FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Segurança e privacidade (CC2009)

PL02_9
Ana Leitão (up202207720)
João Baptista (up202207629)

A. Generating the text files

The files are generated with random text containing upper/lower letters, numbers and special characters with specified sizes for each encryption methods.
The 'gerar(file_type)' function selects the encryption method based on the argument ('file_type') and creates each file size associated with the chosen encryption type.

For each encryption method, files are generated with the following sizes:

- For AES (in bytes): 8, 64, 512, 4096, 32768, 262144, 2097152
- For SHA (in bytes): 8, 64, 512, 4096, 32768, 262144, 2097152
- For RSA (in bytes): 2, 4, 8, 16, 32, 64, 128

B. Encrypt and decrypt all files using AES (and measure the time)

The provided code performs AES encryption and decryption on a set of text files of varying sizes and measures the time taken for these operations.
The 'gerar('AES') function is called to generate text files (size of 8 bytes to 2097152 bytes) with random content for AES operations.
The code imports necessary libraries and modules for AES encryption and decryption and as a list named 'FILES_AES' that contains the name of the text files to be encrypted and decrypted.
For each file in 'FILES_AES', the code reads the file content, pads it using PKCS7 padding, and initializes AES encryption with a randomly generated key with 256 bits and IV with 128 bits. It then measures the time taken to encrypt the file using the 'timeit.default_timer()' function, converts the time to microseconds.
Similarly, the code decrypts and stores the time taken and returns a list with the mean of time taken to encrypt and decrypt for each file.

C. Encrypt and decrypt files using RSA

This code demonstrates the use of RSA encryption and decryption algorithm.
The 'generate_keypair()' function is defined to create an RSA key pair consisting of a private key and a corresponding public key with size of 2048 bits. The private key is generated using the 'rsa.generate_private_key()' function from the 'cryptography' library with specified parameters.
The 'encrypt()' function takes a plaintext message and the recipient's public key as input, encrypts the message using the RSA algorithm with OAEP, and returns the ciphertext.
The 'decrypt function' takes a ciphertext and the private key as input, decrypts the ciphertext using the RSA algorithm with OAEP, and returns the plaintext message.

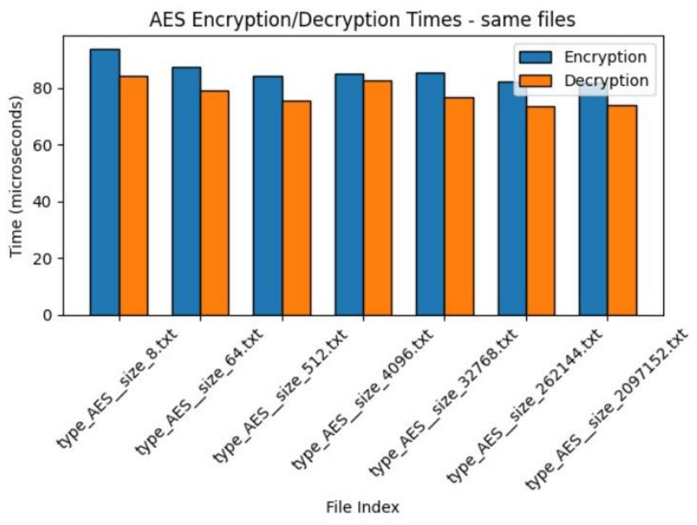
For each iteration, the code measures the time taken for encryption and decryption using the 'timeit.default_timer()' function to capture the start and end times, converts it to microseconds and stores it, returning the mean of time taken for each file size in a list.

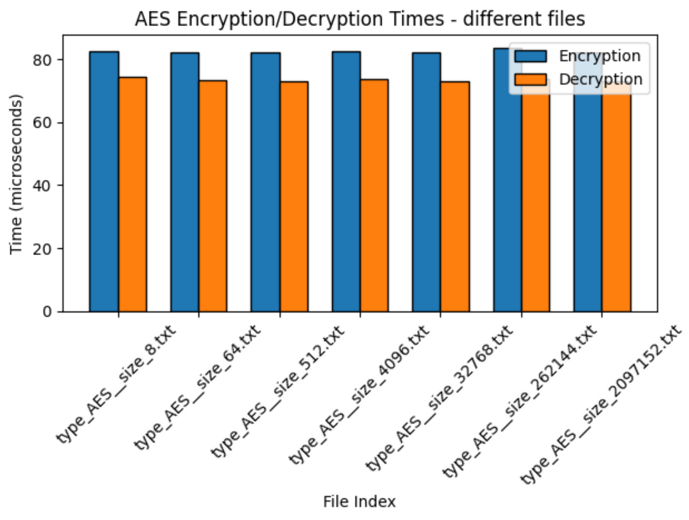
D. Measure the time for SHA-256 hash generation

The code provides a way to measure the time required to generate the SHA-256 hash of files of different sizes.
The 'SHA256()' function is responsible for calculating the SHA-256 hash of a text using the cryptography library, it initializes a SHA-256 hash object, updates it with the plaintext data, and then finalizes the hash computation.
As for determine the time taken we use timeit.default_timer() as for the previous encrypt/decrypt methods and return a list with the mean time taken for each file.

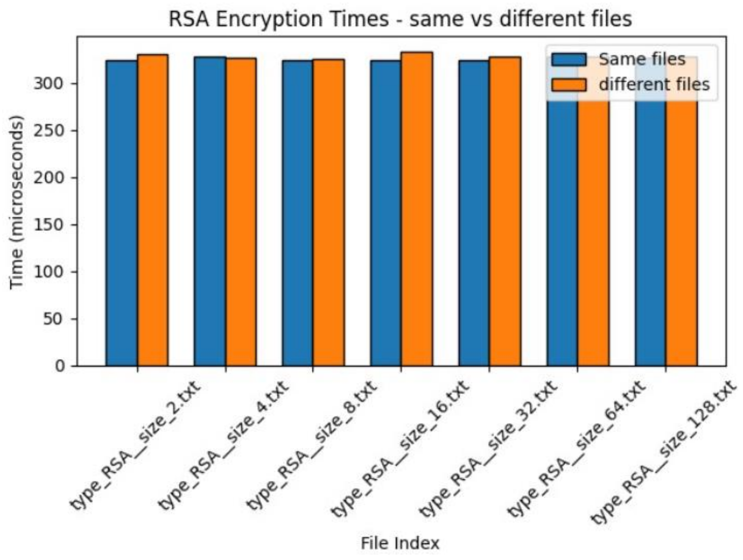
Plots showing:

(i) AES encryption/decryption times

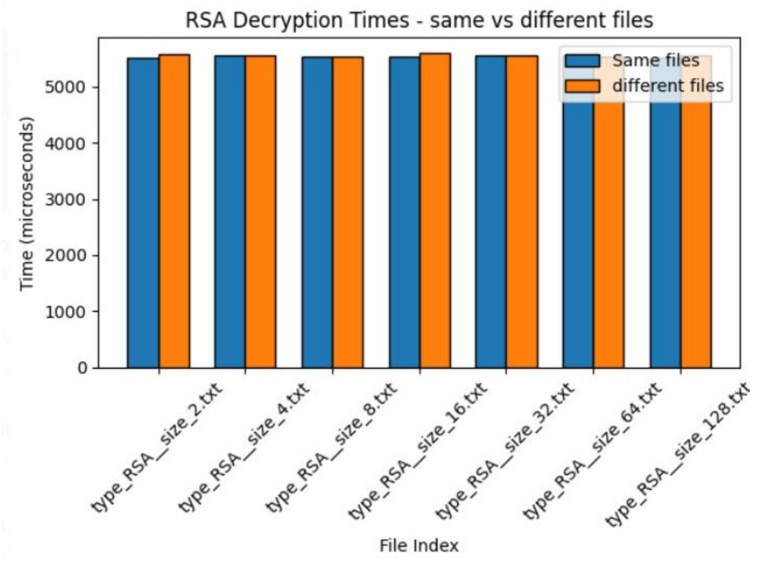




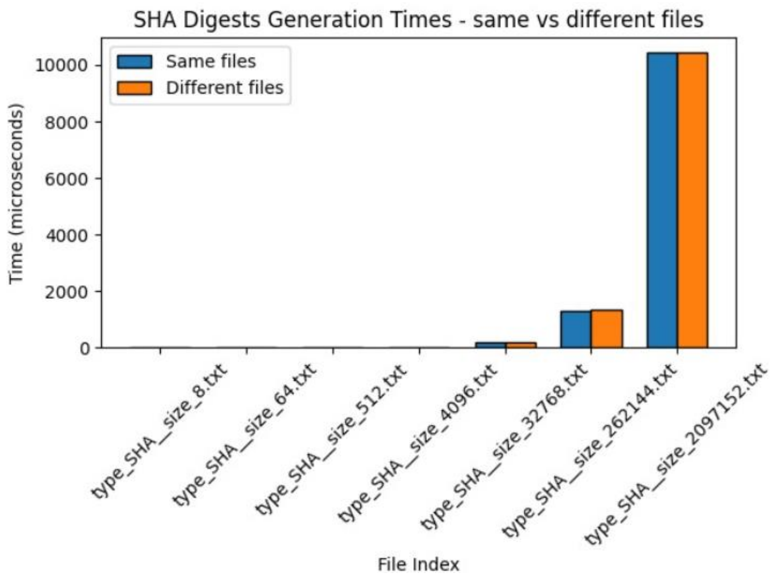
(ii) RSA encryption times



(iii) RSA decryption times



(iv) SHA digests generation times



Generation the results

We automated each algorithm in Python to iterate over the files individually, for small files it took 500 iterations and for larger files 100 iterations, the results of the first 10 tests of each algorithm were ignored.

To obtain the results for different files, we create new files 25 times and use the same rule for iterations and calculate the average of the averages of the 25 files.

The data was obtained with a computer integrated with a CPU AMD Ryzen 5-3600 (with 3.8GHz) and RAM 16GB 3200MHz.

- Comparison between AES encryption and RSA encryption

RSA encryption times are much higher than AES encryption times, while in AES the time is in the range of [80,90] microseconds, in RSA the range is [300,400] microseconds. RSA is slower than AES due to the computational complexity involved in encrypting RSA using the public key.

With this difference in time we assume that RSA taking more time means it has a better encryption method that is harder to overcome.

- Comparison between AES encryption and SHA digest generation

The digest generation time of SHA is greater than the encryption time of AES, while that of AES is in a range of [80,90] microseconds, the SHA time can go up to 10000 microseconds.

- Comparison between RSA encryption and decryption times

RSA's decryption time is greater than its encryption time, while the encryption time is in a range of [300,400] the decryption time is in a range of [5000,6000].

Encryption operation with RSA involves generating a public key, while decryption involves generating a private key. Decryption time is generally slower than encryption time due to the computational complexity involved in factoring large numbers.

Webography

Crypto101: when we wanted to better understand some concepts

Chatgpt

- B.
  (we based ourselves on exercise 3 of practical sheet 2 to encrypt, so we recur to chat gtp for decrypt)
  Q: using cryptography module from python, how do i decrypt a message that was encrypted with AES algoritm and CBC mode
- C.
  Q: Using the python module cryptography can u write a function for RSA encryption and decryption with a key of size 2048 bits
- D.
  Q: use cryptography module in python to implement an algoritm to generate hash with SHA-256 to a set file