Understanding the Galois/Counter Mode (GCM)
HW3 - CNS Sapienza

Lombardo Andrea 1893440

21/11/2019

1 Introduction

Nowadays, the protection of transferring data is important to prevent the data hack easily. Advanced Encryption Standard with Galois Counter Mode (AES-GCM) plays an important role to provide high assurance of authenticity and data confidentiality in electronics, computers and other communication applications. How we know there's a plaintext, the clear text which is transformed to different information that we can call a ciphertext. This changing is done by an algorithm with a particular length key and cipher mode. In this report we're focus analyze the usage of Galois Counter Mode in comparison with cbc mode with a block algorithm (aes) with 256 as block size.

2 Galois/Counter Mode

Galois/Counter Mode (GCM) is a mode of operation for symmetric-key cryptographic block ciphers which ,using hardware resource, provides data integrity and confidentiality. Its performances (throughput rates, high-speed communication channels) are really good.

3 Design

I decide to implement this testing comparation using the two C files which let me to atomize the jobs:

- **Create-binary-file-hw3-1893440.c**: the following file is used to develop a program which helps me to create a binary file with a random content. After computing the compilation with gcc I should give in input the pair name and size I would like of the file.
  Implementation:
  ./Create-binary-file <file-name><size-of-file>

- **Encrypt-hw3-1893440.c** the following file is used to develop a program which helps me to run a encryption and decryption of the same files giving in input . The main goal is to analyse performance of pair of Operative Modes(CBC-GCM). After computing the compilation with gcc I should give in input the following data.

  Implementation:
  ./Encrypt <file-input> <file-ouput>
  When we run the following C file, it's waiting a message of stdin after printing in stdout
  ("Insert key through you can make testing:").
  Now the content tapped by you is reversed from stdin stream into a

char * variable through **fgets** routine. This avoid to write more time same password and let me to construct the string for running the bash command using **system** call. Then the file C makes testing according the following sequence of commands:

- system(**openssl enc -aes-256-cbc -k "password-inserted" -pbkdf2 -e -in argv[1] -out argv[2]**)
- system(**openssl enc -aes-256-cbc -k "password-inserted" -pbkdf2 -d -in argv[2] -out argv[1]**)
- system(**openssl enc -aes-256-gcm -k "password-inserted" -pbkdf2 -e -in argv[1] -out argv[2]**)
- system(**openssl enc -aes-256-gcm -k "password-inserted" -pbkdf2 -d -in argv[2] -out argv[1]**)

We know that for analyzing performance we should describe the time.

3.1 Time

The <sys/time.h> header defines the timeval structure that includes at least the following members:

time_t tv_sec seconds

suseconds_t tv_usec microseconds

So after defining two variables tv1 and tv2 of timeval structure for each system operation that I should analyze, so I need 8 variables of timeval structure, I decide to set them according **gettimeoftheday(*timeval,NULL)** which obtains the current time, expressed as seconds and microseconds since 00:0 and set the variable in input with the new time evaluated. So according this script I define a speed function and according the content of the string we refer to encryption or decryption function.

```
gettimeofday(&tv1,NULL);

system(string);

gettimeofday(&tv2,NULL);

total_time = ((double)(tv2.tv_usec -tv1.tv_usec) /1000000 +(double)(tv2.tv_sec - tv1.t
```

The total_time variable evaluated is in **microsec**.
After evaluating the two total time I manage to evaluate the speed ratio:

The speed ratio= $\frac{average-speed-of-encryption}{average-speed-of-dencryption}$

4 Comparison with CBC mode

In this report we should test difference of these 2 Modes(CBC,GCM) with a same algorithm. We can analyze that AES-GCM is more secure chiper according to the fact CBC operates by XOR'ing each block with the previous one and without a parallel writing. CBC is also vulnerable to a padding oracle attack which is an attack that uses the padding validation of a cryptographic message to decrypt the ciphertext in detail whether the padding of an encrypted message is correct or not without knowing the encryption key, we're able to discover data through the oracle using the oracle's key.
GCM works in parallel processing due it's implemented with an instruction pipeline or a hardware pipeline. In contrast, the cipher block chaining (CBC) mode of operation incurs significant pipeline stalls that hamper its efficiency and performance.

5 Report testing

Here we can see two tables where I've proved the performances differences according different binary file where the content is different for size reason and the same kind of test with pictures file.

Next report is focus on binary file of different size(from 10kb to 100MB).

Speed Encryption	Speed Decryption	Ratio Speed	File-size	modes	kind-of-file
7.946364	3.858094	2.059661	10kB	cbc	binary
5.183286	1.804799	2.871946	100kB	cbc	binary
3.211328	0.937233	3.426392	1MB	cbc	binary
2.705189	0.686286	3.941781	10MB	cbc	binary
3.330719	1.412433	2.358143	100MB	cbc	binary
5.031048	2.298747	2.188604	10kB	gcm	binary
3.390072	2.020719	1.677656	100kB	gcm	binary
1.877212	2.835747	0.661981	1MB	gcm	binary
1.360843	1.566537	0.868695	10MB	gcm	binary
2.342711	2.354113	0.995157	100MB	gcm	binary

Next report is focus on picture files of different size downloaded here **https://sample-videos.com/download-sample-jpg-image.php**

Speed Encryption	Speed Decryption	Ratio Speed	File-size	modes	kind-of-file
2.932857	1.019337	2.877220	100kB	cbc	picture
2.766678	0.831063	3.329083	1MB	cbc	picture
2.206279	0.896675	2.460511	10MB	cbc	picture
2.124189	1.842140	1.153109	100kB	gcm	picture
1.488929	1.159010	1.284656	1MB	gcm	picture
1.517238	1.026948	1.477424	10MB	gcm	picture

6 Conclusion

In conclusion we can see how we can see in the report tables the speeds encrytion of gcm works better than speeds of cbc but we couldn't say the same thing to the decryption function. In general performances with my test set are better with a picture file.

CBC does allow random access and limit parallelization for decryption, for this point is no worse than GCM. Instead with GCM you cannot authenticate a partial message, if you want authenticate the entire message

References

[1] https://www.privateinternetaccess.com/helpdesk/kb/articles/what-s-the-difference-

[2] https://linuxg.net/how-to-install-libressl-2-1-6-on-linux-systems/

[3] https://www.openssl.org/docs/man1.1.1/man1/enc.html

[4] http://man7.org/linux/man-pages/man2/gettimeofday.2.html

[5] https://pubs.opengroup.org/onlinepubs/7908799/xsh/systime.h.html

[6] https://en.wikipedia.org/wiki/Galois/Counter_Mode

[7] https://iopscience.iop.org/article/10.1088/1742-6596/1019/1/012008/pdf