Cryptographic Protocol Evaluation and Implementation

Kaushik Kumar TRS (50471138)
kthogulu@buffalo.edu
University at Buffalo

1. **Chosen Programming Language:** Python 3.9.12

   Libraries Utilized: The main library used for our experiment is **Pycryptodome,** and below are the functions utilized for various cryptographic functions

   - **AES** - *Crypto.Cipher.AES*
   - **RSA** - *Crypto.Cipher.PKCS1_OAEP, Crypto.Cipher.RSA*
   - **Hashing** - *Crypto.Hash import SHA256, SHA512, SHA3_256*
   - **Digital Signature** - *Crypto.Signature.DSS, Crypto.Signature.DSA*

2. **Algorithm Timings:**

   Key Generation:

Algorithm	Key Size	Key Generation Time (Milliseconds)
AES	128	0.003
AES	256	0.003
RSA	2048	407.677
RSA	3072	820.441
DSA	2048	731.373
DSA	3072	4954.727

   Observation:

   AES: The key generation time for AES 128 bits and 256 bits did not have much difference
   RSA: The key generation time for 3072-bit is significantly higher than generating a 2048 bit key
   DSA: The key generation time for 3072-bit key was significantly large when compared to that of generating a 2048-bit key, which might be because of the library and system-dependent hardware level configuration.
   All the presented key generation times were observed multiple times and every time there was a slight variation in the time taken to generate the key which is because of the pseudo randomness.

Encryption Algorithms:

Algorithm	File Size & Key Size (bits)	Encryption Time (millisecond)	Encryption Speed (milliseconds/byte)	Decryption Time (milliseconds)	Decryption Speed (milliseconds/byte)
AES (CBC)	1KB - 128	0.586	5.181 E-4	0.042	3.794 E-5
	10MB - 128	27.878	2.778 E-6	21.126	2.105 E-6
AES (CTR)	1KB - 128	0.304	2.694 E-4	0.033	2.930 E-5
	10MB - 128	9.237	9.206 E-7	8.445	8.417 E-7
AES (CTR)	1KB - 256	0.030	2.719 E-5	0.020	1.770 E-5
	10MB - 256	8.889	8.860 E-7	8.938	8.908 E-7
RSA (OAEP)	1KB - 2048	2.809	2.484 E-3	13.869	1.226 E-2
	1MB - 2048	1993.272	1.901 E-3	11192.075	1.067 E-2
RSA (OAEP)	1KB - 3072	2.574	2.276 E-3	21.871	1.933 E-2
	1MB - 3072	1913.394	1.825 E-3	16242.823	1.549 E-2

Hashing Algorithms:

Algorithm	File Size	Hashing Time (millisecond)	Hashing Speed (milliseconds/byte)
SHA - 256	1KB	3.409 E-2	3.014 E-5
	10MB	54.692	5.450 E-6
SHA - 512	1KB	2.338 E-1	2.067 E-4
	10MB	29.495	2.939 E-6
SHA3 - 256	1KB	1.869 E-1	1.652 E-4
	10MB	25.514	2.542 E-6

Digital Signature Algorithms:

Key Size	File Size	Signature Time (millisecond)	Signing Speed (milliseconds/byte)	Verification Time (milliseconds)	Verification Speed (milliseconds/byte)
2048	1KB	0.679	6.012 E-4	1.042	9.220 E-4
2048	10MB	54.980	5.479 E-6	55.180	5.499 E-6
3072	1KB	1.3062	1.154 E-3	2.326	2.056 E-3
3072	10MB	55.557	5.537 E-6	56.469	5.628 E-6

3. Performance Report:

a. Per byte speed changes for different algorithms between small and large files

Expected Performance:

Symmetric key algorithms such as AES-CBC, AES-CTR are generally expected to be **faster** than **asymmetric key algorithms** (Public Key algorithms) such as RSA, because AES uses the **same key** for both encryption and decryption, whereas RSA uses two **different keys** for encryption and decryption.

AES is expected to work much **faster** in **CTR** (Counter Mode) when compared to **CBC** (Cipher Block Chaining), due to fact of dependency and parallelization. Every block in CBC in dependent on the previous block which restricts parallelization, whereas every block in CTR mode works independently therefore performing faster than CBC.

Actual Performance:

Though there isn't a significant change in the speed for small files which could vary depending on the stochastic nature of the pseudo randomness, we could observe a significant change in the larger files.

AES-CBC mode taking the longest of 27 milliseconds, followed by AES-CTR mode taking around 8-9 milliseconds, and RSA taking the longest time of 1993 milliseconds for encrypting 10MB data, and the speed per byte resonates with the above.

Actual Performance of speed per byte matches with the expected results
Speed: AES (CTR) > AES(CBC) > RSA

b. Encryption and decryption time difference for a given encryption algorithm

Expected Performance:

The **encryption speed** and **decryption speed** should be **equal** in the **symmetric encryption** algorithms because both the encryption and decryption process use the same keys and same process to encipher and decipher the plaintext.

Conversely, for asymmetric key algorithms such as **RSA**, **decryption takes longer time than encryption**, because decryption uses private key which is typically larger than a public key. And RSA utilizes complex mathematical operations such as **modular exponentiation** to encrypt and decrypt texts which makes it even slower.

Actual Performance:

The encryption and decryption of large files in AES CBC mode varies around 20 milliseconds, and AES CTR mode takes around 9 milliseconds. Whereas RSA algorithm takes 1993 milliseconds for encrypting and 11192 milliseconds for decrypting which is approximately 10 times slower than encryption.

The actual performance resonates with the expected performance.

c. Key generation, encryption, and decryption times difference with the increase in the key size

Expected Performance:

In **Symmetric key algorithms**, increase in key size **does not significantly affect** the key generation time, encryption, and decryption time. As key generation in AES takes in the concept of **round key generation** from a master key which is the same process for both 128 bit and 256 bits key, the key generation time remains the same. Encryption and decryption time also remains the same, considering the fact that it uses the same process for enciphering and deciphering the plaintext.

In **Asymmetric key algorithms,** increase in key size **affects** the key generation time, encryption, and decryption time. As key generation in RSA requires to find two large prime numbers and do complex mathematical operations to find the public key and private key pair, and utilizing those **large prime numbers** to do encryption and decryption also takes lots of time.

Actual Performance:

The key generation times for AES-128, AES-256, RSA-2048, RSA-3072 are 0.003, 0.003, 407.677, and 820.441 respectively. The encryption and decryption times for RSA-2048 is 1993 & 11192 and for RSA-3072, its 1993 & 16242 respectively.

The actual performance resonates with the expected performance

d. Hashing time difference between the algorithms and with increase of the hash size

Expected Performance:

SHA3 is expected to perform slow when compared to SHA2 algorithms due to its innovative way of hashing technique. It uses a hashing technique called Sponge system, which focusses more on the security of the hashing and reducing collision probabilities rather than speed.

As the hash output size increases, the hashing time typically increases as more rounds of hashing are required to produce the longer hash output. For example, SHA-2 256 is expected to be faster than SHA-2 512, as it produces a shorter hash output and requires fewer rounds of hashing.

Actual Performance:

SHA3 takes the least time to produce the digest when compared to SHA2 algorithms. SHA3-256 took 25 milliseconds to hash, SHA2-512 took 29 milliseconds, and SHA2-256 took 54 milliseconds to produce the hash.

The actual performance differs from the expected performance which could be due to the underlying implementation strategy of the library used(Pycryptodome) and its corresponding hardware compatibility

e. Performance of symmetric key encryption (AES), hash functions, and public-key encryption (RSA) compared to each other

The performance of the symmetric key encryption, hash functions and RSA algorithms vary significantly with the use case and purpose.

In general, AES is much faster than RSA. Because symmetric algorithms re-uses the same key for both encryption and decryption process and asymmetric algorithms re-uses different key (public key and private key) for encryption and decryption. Generating the public-private key pair and maintaining the same creates a performance overhead.

Symmetric key algorithms are generally used for transferring larger amounts of data and asymmetric key algorithms are used for transferring small amount of data such as keys over an unsecured channel

Hash functions are generally faster than encryption algorithms as it just requires generating a hex digest using a fixed-length hash value.

The above experiment collectively compares the use of symmetric, asymmetric and hashing algorithms, and the above verdict matches with the actual performance reports.