Uma imagem com texto

Descrição gerada automaticamente

**Performance Benchmarking of Cryptographic Mechanisms**

**Relatório 1**

João Sousa up202205238

Rui Santos up202109728

CC2009: Segurança e Privacidade

Prof.º Manuel Correia

Prof.º João Vilela

Prof.º Henrique Faria

Março de 2024

Índice

[Implementação 2](#_Toc162023643)

[Descrição código para alínea B 2](#_Toc162023644)

[Imports 2](#_Toc162023645)

[Função de encriptação 2](#_Toc162023646)

[Função de desencriptação 3](#_Toc162023647)

[Funções auxiliares 3](#_Toc162023648)

[Funções relacionadas com gráficos 8](#_Toc162023649)

[Descrição código para alínea C 9](#_Toc162023650)

[Imports 9](#_Toc162023651)

[Criação de chave publica e chave privada 10](#_Toc162023652)

[Função de encriptação 10](#_Toc162023653)

[Função de desencriptação 11](#_Toc162023654)

[Funções auxiliares 11](#_Toc162023655)

[Funções relacionadas com gráficos 14](#_Toc162023656)

[Descrição código para alínea D 14](#_Toc162023657)

[Imports 14](#_Toc162023658)

[Geração de hash 15](#_Toc162023659)

[Funções auxiliares 15](#_Toc162023660)

[Funções relacionadas com gráficos 16](#_Toc162023661)

[Explicação de como geramos/obtemos os resultados 17](#_Toc162023662)

[Plots showing 18](#_Toc162023663)

[AES encryption and RSA encryption times: 23](#_Toc162023664)

[AES encryption and SHA digest generation times: 24](#_Toc162023665)

[RSA encryption and decryption times: 24](#_Toc162023666)

[Bibliografia 25](#_Toc162023667)

# Implementação

De seguida temos a descrição dos componentes principais para cada porção de código utilizada para as alíneas B, C e D do projeto.É importante referir que em todo o projeto utilizamos a biblioteca de criptografia desenvolvida por a organização PyCA.

## Descrição código para alínea B

Para melhor descrever o código decidimos dividir em diferentes secções e explicar cada uma delas com uma imagem do código e uma breve descrição.

### Imports

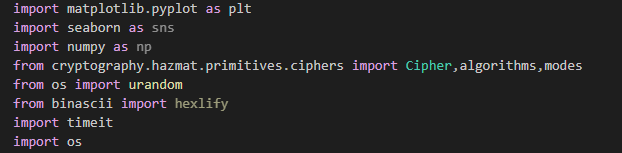




Figura 1- imports alínea B

### Função de encriptação

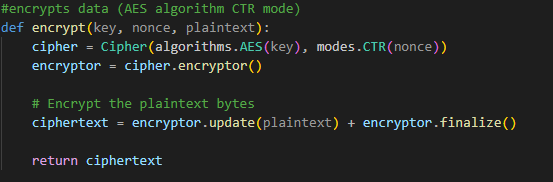


Figura 2 - função de encriptação

Para a encriptação criamos uma função que recebe os argumentos:

* Key : palavra chave secreta (terá sempre 256 bits);
* Nonce : utilizado pelo modo CTR;
* Plaintext: texto a encriptar.

No final a função devolve o texto cifrado.

### Função de desencriptação

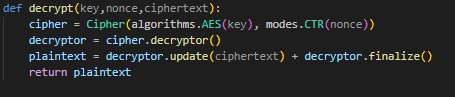


Figura 3 - função de desencriptação

Para a desencriptação criamos uma função que recebe os argumentos:

* Key : palavra chave secreta (terá sempre 256 bits);
* Nonce : utilizado pelo modo CTR;
* ciphertext: texto a desencriptar.

No final a função devolve o texto original.

### Funções auxiliares

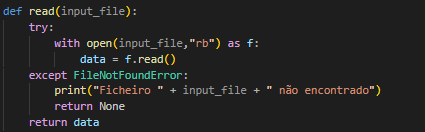


Figura 4 - função de leitura de ficheiro

Função utilizado para leitura de dados presentes num ficheiro passado como argumento. Em caso de sucesso, retorna os conteúdos de input\_file sobre a forma de sequência de bytes. Em caso de insucesso retorna None.

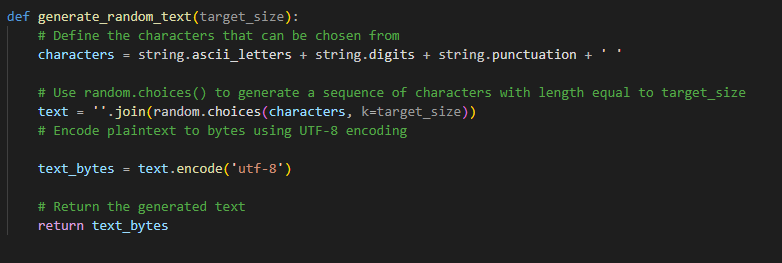


Figura 5 – função de geração de texto aleatórios

A função recebe o tamanho do ficheiro a criar, depois criar um ficheiro com o número de bytes requerido, que é codificado em bytes retornado o texto codificado.

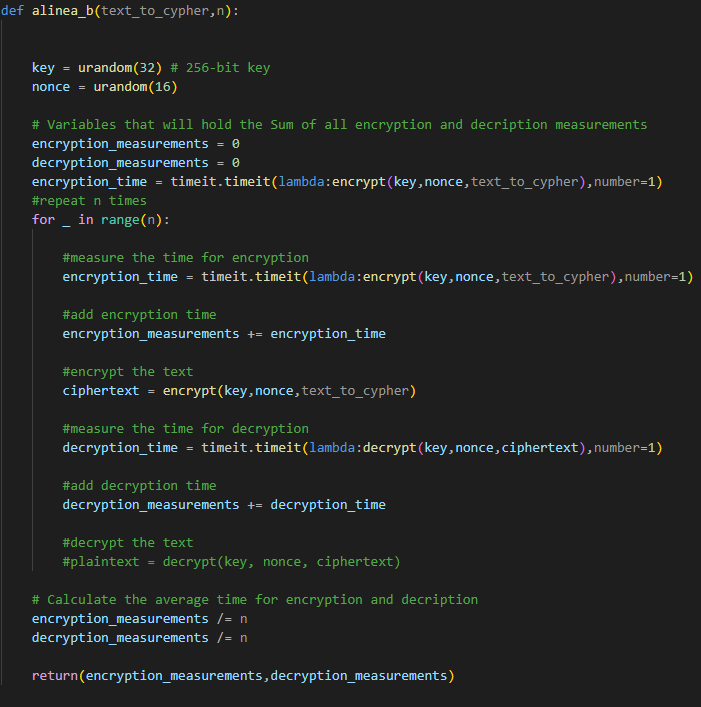


Figura 6 - função que implementa a parte “principal” da alínea B do projeto

Esta função tem como objetivo:

* Receber texto que vai ser cifrado;
* Fazer a encriptação e decriptação desse texto n vezes (nota: realizamos uma medição inicial antes do ciclo for , porque a primeira utilização do timeit dava tempos superiores ao que era suposto);
* Em cada iteração calcular o tempo que demorou a encriptar e a desencriptar (individualmente);
* Adicionar os resultados de cada iteração as respetivas variáveis auxiliares;
* Por fim , após as n iterações , fazer a média da encriptação e decriptação e retornar estes valores.

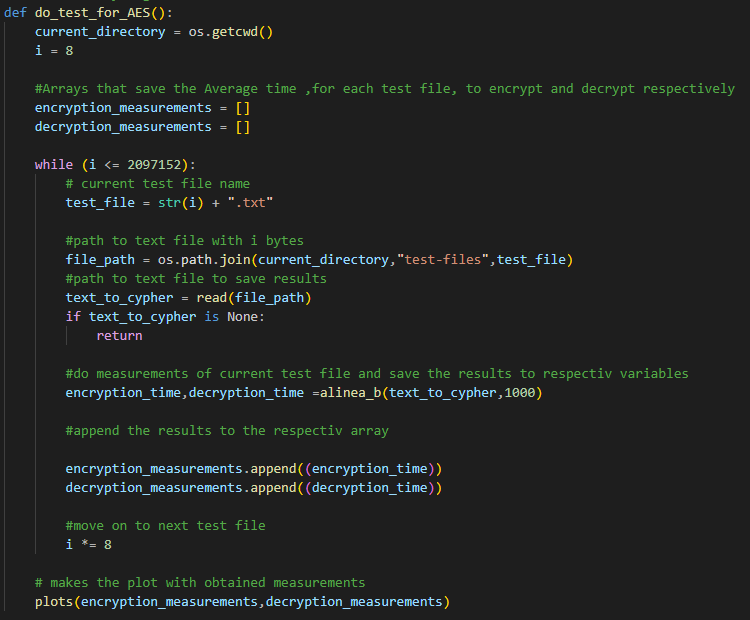


Figura 7 - função que realiza todos os testes definidos para a alínea B

Função que realizei os testes em todos os casos de teste, mencionados na alínea A do projeto para o algoritmo AES. É importante referir que nos realizamos 1000 repetições para garantir resultados estatisticamente significantes. No final são feitos os gráficos.

Uma imagem com texto, captura de ecrã, software, Sistema operativo

Descrição gerada automaticamente

Figura 8- Função que realiza os teste definidos na alínea B para ficheiros de texto aleatórios

Esta função tem como objetivo:

* Para cada i gerar n ficheiros aleatórios
* Chamar n vezes a função que calcula a média do tempo a encriptação e decriptação desse texto n vezes
* Calcular a média das médias retornadas pela função ‘alínea\_b’
* Coloca esses valores numa lista
* Chama a função ‘plots’ para fazer um gráfico

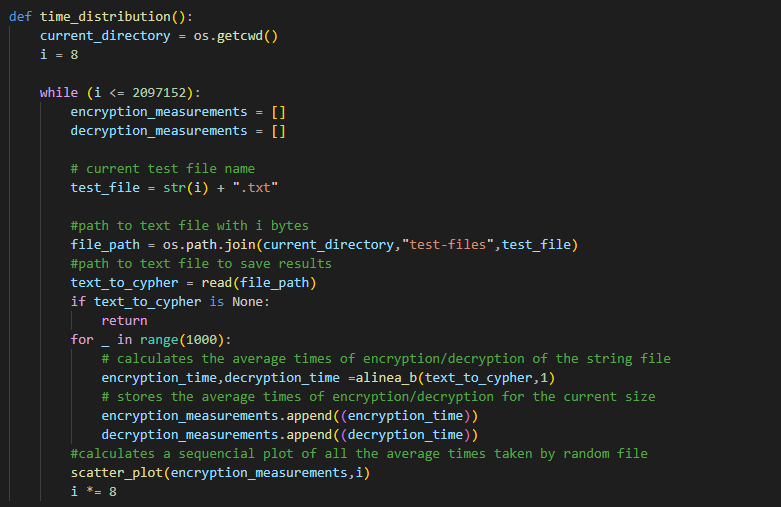


Figura 9 – **Função que calcula os tempos de encriptação n vezes para o mesmo ficheiro**

Esta função tem como objetivo:

* Receber o ficheiro texto, que vai ser cifrado;
* Chamar n vezes a função que calcula a média tempo a encriptação e decriptação desse texto 1 vezes (para não obtermos uma média)
* Em cada iteração adiciona esse tempo a uma lista;
* Chamar a função que faz um gráfico de distribuição;

### Funções relacionadas com gráficos

De seguida temos as funções que utilizamos para todos os gráficos relacionados com a alínea B do projeto.

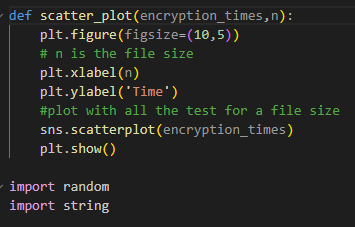


Figura 10 – Função que cria um gráfico com o número registos de tempos

Esta função criar um gráfico com todos os tempos encriptação registados na função ‘time\_distribution’ de um tamanho de ficheiro n, no eixo do x encontra-se na legenda o numero de bytes do ficheiro e no eixo do y encontra-se na legenda ‘Time’

Uma imagem com texto, eletrónica, captura de ecrã, software

Descrição gerada automaticamente

Figura 11 – Função que cria um gráfico da media dos tempos de encriptação e decriptação

A função recebe com argumentos uma lista dos tempos de encriptação ‘encryption\_times’ e uma lista dos tempos de decriptação ‘decryption\_times’ depois cria ‘x\_val’ que é uma lista cujo conteúdo é uma sequencia de números de 0 a n com tamanho do ‘encryption\_times’.

Faz um gráfico para a encriptação e outro para a decriptação

## Descrição código para alínea C

Mais uma vez, para melhor descrever o código decidimos dividir em diferentes secções e explicar cada uma delas com uma imagem do código e uma breve descrição.

### Imports

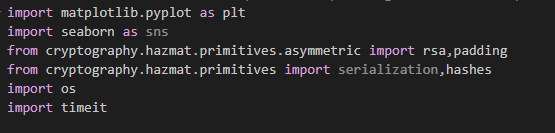


Figura 12- imports alínea C

### Criação de chave publica e chave privada

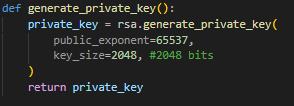


Figura 13 - função que gera chave privada e variáveis globais que representam chave privada e chave publica

Para a criação da chave privada, definimos uma função que queria uma chave privada com o tamanho da chave igual a 2048 bits e retornamos essa chave privada. É importante referir que utilizamos variáveis globais que guardam a chave privada e chave pública, e não tivemos em conta este tempo de geração da chave privada para os tempos medidos.

### Função de encriptação

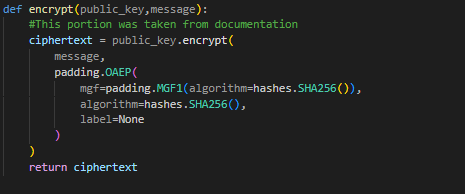


Figura 14 - função de encriptação

Para a encriptação criamos uma função que recebe os argumentos:

* Public\_key : palavra chave publica ,usada para a encriptação (obtida através da chave privada );
* message: mensagem que vamos encriptar.

Neste caso seguimos a encriptação com os parâmetros sugeridos pelos criadores da biblioteca.

### Função de desencriptação

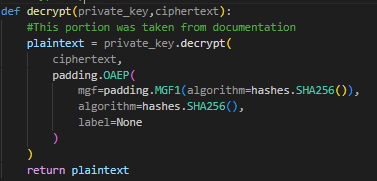


Figura 15 - função de desencriptação

Para a desencriptação criamos uma função que recebe os argumentos:

* Private\_key : palavra chave privada ,usada para a desencriptação;
* Ciphertext : texto cifrado que vamos desencriptar.

Neste caso seguimos também os parâmetros sugeridos pelos criadores da biblioteca.

### Funções auxiliares

Decidimos não acrescentar as funções read porque é exatamente igual a definidas na alínea B.

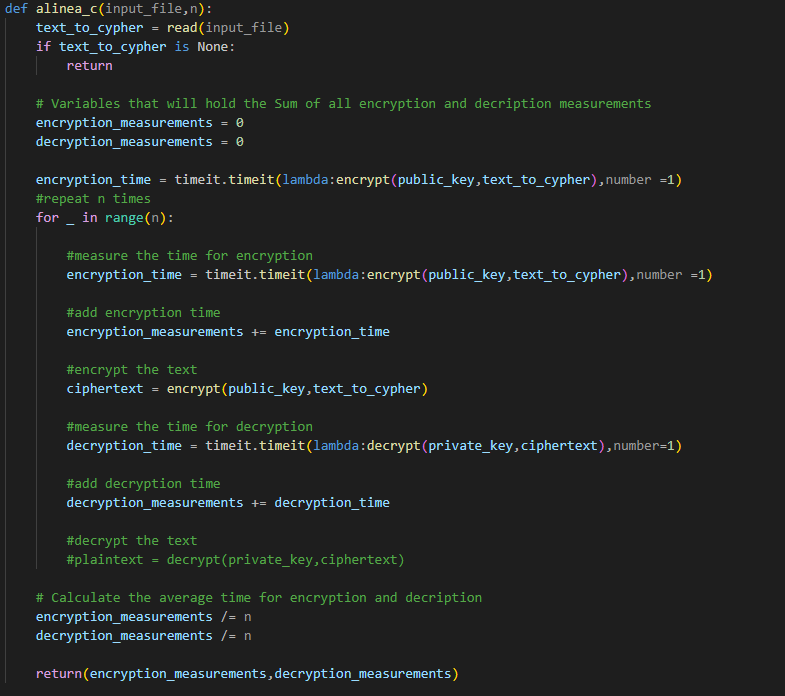


Figura 16 - função que implementa a parte “principal” da alínea C do projeto

Esta função tem como objetivo:

* Receber ficheiro de input, que vai conter o texto a ser cifrado;
* Fazer a encriptação e decriptação desse texto n vezes (nota: realizamos uma medição inicial antes do ciclo for , porque a primeira utilização do timeit dava tempos superiores ao que era suposto);
* Em cada iteração calcular o tempo que demorou a encriptar e a desencriptar (individualmente);
* Adicionar os resultados de cada iteração nas variáveis auxiliares respetivas;
* Por fim, após as n iterações, fazer a média da encriptação e decriptação e retornar estes valores.

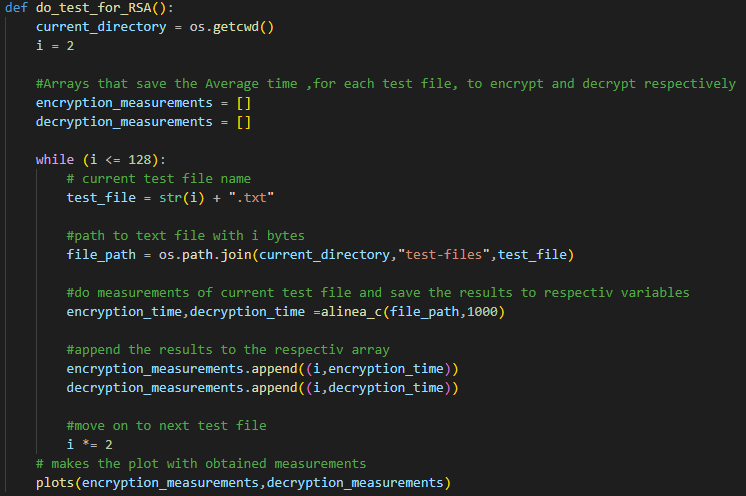


Figura 17 - função que realiza todos os testes definidos para a alínea C

Função que realizei os testes em todos os casos de teste mencionados na alínea A do projeto para o algoritmo RSA. É importante referir que nos realizamos 1000 repetições para garantir resultados estatisticamente significantes. No final são feitos os gráficos.

### Funções relacionadas com gráficos

De seguida temos as funções que utilizamos para todos os gráficos relacionados com a alínea C do projeto.

Uma imagem com texto, eletrónica, captura de ecrã, software

Descrição gerada automaticamente

Figura 18 - Função que cria um gráfico da media dos tempos de encriptação e decriptação

A função recebe com argumentos uma lista dos tempos de encriptação ‘encryption\_times’ e uma lista dos tempos de decriptação ‘decryption\_times’ depois cria ‘x\_val’ que é uma lista cujo conteúdo é uma sequencia de números de 0 a n com tamanho do ‘encryption\_times’.

Faz um gráfico para a encriptação e outro para a decriptação

## Descrição código para alínea D

Tal como fizemos anteriormente, vamos descrever o código em diferentes secções e explicar cada uma delas com uma imagem do código e uma breve descrição.

### Imports

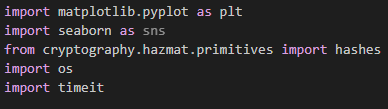


Figura 19 - imports alínea D

### Geração de hash

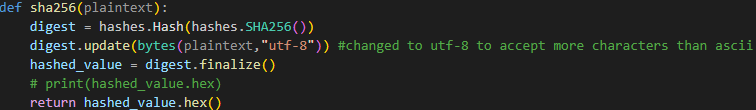


Figura 20 - função que gera hash SHA-256

Recebe como argumento o texto em que vamos aplicar a função de hashing.No final retorna a resultante da aplicação da função hash em Hexadecimal.

### Funções auxiliares

Decidimos não acrescentar as funções read porque é exatamente igual a definida na alínea B.

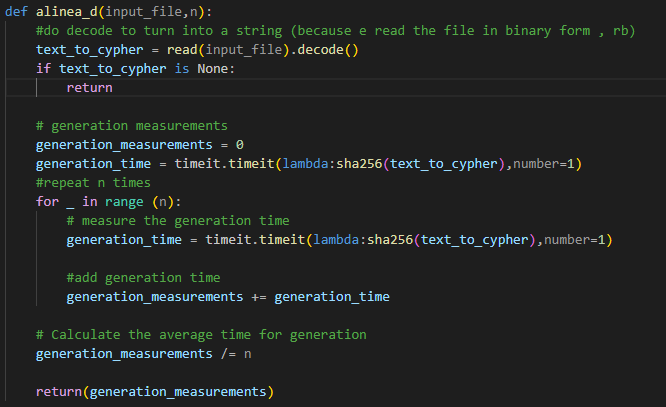


Figura 21 - função que implementa a parte “principal” da alínea D do projeto

Esta função tem como objetivo:

* Receber ficheiro de input, que vai conter o texto que vamos aplicar a função de hash;
* Aplicar a função de hash a esse texto n vezes (nota: realizamos uma medição inicial antes do ciclo for , porque a primeira utilização do timeit dava tempos superiores ao que era suposto);
* Em cada iteração calcular o tempo que demorou a gerar a hash ;
* Adiciona o resultado de cada iteração numa variável auxiliar;
* Por fim, após as n iterações, fazer a média geração e retorna este valor.

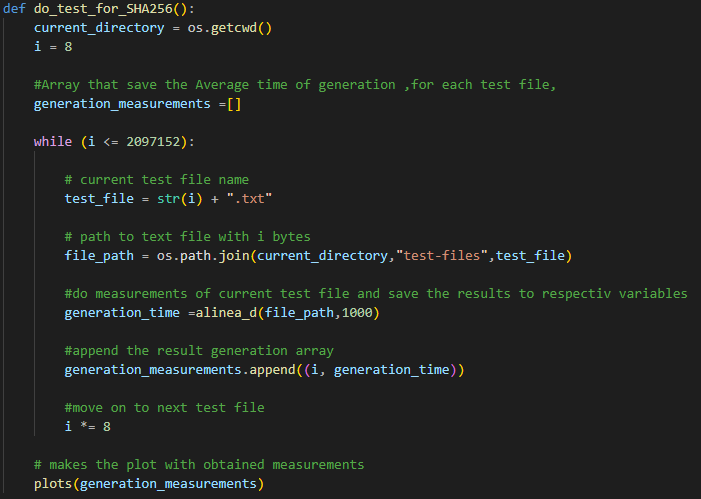


Figura 22- função que realiza todos os testes definidos para a alínea D

Função que realizei os testes em todos os casos de teste mencionados na alínea A do projeto para o algoritmo SHA-256. É importante referir que nos realizamos 1000 repetições para garantir resultados estatisticamente significantes. No final são feitos os gráficos.

### Funções relacionadas com gráficos

De seguida temos as funções que utilizamos para todos os gráficos relacionados com a alínea D do projeto.

Uma imagem com texto, captura de ecrã, Tipo de letra

Descrição gerada automaticamente

Figura 23 - Função que cria um gráfico da media dos tempos de encriptação e decriptação

A função recebe com argumentos uma lista dos tempos de geração ‘generation\_times’ depois cria ‘x\_val’ que é uma lista cujo conteúdo é uma sequência de números de 0 a n com tamanho do ‘generation\_times’.Faz um gráfico para a geração.

# Explicação de como geramos/obtemos os resultados

Especificações do computador:

Nome do dispositivo: Huawei Matebook D15

Processador: Intel Core i5 1135G7 / 2.4 GHz

Sistema operativo: Windows 11 Home

Graphics card: Iris Xe Graphics

RAM capacity: 8 GB

Capacidade de armazenamento: 512 GB SSD NVMe

Para gerar o tempo de encriptação/decriptação fizemos 1000 iterações por tamanho de arquivo usando o mesmo arquivo em cada iteração, com a média desses tempos para um determinado arquivo calculamos um gráfico que mostra a diferença de tempo conforme os arquivos ficam maiores, o eixo do x no gráfico não mostra o número de bytes do ficheiro por motivos de escala, uma tabela de correspondência está perto de cada gráfico. E por motivos de escala, pode haver 2 gráficos para o mesmo teste, sendo que um deles não possui os tamanhos dos arquivos maiores. Especificamente no AES existe um gráfico de dispersão que representa cronologicamente os tempos encriptação/decriptação num tamanho de arquivo específico, existe também outra função que gera 100 ficheiros aleatórios de um determinado tamanho e cada arquivo é testado 100 vezes e o tempo médio gasto por cada arquivo é usado para calcular a média de um determinado tamanho de arquivo

# Plots showing

Figure 1 - Gráfico da média dos tempos de encriptação para todos os tamanhos de ficheiros pedidos

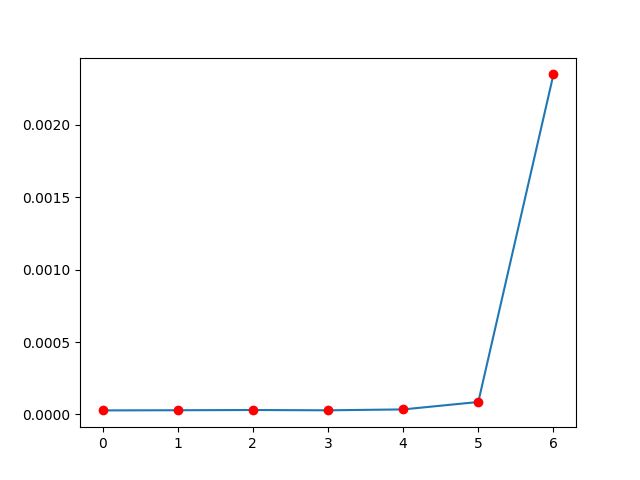
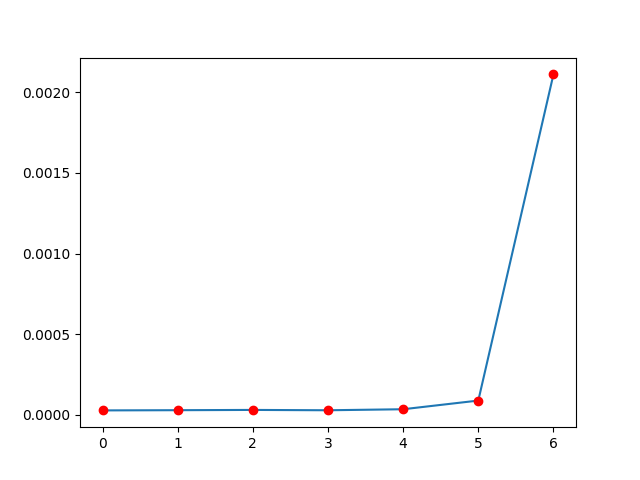
AES\_ENCRYPTION\_TIMES AES\_DECRYPTION\_TIMES

Figure 2 - Gráfico da média dos tempos de decriptação para todos os tamanhos de ficheiros pedidos

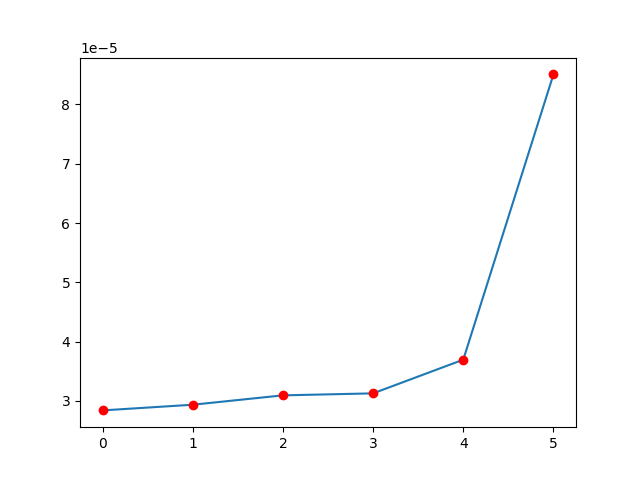


Figure 3 - Gráfico da média dos tempos de encriptação para todos os ficheiros pedidos exceto 2097152 bytes(por motivos de escala)

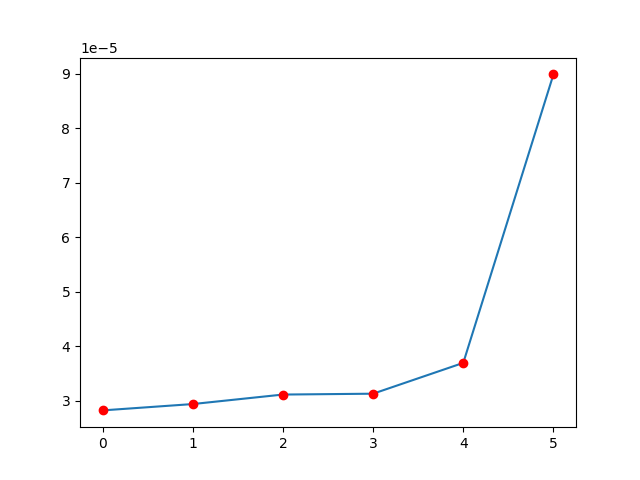
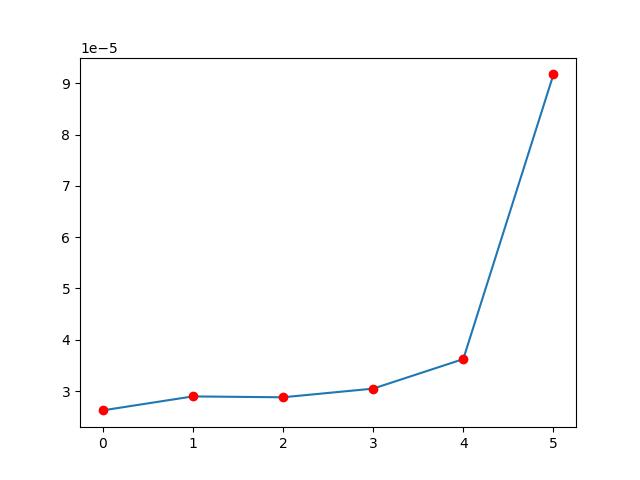


Figure 4 - Gráfico da média dos tempos de decriptação para todos os ficheiros pedidos exceto 2097152 bytes(por motivos de escala)

|  |  |  |
| --- | --- | --- |
| Valor de X no gráfico | Tamanho em bytes correspondente | Como espectável os tempos de encriptação desencriptação são muito semelhantes por o AES se tratar de um algoritmo de encriptação simétrica o que significa que o processo de encriptação e decriptação são idênticos sendo um  algoritmo bastante rápido um tempo próximo de 3 µs o que se deve ao facto de no modo RBC  cada bloco de texto que é encriptado é independente e por isso podem ser encriptados em paralelo aumentando a eficiência do algoritmo |
| 0 | 8 |
| 1 | 64 |
| 2 | 512 |
| 3 | 4096 |
| 4 | 32786 |
| 5 | 262144 |
| 6 | 2097152 |

Uma imagem com texto, file, diagrama, captura de ecrã

Descrição gerada automaticamenteAES RANDOM FILES ENCRYPTION/DECRYPTION TIMES

Figure 5 - Gráfico da média, da media dos tempos de encriptação/decriptação para ficheiros aleatórios exceto 2097152 bytes(por motivos de escala)

Figure 6 - - Gráfico da média, da media dos tempos de encriptação/decriptação para ficheiros aleatórios

|  |  |  |
| --- | --- | --- |
| Valor de X no gráfico | Tamanho em bytes correspondente | Como espectável os tempos de encriptação desencriptação de ficheiros aleatórios são muito semelhantes aos tempos para um só ficheiro porque os tempos de encriptação/decriptação do AES no modo RBC porque a única coisa que influencia o tempo de encriptação/decriptação do ficheiro para alem da variabilidade da maquina é o numero de blocos que necessitam que não mudam com a aleatoridade |
| 0 | 8 |
| 1 | 64 |
| 2 | 512 |
| 3 | 4096 |
| 4 | 32786 |
| 5 | 262144 |
| 6 | 2097152 |

Uma imagem com texto, captura de ecrã, file, quadro branco

Descrição gerada automaticamenteOs gráficos abaixo representam o tempo necessário, em segundos, para encriptar um arquivo 1000 vezes, cada ponto representa um teste, cada gráfico tem um número no eixo representando o número de bytes que o arquivo possui, a representação é cronológica

Uma imagem com texto, captura de ecrã, file, diagrama

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, file, Gráfico

Descrição gerada automaticamenteUma imagem com captura de ecrã, texto, diagrama, file

Descrição gerada automaticamente

Uma imagem com captura de ecrã, texto, diagrama, Gráfico

Descrição gerada automaticamente

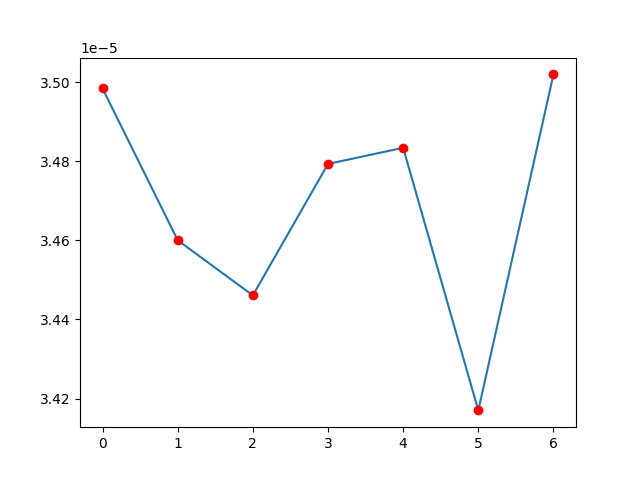
Uma imagem com texto, captura de ecrã, diagrama

Descrição gerada automaticamente

Uma imagem com texto, captura de ecrã, file

Descrição gerada automaticamente

Como se pode verificar nos gráficos acima os tempos de encriptação não são constantes, existe alguma variação o que na nossa opinião deve-se ao facto que o computador não executa imediatamente todas as operações podendo por isso o mesmo teste demorar mais ou menos tempo dependendo das tarefas que está a executar num momento específico, daí a importância de usar medias estatisticamente relevantes para calcular os tempos de encriptação/decriptação

RSA\_ENCRYPTION\_TIMES  RSA\_DECRYPTION\_TIMES

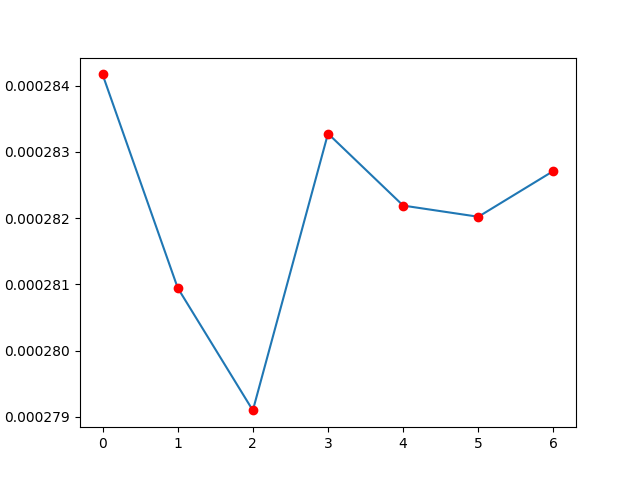


Figure 7 - Gráfico da média dos tempos de decriptação para todos os ficheiros pedidos

Figure 8 - Gráfico da média dos tempos de encriptação para todos os ficheiros pedidos

|  |  |  |
| --- | --- | --- |
| Valor de X no gráfico | Tamanho em bytes correspondente | Como espectável os tempos de encriptação estão na ordem dos 35 µs já o tempo de desencriptação é da casa dos 280 µs quase 100 vezes mais lento o que é explicável pelo facto de do expoente de encriptação ‘e’ ser muito menor que o expoente de decriptação ‘d’o que resulta num aumento no numero de multiplicação necessárias.  Encrypt: C = M^e mod n  Decrypt: M = C^d mod n = (Me)d mod n = M |
| 0 | 2 |
| 1 | 4 |
| 2 | 8 |
| 3 | 16 |
| 4 | 32 |
| 5 | 64 |
| 6 | 128 |

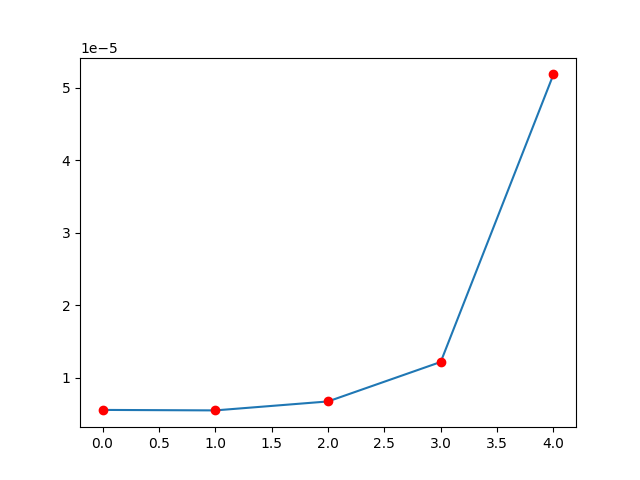
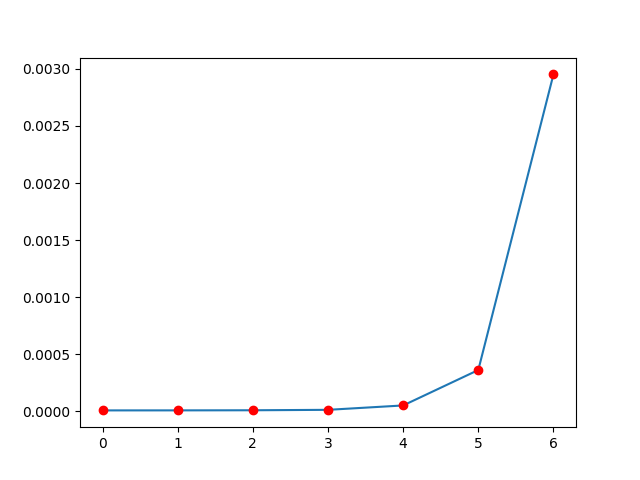
SHA\_GENERATION\_TIMES

Figure 9 - - Gráfico da média dos tempos de geração para todos os ficheiros pedidos exceto,262144 e 2097152 bytes(por motivos de escala)

Figure 10 - - Gráfico da média dos tempos de geração para todos os ficheiros pedidos

|  |  |  |
| --- | --- | --- |
| Valor de X no gráfico | Tamanho em bytes correspondente | Os tempos de geração de SHA para ficheiros até  512 bytes é menos de 1 µs sendo que existe um aumento para pouco acima de 1 µs para 512 bytes 8 µs para 32786 bytes, para 262144 bytes menos de 500 µs e para 20987152 bytes 3000 µs |
| 0 | 8 |
| 1 | 64 |
| 2 | 512 |
| 3 | 4096 |
| 4 | 32768 |
| 5 | 262144 |
| 6 | 2097152 |

# AES encryption and RSA encryption times:

Os tempos de encriptação AES e RSA são semelhantes apesar do tamanho dos ficheiros RSA ser substancialmente inferior, por exemplo o tempo de encriptação de 128 bytes em RSA é equivalente ao tempo de encriptar 32768 bytes em AES isto deve-se ao facto da encriptação em RSA envolver exponenciação que é um processo computacionalmente pouco eficiente.

# AES encryption and SHA digest generation times:

No que consta a encriptação do algoritmo AES , quando comparado ao algoritmo SHA , concluimos que para ficheiros com tamanho menor a 4096 bytes ambos os algoritmos tem tempos semelhantes contudo a medida que os ficheiro vão aumentando de tamanho , o SHA tem melhor performance . Este comportamento é expectavel , visto que o objetivo de algoritmos de *Hashing* , como o SHA ,é transformar um bloco grande de dados em blocos menores usando uma função de Hash , sendo essa hash gerada irreversivel. Pelo contrário , o AES tem uma performance pior com ficheiros de maior tamanho porque o proprio funcionamento do algoritmo envolve diversas operações sobre blocos ( 16 bytes cada) em 14 rondas (devido a chave de 256 bits) , tornado-o computacionalmente mais complexo.

# RSA encryption and decryption times:

Como referido anteriormente isto deve-se ao facto do expoente de encriptação ser muito inferior ao de decriptação e como a exponenciação é um processo computacionalmente pouco eficiente um aumento no expoente resulta num aumento significativo no tempo de execução dái o facto do tempo encriptação ser muito inferior ao tempo de decriptação

# Bibliografia

* <https://cryptography.io/en/latest/hazmat/primitives/>
* <https://cryptography.io/en/latest/hazmat/primitives/asymmetric/rsa/>
* <https://cryptography.io/en/latest/hazmat/primitives/symmetric-encryption/>
* <https://ioflood.com/blog/timeit-python/>
* <https://docs.python.org/3/library/timeit.html>
* <https://stackoverflow.com/questions/18458734/how-do-i-plot-list-of-tuples>
* [https://www.loginradius.com/blog/engineering/encryption-and-hashing](https://www.loginradius.com/blog/engineering/encryption-and-hashing/)/