a -salt -pbkdf2 -in
               "$text.enc" -out "$text.new.txt" -
               pass pass:$PASSWORD
            runtime="$(($runtime+($(date +%s%N)-
               $start)))"

        done

        dec_runtime_scaled=$(bc <<< "scale=3;
           $runtime/$CICLI/1000000")
        echo DEC_SPEED $dec_runtime_scaled
        echo $size >> $OUTPUT
        echo ENC";"$enc_runtime_scaled >> $OUTPUT
        echo DEC";"$dec_runtime_scaled >> $OUTPUT

    done

    for bin in 1024 3238 10240
        do
        dd if="/dev/urandom" of="$bin.file" bs="$bin
           " count="$bin"
        echo ===== ENCRYPTION =====
        echo $bin >>  $OUTPUT
        runtime=0
        for i in {1..$CICLI}
        do
            start=$(date +%s%N)
            openssl $cipher -a -salt -pbkdf2 -in "
               $bin.file" -out "$bin.enc" -pass pass
               :$PASSWORD
            runtime="$(($runtime+($(date +%s%N)-
               $start)))"

        done

        enc_runtime_scaled=$(bc <<< "scale=3;
           $runtime/$CICLI/1000000")
        echo ENC_SPEED $enc_runtime_scaled

        echo ===== DECRYPTION =====
```

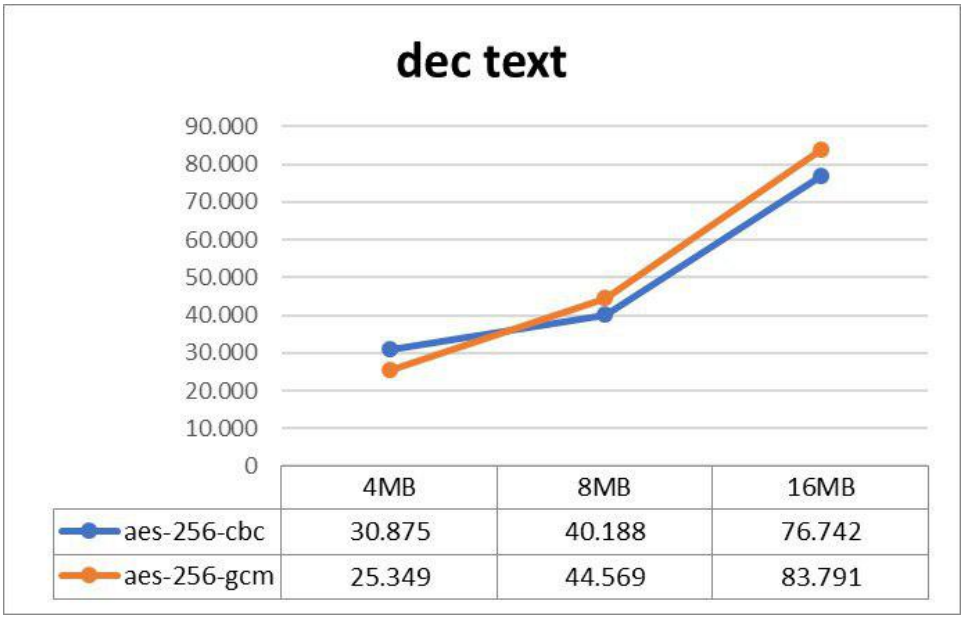
```
        runtime=0
        for i in {1..$CICLI}
        do
            start=$(date +%s%N)
            openssl $cipher -d -a -salt -pbkdf2 -in
               "$bin.enc" -out "$bin.new.file" -pass
                pass:$PASSWORD
            runtime="$(($runtime+($(date +%s%N)-
               $start)))"

        done

        dec_runtime_scaled=$(bc <<< "scale=3;
           $runtime/$CICLI/1000000")
        echo DEC_SPEED $dec_runtime_scaled
        echo $size >> $OUTPUT
        echo ENC";"$enc_runtime_scaled >> $OUTPUT
        echo DEC";"$dec_runtime_scaled >> $OUTPUT

    done


done

for bin in 1024 3238 10240
        do
        rm "$bin.file"
        rm "$bin.enc"
        rm "$bin.new.file"
done

for text in text1 text2 text3
    do
    rm "$text.enc"
    rm "$text.new.txt"
done
```

This code simply encrypt and decrypt files of different size (text and binary). The process is repeated multiple times for each file to have an average time speed.

3 Results

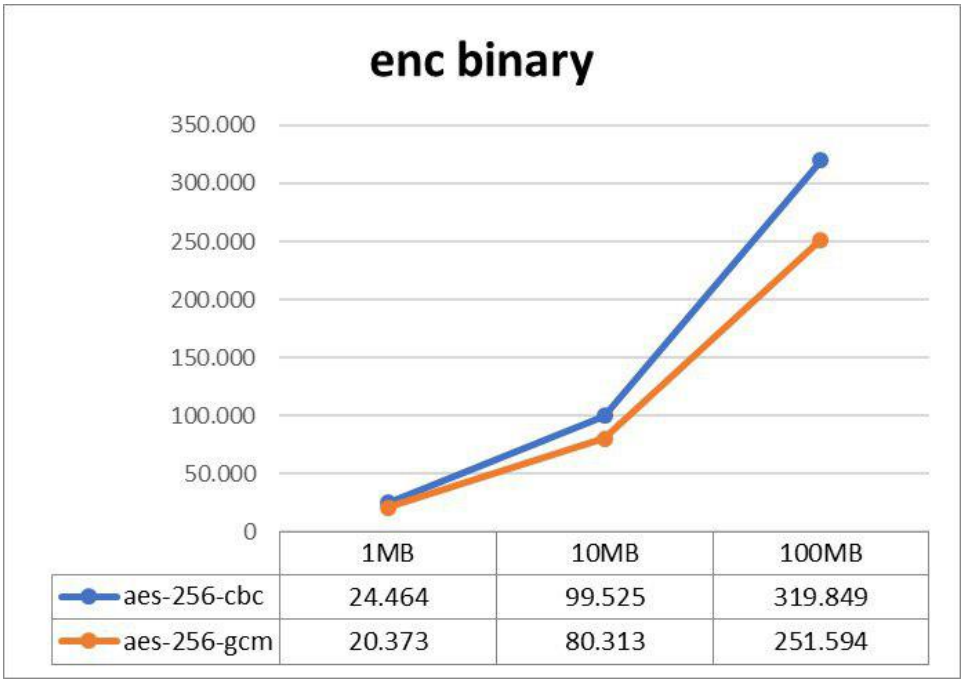


	4MB	8MB	16MB
aes-256-cbc	52.817	82.871	131.220
aes-256-gcm	41.973	71.297	139.899

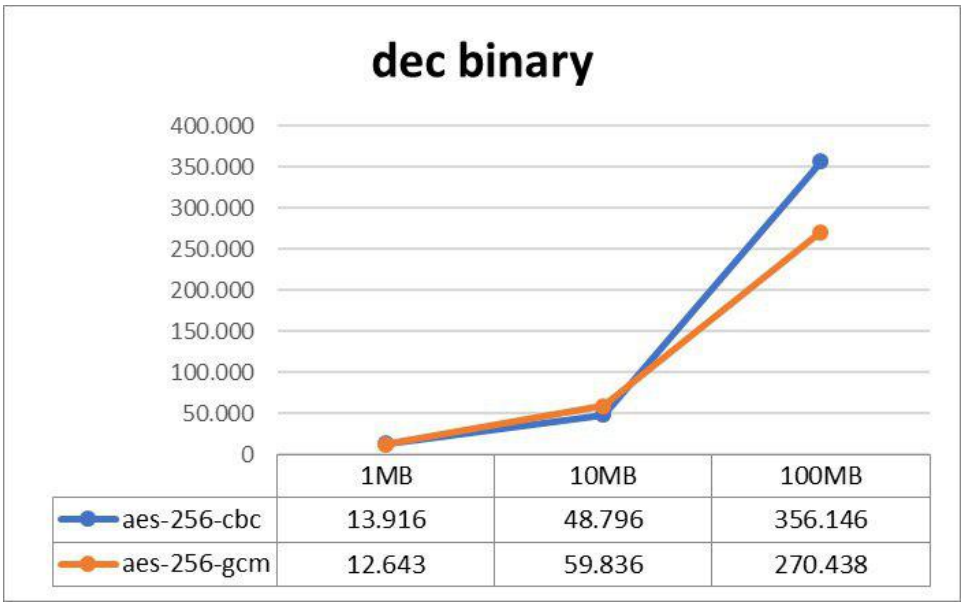


	4MB	8MB	16MB
aes-256-cbc	30.875	40.188	76.742
aes-256-gcm	25.349	44.569	83.791

GCM and CBC have almost the same behaviour on text files (GCM a bit slower for grater ones).



	1MB	10MB	100MB
aes-256-cbc	24.464	99.525	319.849
aes-256-gcm	20.373	80.313	251.594



	1MB	10MB	100MB
aes-256-cbc	13.916	48.796	356.146
aes-256-gcm	12.643	59.836	270.438

For binary files instead the greater is the file, the greater is the difference between GCM and CBC where GCM is much faster.