U.PORTO

FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Professor João Vilela

Segurança e Privacidade

Primeiro Trabalho: Criptografia

Abril de 2022

Trabalho realizado pelo Grupo S, constituído por:

Pedro Leite - 201906697

Pedro Carvalho - 201906291

Índice

1. Introdução

Foi nos proposto aplicar dois algoritmos de encriptação/desincriptação : o AES e o RSA e um algoritmo de digestão: o SHA. Ao longo do relatório explicamos os diversos algoritmos e a sua implementação em Python. Também, demonstramos e comparamos os resultados do tempo que demora a processar ficheiros com tamanhos variados, do AES com o RSA, do AES com o SHA, e dos tempos de encriptção e desincriptação do RSA. Para a geração dos ficheiros criamos 3 funções diferentes para cada algoritmo, onde no AES e no SHA cria 7 ficheiros diferentes com 8, 64, 512, 4096 32768, 262144 e 2047152 bytes (ou seja, 8^n sendo n o número do ficheiro de 1 a 7) e no SHA cria também 7 ficheiros diferentes, mas com 2, 4, 8, 16, 32, 64 e 128 bytes (ou seja, 2^n sendo n o número do ficheiro de 1 a 7). Para determinar o tempo que demorou a encriptar/desincriptar/digerir, medimos o tempo no início do algoritmo e no fim e calculamos a diferença. Para tal utilizamos a biblioteca "time". Para a demonstração gráfica utilizamos as funções "matplotlib" e "statics". As demonstrações gráficas foram feitas num MacBook Pro com o sistema operativo macOS Big Sur 11.6.4 e processador 1.4GHz Intel Core i5 , na versão de Python 3.9.5.

2. AES (Advanced Encryption Standard)

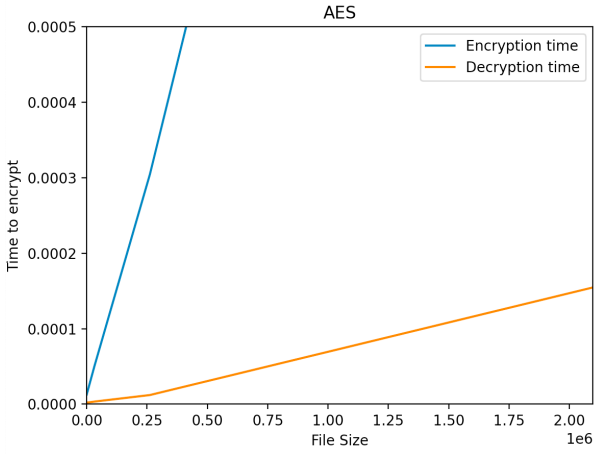
AES é um algoritmo de encriptação simétrica, o que significa que utliza a mesma chave para encriptar o texto inicial (plain textl) e para desencriptar o texto cifrado (cipher text), logo, o emissor e o recetor têm de saber qual é a chave. A encriptação é feita em blocos (inicial vector), que mais tarde se juntam para formar o cipher text. Às vezes o tamanho do plain text pode não ser suficiente, porque não é múltiplo do tamanho do bloco de encriptação, e para esse caso é necessário adicionar enchimento (padding) para encaixar o conteúdo da mensagem no bloco, mas depois será removido aquando da desencriptação. A maneira como o algoritmo de encriptação AES funciona é, para chegarmos a um "x" que é o estado, fazemos o "xor", de "p" que é o plain text, com a chave "k", isto ao longo de várias rondas.

2.1. Implementação do AES

Começamos por definir o padder (e o unpadder) e os blocos de encriptação que vão ser de 128 bits (16 bytes). E o tamanho da chave que vai ter 256 bits (32 bytes). A encriptação vai ser feita com o modo "CBC (Block Cipher Mode of Operation)", que faz a ligação dos blocos e garante confidencialidade e autenticidade. Dentro do ficheiro, convertemos a string de caracteres que lemos do ficheiro em bytes e começamos o processo de encriptação. Primeiramente adicionamos o padding caso seja necessário. E de seguida encriptamos o nosso plain text e geramos o ciphertext. Depois de termos o ciphertext guardado, fazemos o processo inverso para a desencriptação, primeiro transformamos o ciphertext em plaintext e de seguida removemos o padding.

2.2. Demonstração dos Resultados do AES

Encriptamos/desincriptamos o conteúdo de cada ficheiro 1000 vezes, vemos quanto tempo demorou em cada um deles e calculamos a média. É possível ver através do gráfico que quão maior for o tamanho do ficheiro, maior vai ser o tempo a encriptar/desincriptar, e o tempo de encriptação, vai ser sempre substancialmente maior do que o de desincriptação.



Cada célula do array do std (standard deviation) e da media, representam um ficheiro. Encriptamos e desincriptamos o conteúdo de cada ficheiro 1000 vezes, para aproximar a média do desvio padrão.

Std encrypt: [5.366752559327504e-06, 4.474751210782591e-06, 2.7506184580798174e-06, 5.152830548608079e-06, 8.616414660495294e-06, 5.1961386448802354e-05, 0.00017782083588171213]

Média encrypt: [1.261299999999843e-05, 1.2124999999998387e-05, 1.3107999999996567e-05, 1.7166999999999932e-05, 5.087899999999923e-05, 0.0003045480000000018, 0.0026925240000000065]

Std decrypt: [4.160089222882494e-06, 3.477409434672264e-06, 4.082112031612557e-06, 6.674195148547279e-06, 8.72048983388762e-06, 1.6340825762071588e-05, 8.562582546162335e-05]

Média decrypt: [1.7842000000005243e-06, 1.8346000000001971e-06, 2.015499999999837e-06, 2.1892000000000688e-06, 3.4348000000001378e-06, 1.207060000000051e-05, 0.0001545785999999997]

2. RSA (Rivest Shamir Adleman)

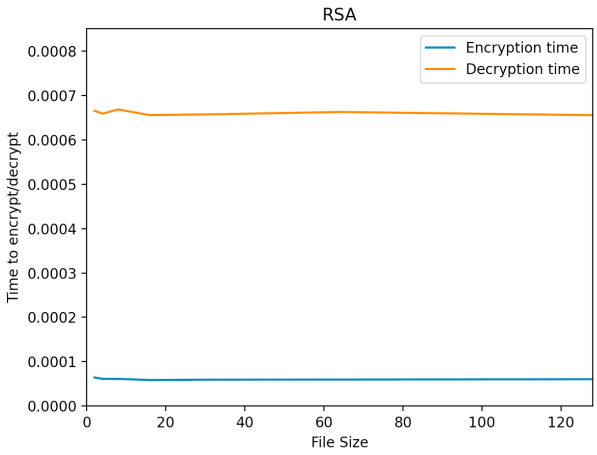
RSA é um algoritmo de encriptação assimétrica, o que significa que utiliza uma chave pública, que pode ser do conhecimento do emissor e do recetor, e uma chave privada, que só é acessível ao proprietário. O plain text é cifrado utilizando a chave pública, mas só pode ser decifrado usando a chave privada e vice-versa, se for cifrado com a chave privada, só pode ser decifrado utilizando a chave pública. Para encriptar um texto "m", que quando cifrado se transforma em "c", utilizando a chave pública "n" e "e", fazemos: "$m^e \equiv c \bmod n$". Na desincriptação para voltarmos à mensagem "m", fazemos: "$c^d \equiv m \bmod n$".

2.1. Implementação do RSA

Geramos a chave privada, que tem de ter como expoente, 3 ou 65537, neste caso escolhemos 65537, que é o expoente mais usual, o seu tamanho tem de ser pelo menos 2048, que foi o valor que escolhemos. A partir da chave privada que acabamos de gerar, geramos a respetiva chave pública. Dentro do ficheiro, convertemos a string de caracteres em bytes. E encriptamos usando a nossa chave pública. Para desincriptar utilizamos a chave privada. Para a encriptação e desincriptação utilizamos o paddding OAEP ("Optimal Asymmeetric Encryption Padding"), este adiciona elementos aleatórios que permite converter esquemas de encriptação determinísticos em esquemas probabilísticos e impede a desencriptação parcial do cipher text.

2.2. Demonstração dos Resultados do RSA

Encriptamos/desincriptamos o conteúdo de cada ficheiro 1000 vezes, vemos quanto tempo demorou em cada um deles e calculamos a média. É possível ver através do gráfico que o tempo de desincriptação vai ser sempre muito superior ao de encriptação, sendo que ambos não variam com o tamanho do ficheiro, mantendo o tempo sempre constante. Se a nossa janela de repetições fosse maior, poderíamos ver que ambas as funções iriam ser exponenciais.



Cada célula do array do std (standard deviation) e da media, representam um ficheiro. Encriptamos e desincriptamos o conteúdo de cada ficheiro 1000 vezes, para aproximar a média do desvio padrão.

Std encrypt: [1.831751149111458e-05, 8.915099270068308e-06, 9.486794150986442e-06, 7.300393122261761e-06, 1.0067059237032235e-05, 9.122983548659533e-06, 9.49016138763022e-06]

RSA Encryption Time Array: [6.445200000000396e-05, 6.152200000000053e-05, 6.135599999999863e-05, 5.883400000000094e-05, 5.960799999998567e-05, 5.981999999999488e-05, 6.062999999997487e-05]

Std decrypt: [0.0001573776019279374, 0.0001537464202404112, 0.0001616351206149648, 0.0001515563233591811, 0.0001560354722093327 3, 0.0001581507039034138 5, 0.0001601708140225988 7]

RSA Decryption Time Array: [0.000665510999999998, 0.0006588349999999927, 0.0006682639999999935, 0.0006555129999999973, 0.0006573089999999908, 0.0006625500000000058, 0.0006553550000000019] 1.8346000000001971e-06, 2.015499999999837e-06,

2.1892000000000688e-06, 3.4348000000001378e-06,
1.207060000000051e-05, 0.00015457859999999997]

2. SHA (Security Hashing Algorithm)

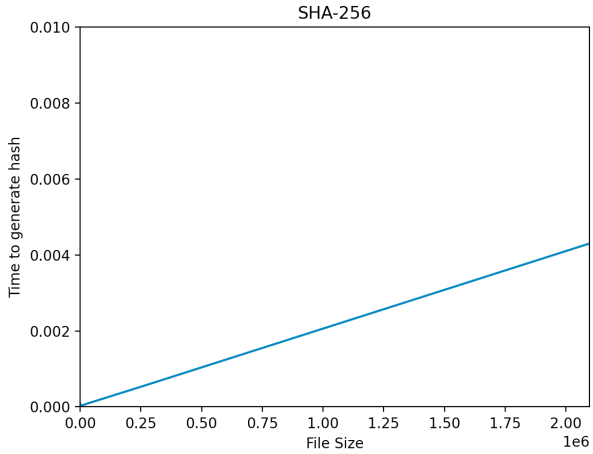
SHA é um algoritmo que através de um plain text, produz uma hash de 32 bytes (no caso da SHA-256). Uma hash é originada, quando transformamos um bloco de texto, noutro completamente indistinguível, sendo impossível conseguir voltar ao bloco de texto original. A maneira de como o algoritmo de digestão funciona é: adiciona bits de padding ao plain text, inicializa o buffer da hash com 256 bits, processa o plain text em blocos, cada ronda recebe um buffer da hash e uma parte do plain text em blocos, o output vai ser um texto com 256 bits.

2.1. Implementação do SHA

Começamos por converter a string de caracteres em bytes. E aplicamos a função de hash, neste caso a de 256 bits, ao plain text.

2.2. Demonstração dos Resultados do SHA

Calculamos a digestão do conteúdo de cada ficheiro 1000 vezes, vemos quanto tempo demorou em cada um deles e calculamos a média. Que é uma função linear crescente, como se pode verificar.



Cada célula do array do std (standard deviation) e da media, representam um ficheiro. Aplicamos a digestão ao conteúdo de cada ficheiro 1000 vezes, para aproximar a média do desvio padrão.

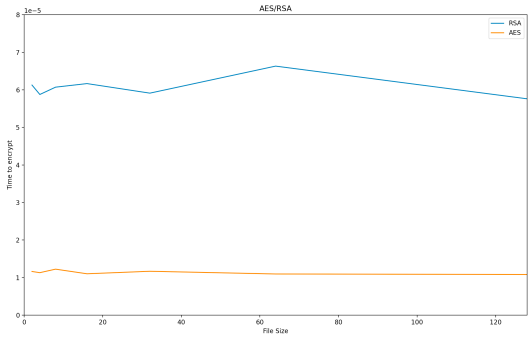
Std Hashing: [0.0017322061414371064, 6.701240976905418e-06, 4.783813214249197e-06, 3.9245791266504895e-06, 1.1975329829464801e-05, 2.7797024185222037e-05, 0.00011973849918545631]

SHA Hashing Average Time Array: [7.76780000000008e-05, 2.5173999999997808e-05, 2.350400000000352e-05, 3.100900000000428e-05, 9.23780000000014e-05, 0.0005525110000000002, 0.004300260999999995]

5. Comparação de Resultados

5.1. Comparação entre o AES e RSA

Para ficheiros do mesmo tamanho (potências de base 2 até 2^7 inclusive), reparamos que se usássemos potências de base 8 até 8^7, quando chegávamos ao ficheiro de 4096 bytes, em RSA, encontrávamos um erro que dizia que a encriptação do mesmo não era possível. Posto isto, neste caso a encriptação em RSA é mais rápida do que em AES.

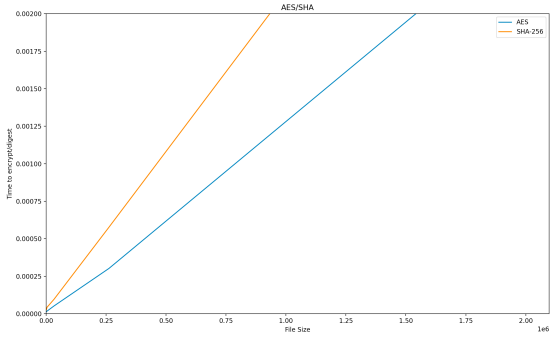


avg RSA encryption time: [6.128300000000064e-05, 5.877300000000352e-05, 6.0723000000004304e-05, 6.167399999998758e-05, 5.912999999996149e-05, 6.632999999998823e-05, 5.7623999999994344e-05]

avg AES encryption time: [1.1620000000002406e-05, 1.1303000000003393e-05, 1.2238999999988564e-05, 1.1008000000004791e-05, 1.1675999999995134e-05, 1.0955000000015148e-05, 1.0815000000006236e-05]

5.2. Comparação entre o AES e SHA

Para ficheiros do mesmo tamanho (potências de base 8 até 8^7 inclusive), percebemos que demoramos mais tempo para encriptar o ficheiro usando AES do que a fazer a digestão com o SHA-256. Por análise dos valores da média dos tempos percebemos que o AES demora cerca de duas vezes menos.



AES avg encryption time: [1.2849999999998918e-05, 1.22530000000034e-05, 1.3434999999999642e-05, 1.7544999999998366e-05, 5.146599999999779e-05, 0.0003020849999999955, 0.0027374480000000078]

SHA avg encryption time: [2.3269000000005313e-05, 2.46269999999964e-05, 2.532699999999899e-05, 4.345100000000257e-05, 9.58619999999919e-05, 0.0005770350000000008, 0.0044724549999999915]

5.3. Comparação entre os Tempos de Encriptação/Desincriptação do RSA

No RSA o tempo de desincriptação é maior do que o de encriptação, apesar de que para os ficheiros utilizados, tanto a encriptação, como a encriptação é constante. Mas para ficheiros maiores, este crescimento para de ser constante.

6. Conclusão

Concluímos, que o AES demonstrou ser mais rápido para os ficheiros que geramos. E que ao testar ficheiros maiores (a partir de 4096 bytes) o RSA simplesmente não conseguia encripta-los, devido ao seu crescimento exponencial, portanto este algoritmo deve ser utilizado apenas, para ficheiros mais pequenos.