Course: SIL765 - Networks and Systems Security

Assignment 2 : Evaluating Cryptographic Primitives

Submitted by: **Abhishek Singh (2022JCS2665)**

Email: jcs222665@iitd.ac.in

MTech (Cybersecurity)

Problem Statement:

The aim of this assignment is to evaluate and compare the computational, communication

and storage cost of various cryptographic algorithms for encryption and authentication.

Solution:

Library Used:

I have used following library mentioned below:

1. Cryptography: It includes both high level recipes and low level interfaces to common cryptographic algorithms such as symmetric ciphers, message digests, and key derivation functions.

2. Pycryptodome: It is a self-contained Python library that provides cryptographic functions and algorithms. It supports a wide range of cryptographic operations, including symmetric and asymmetric encryption, digital signatures, hash functions, and more. The library is a fork of the PyCrypto library and aims to provide a more secure and up-to-date implementation of cryptographic algorithms.

3. Rsa: RSA is a widely used algorithm for public-key cryptography. It is based on the mathematical concept of prime factorization and can be used for encryption, digital signatures, and key exchange.

Pros and Cons of Algorithm:

AES-128 in CBC Mode:

Pros:

1. AES-128 in CBC mode (Cipher Block Chaining) is a symmetric encryption algorithm that uses the Advanced Encryption Standard (AES) algorithm with a 128-bit key length and the CBC mode of operation.

2. In CBC mode, each block of plaintext is XORed with the previous ciphertext block before being encrypted with the AES algorithm. This provides an added layer of security as each block of ciphertext depends on all previous blocks, making it more difficult for an attacker to modify encrypted data without being detected.

3. AES-128 is widely used due to its security and efficiency, and is considered a strong encryption algorithm. It is widely used in various applications, including secure communication, file encryption, and data protection.

Cons:

1. Dependence on Initialization Vector (IV): In CBC mode, each block of the plaintext message is XORed with the previous ciphertext block before being encrypted. This means that the same plaintext message encrypted with the same key will produce different ciphertexts if the IV is different. The IV must be unique for each encryption operation and should not be reused, as otherwise, it can lead to security weaknesses.

2. Padding Oracle Attacks: When using CBC mode with a block cipher such as AES, the padding of the plaintext message must be carefully managed. Padding oracle attacks exploit the fact that the padding of a message can be checked even if the plaintext message is not known, allowing an attacker to recover the plaintext message through a series of queries.

3. Slower performance: CBC mode requires each block of the plaintext message to be processed in sequence, which can be slower compared to other modes of operation, such as CTR (Counter) mode.

AES-128 in CTR Mode:

Pros:

1. AES-128 in CTR mode (Counter mode) is a symmetric encryption algorithm that uses the Advanced Encryption Standard (AES) algorithm with a 128-bit key length and the CTR mode of operation.

2. In CTR mode, a counter is used as the input to the AES encryption algorithm, and each block of plaintext is encrypted with a unique encryption key. This provides a parallelizable encryption operation, allowing for high-speed encryption and decryption. CTR mode also provides confidentiality and authenticity without the need for a message authentication code (MAC) like CBC mode.

3. Like AES-128 in CBC mode, AES-128 in CTR mode is considered a strong encryption algorithm and is widely used in various applications, including secure communication, file encryption, and data protection.

Cons:

1. Counter synchronization: In CTR mode, the encryption of each block of the plaintext message depends on the current value of a counter. It is important to ensure that the

counter value is synchronized between the sender and receiver, as otherwise, the decryption of the ciphertext message may fail.

2. Nonce reuse: The counter in CTR mode is usually combined with a nonce (number used once) value, which is used to ensure that each encryption operation uses a unique counter. Reusing the same nonce value with the same key can lead to security weaknesses, as it can allow an attacker to recover the plaintext message through a technique known as a two-time pad attack.

3. Vulnerability to block-level tampering: In CTR mode, an attacker can modify a ciphertext block without being detected, as the modification will only affect the decrypted plaintext for the same block. This can lead to message tampering and data integrity issues.

RSA with 2048 bit key:

Pros:

1. RSA with a 2048-bit key is a widely used public-key cryptography algorithm that uses the mathematical concept of prime factorization. The key length of 2048 bits is considered secure for most applications, providing strong protection against brute-force attacks.

2. In RSA, a public key and a private key are generated. The public key can be freely shared and used for encrypting messages or verifying digital signatures, while the private key must be kept confidential and used for decrypting messages and signing digital signatures.

3. RSA with a 2048-bit key is commonly used for secure communication, such as in SSL/TLS connections, digital signatures, and encryption of sensitive data.

Cons:

1. Slower performance: RSA is a computationally intensive algorithm and can be slow compared to symmetric encryption algorithms, such as AES. This can be especially problematic for encryption of large amounts of data.

2. Key size limitations: A 2048-bit RSA key is considered to be secure for most practical purposes. However, with the advancement of quantum computing, it is possible that RSA with 2048-bit keys may become vulnerable in the future.

3. Security limitations: RSA is vulnerable to attacks such as Factoring, Wiener's attack, Common Modulus attack, etc. In some cases, it may be possible for an attacker to recover the private key and use it to decrypt the encrypted message or forge a digital signature.

AES with 128-bit key in CMAC Mode:

Pros:

1. AES with a 128-bit key in CMAC mode (Cipher-based Message Authentication Code) is a symmetric encryption algorithm that uses the Advanced Encryption Standard (AES)

algorithm with a 128-bit key length and the CMAC mode of operation.

2. In CMAC mode, the AES encryption algorithm is used to compute a message authentication code (MAC) that is appended to the encrypted message. The recipient of the message can then use the MAC to verify the authenticity and integrity of the message.

3. AES with a 128-bit key in CMAC mode is widely used for secure communication, as it provides both confidentiality and authenticity for the message. The 128-bit key length is considered secure for most applications, providing strong protection against brute-force attacks.

Cons:

1. Vulnerability to key reuse: Like all symmetric encryption algorithms, AES in CMAC mode is vulnerable to key reuse, which can lead to security weaknesses if the same key is used for multiple encryption operations.

2. Key size limitations: AES with a 128-bit key is considered secure for most practical purposes. However, with the advancement of quantum computing, it is possible that AES with 128-bit keys may become vulnerable in the future.

3. Key management challenges: Proper key management is critical for the security of AES in CMAC mode. This includes generating strong keys, securely distributing the keys, and securely storing the keys. Improper key management can lead to security weaknesses.

SHA 3 with 256-bit key in HMAC Mode:

Pros:

1. SHA-3 with a 256-bit key in HMAC mode (Keyed-Hash Message Authentication Code) is a cryptographic hash function that uses the Secure Hash Algorithm 3 (SHA-3) algorithm with a 256-bit key and the HMAC mode of operation.

2. In HMAC mode, the SHA-3 hash function is combined with a secret key to create a MAC that can be used to verify the authenticity and integrity of the message. The recipient of the message can use the same key to verify the MAC and ensure that the message has not been tampered with during transmission.

3. SHA-3 with a 256-bit key in HMAC mode is widely used for secure communication and data protection, as it provides authenticity and integrity for the message without the need for encryption. The 256-bit key length is considered secure for most applications, providing strong protection against brute-force attacks.

Cons:

1. Vulnerability to hash collisions: SHA-3, like all hash functions, is vulnerable to hash collisions, which can occur when two different inputs result in the same hash output. This can lead to security weaknesses if an attacker is able to find a hash collision and use it to forge a

message.

2. Key size limitations: SHA-3 with a 256-bit key is considered secure for most practical purposes. However, with the advancement of quantum computing, it is possible that SHA-3 with 256-bit keys may become vulnerable in the future.

RSA with a 2048-bit key and SHA3 with a 256-bit output:

Pros:

1. RSA is a public-key cryptography algorithm that uses the mathematical concept of prime factorization to generate a public key and a private key. The public key can be used for encrypting messages or verifying digital signatures, while the private key is used for decrypting messages and signing digital signatures. RSA with a 2048-bit key is considered secure for most applications, providing strong protection against brute-force attacks.

2. SHA-3 is a cryptographic hash function that generates a fixed-length output, also known as a message digest, from an input message. SHA-3 with a 256-bit output generates a 256-bit message digest that can be used for message authentication or integrity checks.

3. Together, RSA with a 2048-bit key and SHA-3 with a 256-bit output provide strong protection for secure communication, digital signatures, and data protection. The RSA algorithm provides confidentiality and authenticity, while the SHA-3 algorithm provides message integrity.

Cons:

1. Performance limitations: RSA is computationally intensive and can be slow for encryption and decryption of large amounts of data. In addition, SHA-3 with a 256-bit output is computationally intensive, which may affect its performance for hash computation of large amounts of data.

2. Cryptographic agility: Cryptographic algorithms should be designed with agility in mind so that they can be easily updated when security weaknesses are discovered. RSA with 2048-bit key and SHA-3 with 256-bit output may not be easily updated, which may affect their long-term security.

Elliptic Curve Digital Signature Algorithm (ECDSA) with a 256-bit key and SHA3 with a 256-bit output:

Pros:

1. ECDSA is a public-key cryptography algorithm that uses elliptic curve mathematics to generate a public key and a private key. The public key can be used for encrypting messages or verifying digital signatures, while the private key is used for decrypting messages and signing digital signatures. ECDSA with a 256-bit key is considered secure for most applications, providing strong protection against brute-force attacks.

2. SHA-3 is a cryptographic hash function that generates a fixed-length output, also known as a message digest, from an input message. SHA-3 with a 256-bit output generates a 256-bit message digest that can be used for message authentication or integrity checks.

3. Together, ECDSA with a 256-bit key and SHA-3 with a 256-bit output provide strong protection for secure communication, digital signatures, and data protection. The ECDSA algorithm provides confidentiality and authenticity, while the SHA-3 algorithm provides message integrity.

Cons:

1. Performance limitations: ECDSA is computationally intensive and can be slower than other digital signature algorithms such as RSA.

2. Side Channel Attack: may target the implementation of the ECDSA algorithm, and attempt to extract secret information, such as the private key, by observing side-channel information, such as the time taken to perform operations, power consumption, or electromagnetic emissions.

AES with a 128-bit key in the GCM mode:

Pros:

1. AES with a 128-bit key in the Galois/Counter Mode (GCM) is a symmetric encryption algorithm that uses the Advanced Encryption Standard (AES) algorithm with a 128-bit key and the GCM mode of operation.

2. In GCM mode, AES provides both confidentiality and authenticity for the encrypted message. The GCM mode uses a counter and a unique initialization vector (IV) to ensure that each encryption produces a unique ciphertext. The ciphertext also includes an authentication tag, which is used to verify the authenticity of the message upon decryption.

3. AES with a 128-bit key in GCM mode is widely used for secure communication and data protection, as it provides both confidentiality and authenticity in a single operation. The 128-bit key length is considered secure for most applications, providing strong protection against brute-force attacks.

Cons:

1. Limited Nonce Space: AES-GCM uses a nonce, which is a unique value used during encryption. The size of the nonce is limited to 64 bits, which can result in a risk of nonce reuse if not managed carefully.

2. Timing Vulnerabilities: AES-GCM is vulnerable to timing attacks, which can reveal information about the encrypted data or the encryption key by observing the time it takes to perform certain operations.

3. Confidentiality Leak: AES-GCM can leak information about the encrypted data if an attacker is able to modify the encrypted data in a way that changes its length.

Results:

Algorithm	Key Length	Execution Time	Packet Length
AES-128-CBC-ENC	128	3.484	5120
AES-128-CBC-DEC	128	0.136	5120
AES-128-CTR-ENC	128	0.343	5064
AES-128-CTR-DEC	128	0.118	5064
RSA-2048-ENC	2048	0.246	2048
RSA-2048-DEC	2048	8.792	2048
AES-128-CMAC-GEN	128	1.038	5192
AES-128-CMAC-VRF	128	0.262	5192
SHA3-256-HMAC-GEN	128	0.665	5320
SHA3-256-HMAC-VRF	128	0.160	5320
RSA-2048-SHA3-256-SIG-GEN	2048	27.699	7112
RSA-2048-SHA3-256-SIG-VRF	2048	0.512	7112
ECDSA-256-SHA3-256-SIG-GEN	256	1.582	5888
ECDSA-256-SHA3-256-SIG-VRF	256	0.211	5888
AES-128-GCM-GEN	128	0.509	5192
AES-128-GCM-VRF	128	0.262	5192

Analysis of Results:

1. The RSA algorithm has the slowest encryption performance and the highest computational cost.

2. In contrast, AES and hashing-based algorithms are faster for authentication compared to RSA and ECDSA, which have longer keys and execution times.

3. The key length of RSA-based algorithms is the longest, making storage more expensive.

4. However, RSA-based algorithms have shorter packet lengths, reducing communication costs.

5. Overall, AES-based algorithms are more efficient in terms of storage, computational, and communication costs.

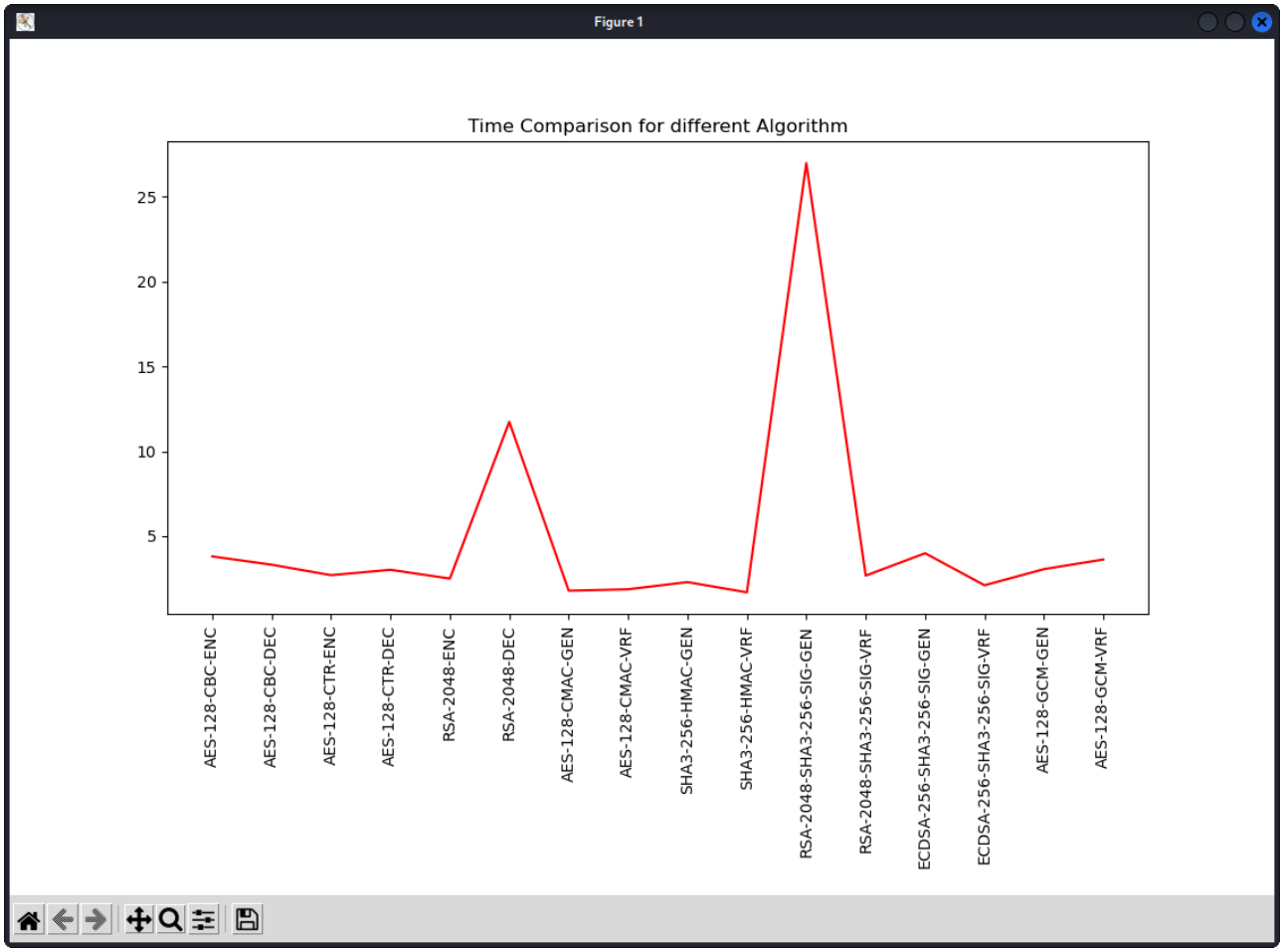


Figure 1
Time Comparison for different Algorithm
25
20
15
10
5
AES-128-CBC-ENC
AES-128-CBC-DEC
AES-128-CTR-ENC
AES-128-CTR-DEC
RSA-2048-ENC
RSA-2048-DEC
AES-128-CMAC-GEN
AES-128-CMAC-VRF
SHA3-256-HMAC-GEN
SHA3-256-HMAC-VRF
RSA-2048-SHA3-256-SIG-GEN
RSA-2048-SHA3-256-SIG-VRF
ECDSA-256-SHA3-256-SIG-GEN
ECDSA-256-SHA3-256-SIG-VRF
AES-128-GCM-GEN
AES-128-GCM-VRF

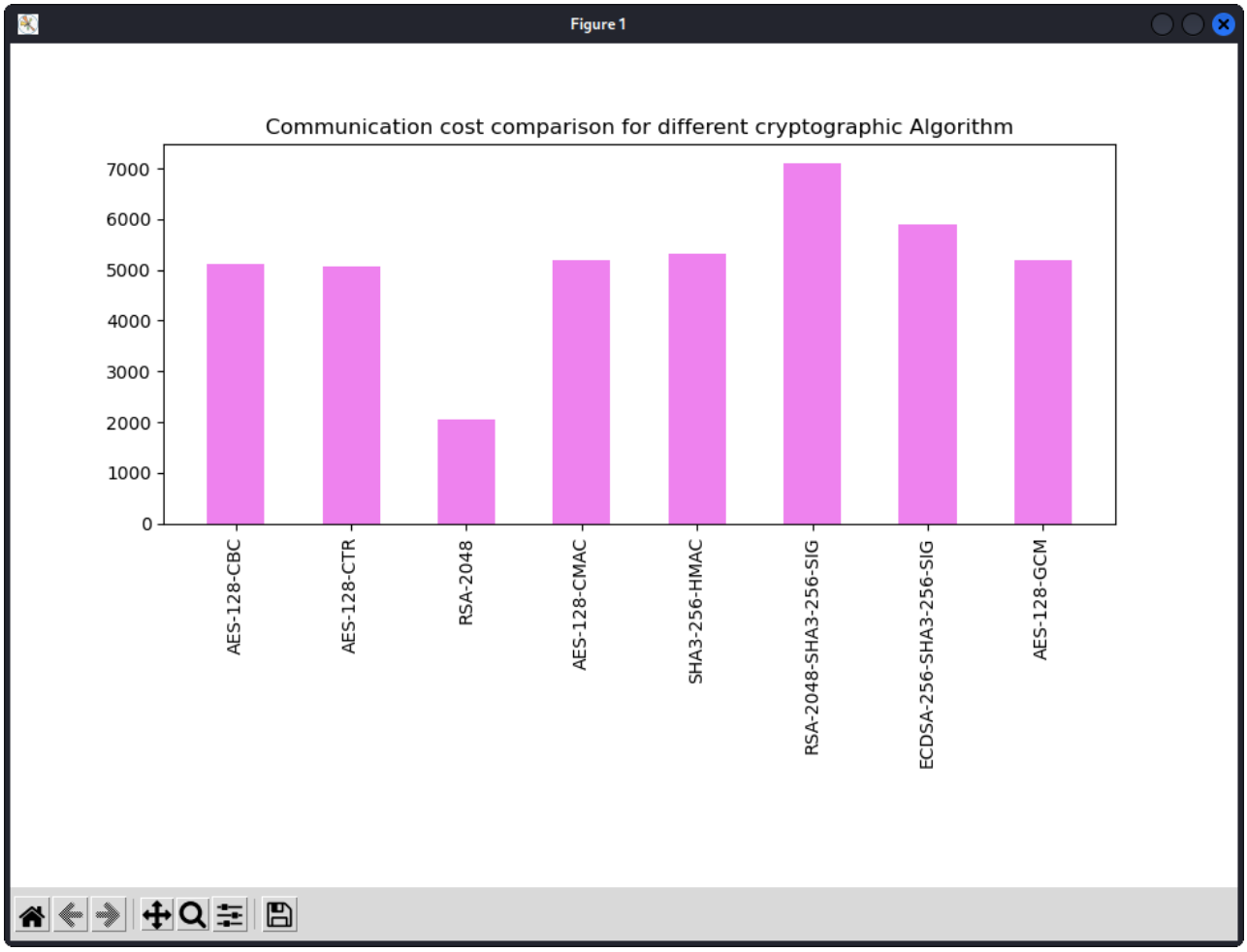
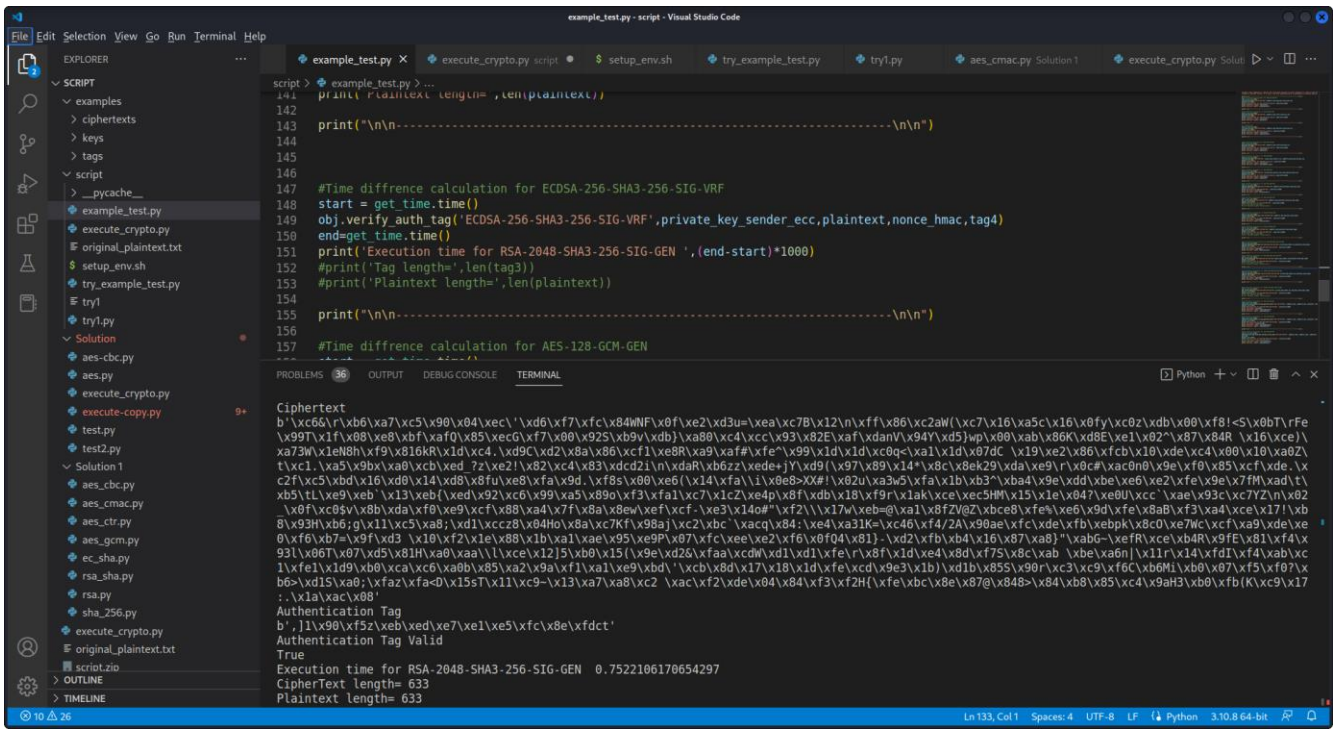


Figure 1
Communication cost comparison for different cryptographic Algorithm
7000
6000
5000
4000
3000
2000
1000
0
AES-128-CBC
AES-128-CTR
RSA-2048
AES-128-CMAC
SHA3-256-HMAC
RSA-2048-SHA3-256-SIG
ECDSA-256-SHA3-256-SIG
AES-128-GCM

Output:


example_test.py - script - Visual Studio Code
print("\n\n----------\n\n")
#Time diffrence calculation for ECDSA-256-SHA3-256-SIG-VRF
start = get_time.time()
obj.verify_auth_tag('ECDSA-256-SHA3-256-SIG-VRF',private_key_sender_ecc,plaintext,nonce_hmac,tag4)
end=get_time.time()
print('Execution time for RSA-2048-SHA3-256-SIG-GEN ',(end-start)*1000)
#print('Tag length=',len(tag3))
#print('Plaintext length=',len(plaintext))
print("\n\n----------\n\n")
#Time diffrence calculation for AES-128-GCM-GEN
Ciphertext
Authentication Tag
Authentication Tag Valid
True
Execution time for RSA-2048-SHA3-256-SIG-GEN 0.7522106170654297
CipherText length= 633
Plaintext length= 633

References:

Network Security Essentials, William Stallings

https://en.wikipedia.org/wiki/One-key_MAC

https://medium.com/asecuritysite-when-bob-met-alice/i-know-hmac-but-whats-cmac-b859799af732

https://en.wikipedia.org/wiki/Block_cipher_mode_of_operation

https://en.wikipedia.org/wiki/RSA_(cryptosystem)

https://en.wikipedia.org/wiki/SHA-3

https://en.wikipedia.org/wiki/HMAC

https://en.wikipedia.org/wiki/Elliptic_Curve_Digital_Signature_Algorithm

https://www.encryptionconsulting.com/education-center/what-is-ecdsa/