Lab2 (Report)

VATSAL AGRAWAL

Master Of Technology

COMPUTER SCIENCE and ENGINEERING

2nd Sem

Entry Number - 2021MCS2157

NETWORK & SYSTEM SECURITY

SIL 765

Introduction

My program is working correctly for all cases. The screenshot of working is in the end. I have used library pycryptodome, which is a forked version of popular pycrypto, and found this library by seeing various sites. I have used pycryptodome over the cryptography library because of its speed and simplicity, and functionality. Pycrypto was found to have some security problems and update issues; thus, choose pycryptodome over it. I have solved the rest of my program by understanding and seeing its documentation at

https://pycryptodome.readthedocs.io/en/latest/src/introduction.html

All implementations are quite very fast and work for a good length of input except RSA-based encryption. A detailed description is given below with a comparison summary. Gradescope submission is also running. I have submitted the same plaintext file as provided and also verified the answer for the same. Except for plaintext and RSA and ECC keys and auth_tag_valid, all data are encoded in base64 for printing.

INDEX

1. Observation
2. Analysis
3. Encryption & Decryption
    a. ALGORITHM
    b. Comparison Summary
4. Authentication & Verification
    a. ALGORITHM
    b. Comparison Summary
5. Encryption & Authentication
    a. ALGORITHM
    b. Comparison Summary
6. Pre-Requisite
    a. How To Make Executable and Run Programme
    b. Structure Of Programme
7. References

Observation

Plaintext Size = 5072 bits

Plaintext Size in base 64 encoding = 6784 bits (=5072+16/6*8)

Algorithm	**Key Length (in bits)**	**Execution Time (in ms)**	**Tag Size (in bits)/nonce size**	**Ciphertext / Plintext Size (in bits)**	**Total packet size needs to be transferred (in bits)**
AES-128-CBC-ENC	128 bits 192 bits(base64 actual stored)	0.201 ms	Nonce (IV) = 128 bits	6848 bits with padding (base64) (cipher)	6848 bits (cipher) (base64)
AES-128-CBC-DEC	128 bits 192 bits(base64 actual stored)	0.204 ms			
AES-128-CTR-ENC	128 bits 192 bits(base64 actual stored)	0.140 ms	Nonce = 64 bits	6784 bits (base64) (cipher)	6784 bits (cipher) (base64)
AES-128-CTR-DEC	128 bits 192 bits(base64 actual stored)	0.065 ms			
RSA-2048-ENC	2048 bits 3600 bits (pem actual stored) (Public Receiver)	3.598 ms		2752 bits (base64) (cipher)	2752 bits (cipher) (base64)
RSA-2048-DEC	2048 bits 13392 bits (pem actual stored) (Private Receiver)	56.898 ms			
AES-128-CMAC-GEN	128 bits 192 bits(base64 actual stored)	0.664 ms	192 bits (base64)	5072 bits 6784 bits (base 64) (plaintext)	5264 bits (plaintext+auth) 6976 bits (base64)
AES-128-CMAC-VRF	128 bits 192 bits(base64 actual stored)	0.266 ms			
SHA3-256-HMAC-GEN	128 bits 192 bits(base64 actual stored)	1.228 ms	352 bits (base64)	5072 bits 6784 bits (base 64) (plaintext)	5424 bits (plaintext+auth) 7136 bits (base64)
SHA3-256-HMAC-VRF	128 bits 192 bits(base64 actual stored)	0.145 ms			
RSA-2048-SHA3-256-SIG-GEN	2048 bits 13392 bits (pem actual stored) (Private Sender)	85.79 ms	2752 bits (base64)	5072 bits 6784 bits (base 64) (plaintext)	7824 bits (plaintext+auth) 9536 bits (base64)
RSA-2048-SHA3-256-SIG-VRF	2048 bits 3600 bits (pem actual stored) (Public Sender)	2.897 ms			
ECDSA-256-SHA3-256-SIG-GEN	256 bits 1920 bits (pem actual stored) (Private Sender)	6.657 ms	704 bits (base64)	5072 bits 6784 bits (base 64) (plaintext)	5776 bits (plaintext+auth) 7488 bits (base64)
ECDSA-256-SHA3-256-SIG-VRF	256 bits 1416 bits (pem actual stored) (Public Sender)	5.978 ms			
AES-128-GCM-GEN	128 bits 192 bits(base64 actual stored)	0.564 ms	Tag = 192 bits (base64); Nonce = 96 bits	6784 bits (base 64) (cipher)	6976 bits (cipher+auth) (base64)
AES-128-GCM-VRF	128 bits 192 bits(base64 actual stored)	0.315 ms			

Table1. Observation result

Note –

Assuming only plaintext/ciphertext and auth tag is transferred as packet.

All Cipheretxt, Auth. Tag are in base 64 encodings and transferred in the same way; In Authentication, the plaintext is transferred with Auth_Tag

Where Plaintext is transferred, its size in base 64 is also mentioned so to make comparison easier. Also, encoding is mentioned for that length, if any done.

All data recordings are taken from the average of 5 readings

Actual Stored Bits of the key are the length of the key stored in memory in base 64 encodings. RSA and ECC keys are in pem format.

The bar graph is given below for all base64 encoding value

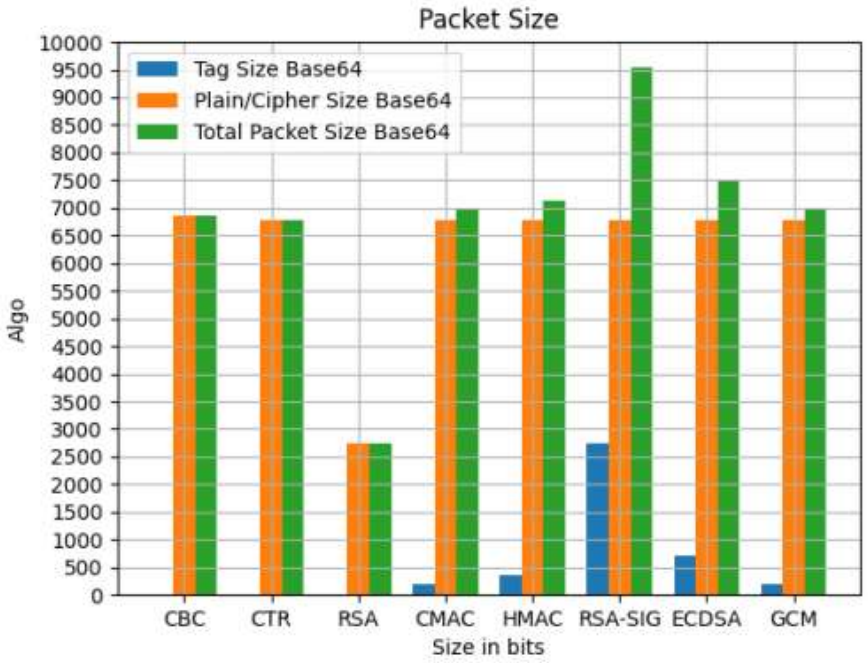

Fig1. Packet Size vs. Algo

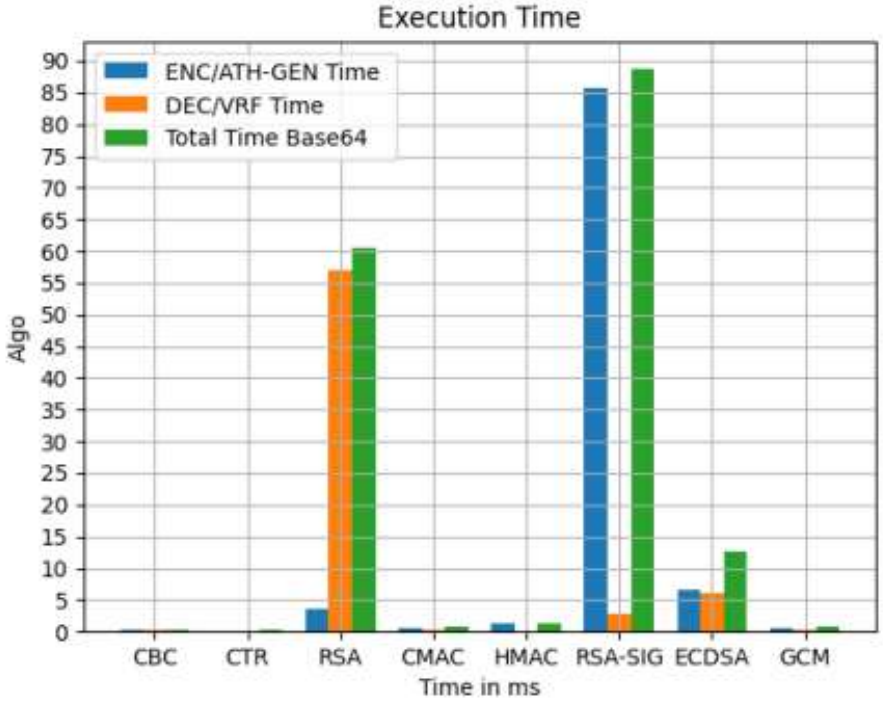

Fig2. Execution Time vs. Algo

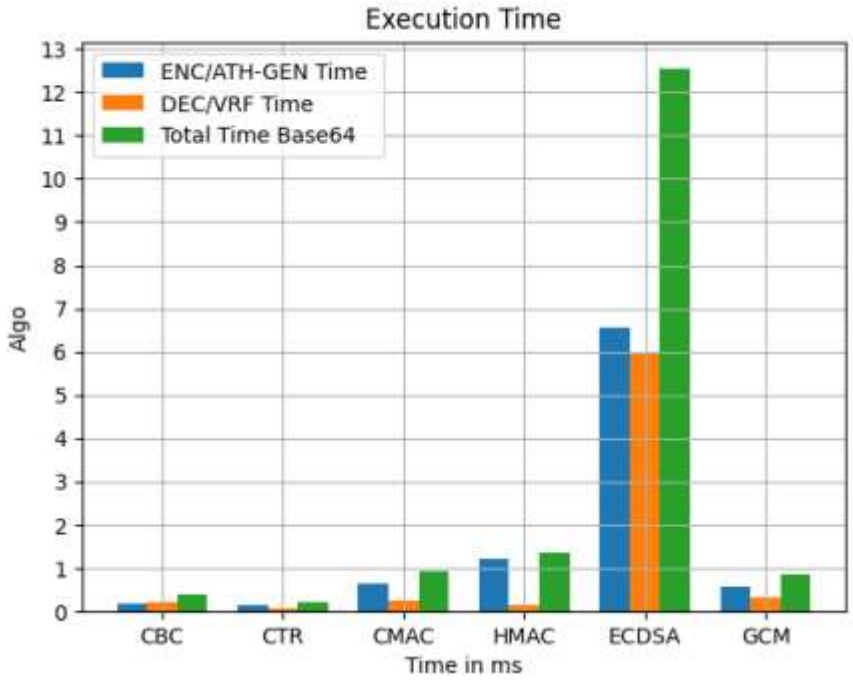

Fig3. Execution Time vs. Algo (without RSA)

Analysis

1. We can clearly see that the key size for RSA is very big, followed by ECDSA and the rest having the same size of 128 bit
2. We can also Clearly see that time RSA Encryption and Decryption is very high, followed by GCM, then CBC, and then CTR
3. Also, RSA ENC time is less than DEC, also for CBC
4. CTR and GCM DEC time is also less than ENC
5. We can also see that plain and cipher size is the same in GCM and CTR.
6. Also, Cipher size is large in RSA ENC than plain. Also, padding is included in CBC
7. RSA Authentication time is very high, followed by ECDSA and the rest of all
8. RSA Sign verification time is very less as compared to its GEN
9. For Authentication, Verification time is less for all than GEN
10. RSA Auth Tag size is very large, followed by ECDSA, then HMAC and then GCM and CMAC
11. RSA proves to be very costly algo in size and time both, as it is asymmetric, followed by ECDSA and then rest

Encryption & Decryption

Encrypt allows us to provide confidentiality to messages we are sending over the channel, which may be exposed to an attacker. Decryption allows us to decrypt the ciphertext to plaintext.

AES-128-CBC-ENC & DEC

This stands for Advance Encryption standard using 128 bit symmetric key with Cipher Bloch Chaining Mode. It is a type of block cipher. In this, each block of ciphertext is Exored with plaintext and then encrypted. I have used nonce as iv in this.

Cons:-

1) Padding is needed in this.

2) Serialized computation of ciphers block

3) one Block loss leads to whole data loss

4) Addition iv need to be secured

Pros:-

1) No pattern can be seen

2) Parallel computation for decryption.

AES-128-CTR-ENC & DEC

This stands for Advance Encryption standard using a 128-bit symmetric key with Counter Mode. It is a type of block cipher that turns into a stream cipher. I have used an 8-byte nonce to generate a counter block with 8 byte counter. In this, the counter block is encrypted and XORed with plain for cipher.

Cons:-

1) Counter may overflow if the nonce is large for large message

2) Size of cipher is same as plain

3) Counter value must be the same on both sides.

Pros:-

1) It is faster than CBC

2) ENC and DEC both can be done in parallel, and preprocessing of block can also be done

3) No padding is also required

4) No relation between blocks; loss of one block do not affect other

RSA-2048-ENC & DEC

This stands for Rivest, Shamir, Adleman using 2048 bit Asymmetric key for encryption and decryption. I have used the symmetric input key as its input. It is based on the factorization problem. The public key of the receiver encodes data, and the private key decodes it. I have used RSAES-OAEP; OAEP is padding here

Cons:-

1) High Computation needed to encrypt and decrypt data

2) Quantum computer can break this easily

3) Cannot deploy for large data due to high computation

4) Third-party verify the public key

Pros:-

1) Hard to crack and more secure than cipher block

2) No preshared key is needed

3) Can be used to encrypt session keys

Comparison Summary

As discussed, none of the above algo provides authentication and integrity, RSA is slower than CBC is slower than CTR, Padding is needed in CBC while it is not needed in CTR, RSA is more secure than rest 2. Loss of complete data in case of CBC while no such case in CTR. Symmetric key, IV, and counter need to be preshared in symmetric while not in RSA.

Authentication & Verification

An authentication tag is needed to ensure that data is not changed/ modified while sending out through the channel and originate from an authentic sender using a preshared key. It may also provide non-repudiation in asymmetric keys.

AES-128-CMAC-GEN & VRF

This stands for AES Cipher-based Message Authentication Code using a 128-bit symmetric key. It is the same as CBC MAC one key.

Cons:-

1) CMC slower than HMAC

2) Not provide non repudation

3) Need of preshared key

Pros:-

1) It is based upon the universal hash

2) It is parallelizable

SHA3-256-HMAC-GEN & VRF

This stands for 256-bit output SHA-3 Hashed-based Message Authentication Code. It uses a crypto hash on plaintext for authentication. It uses the hash function two times.

Cons:-

1) Need of preshared key

2) Not provide non repudation

3) only one sender and receiver

Pros:-

1) It is very fast as it is based upon hashing

2) Length of the digest is small

RSA-2048-SHA3-256-SIG-GEN & VRF

This stands for 256-bit output SHA-3 Hashed-based signature generation using RSA with 2048 bit key. Here digest is created using hash and encrypted using senders private key and decrypted using senders public key. I have used RSASSA-PKCS1-v1_5.

Cons:-

1) Third-party needed for key distribution

2) Computational cost is relatively higher

3) RSA is slower at generation

Pros:-

1) Provide Non Repudation

2) Encorpates feature of sha3 hashing

3) RSA is faster at Verification

ECDSA-256-SHA3-256-SIG-GEN & VRF

This stands for 256-bit output SHA-3 Hashed-based signature generation using RSA with 2048 bit key. Here digest is created using hash and security based on discrete logarithm problem in elliptic curves field. I have used RSASSA-PKCS1-v1_5.

Cons:-

1) Third-party needed for key distribution

2) Computational cost is relatively higher than others

3) Complex method

Pros:-

1) It is faster than RSA

2) uses smaller key

3) Requires less computing than RSA

4) Comparatively more secure

Comparison Summary

In all, as seen asymmetric way is more computationally harder and takes more time but provide more security, ECDSA is not well established but found to be better in today's world of brute force attack as compared to RSA also, its less key size also makes it preferable for the small storage system. CMAC approach is faster than these but slower than HMAC. CMAC can get faster with hardware support. HMAC does not provide parallelization, but CMAC can.

Encryption & Authentication

AES-128-GCM-GEN & VRF

This stands for Advance Encryption standard using 128 bit symmetric key with Golis Counter Mode. It uses CTR mode for its encryption and uses Galois field multiplication polynomial MAC

Cons:-

1) the Same nonce leads to a security breach

2) lookup tables vulnerable to cache timing attacks

3) Vulnerable to cycling attack

Pros:-

1) Lower encryption overheads

2) Do both encryption and authentication

3) Processed in parallel

4) Save from CPA and CCA (chosen ciphertext attack)

Comparison Summary

AES GCM is based on a single cipher, and it is faster, while CTR and HMAC are slower due to non-parallization. As actions are done in parallel, it results in better of all when the combined time of encryption and authentication is taken. But sometimes, it is vulnerable to cycling attacks. Also, performance comes with the threat of cache timing attacks.

Pre-Requisite

How To Make Executable and Run Programme

1) We can use the **make** and **make all** command to see our program running
2) You can also run **python example_test.py**

Structure Of Programme

Program is made up of directory structure which includes

1) example_test.py – code to call function, print time, print length of return parameter and their types
2) execute_crypto.py– code for ALGO
3) Makefile – Makefile command
4) original_plaintext.py – Plaintext
5) Data – folder containing output, plaintext, figures
6) setup_env.sh – To install libraries
7) output.txt – sample output
8) readme.pdf – containing documentation

References

I have completed my assignment by learning from the following sources

1. Pycryptodome documentation
2. Several Sites related to Working of Algorithm and their advantage and disadvantage