FACULDADE DE CIÊNCIAS DA UNIVERSIDADE DO PORTO

RELATÓRIO DO PRIMEIRO TRABALHO DE SEGURANÇA E PRIVACIDADE

Felipe Valverde Garcia (201902399)

Murilo Rosa (201900689)

2022

Chapter 1

Introdução

Este trabalho tem como objetivo analisar as performances de 3 diferentes algoritmos relacionados à criptografia, nomeadamente:

- **AES** - Advanced Encryption Standard, também conhecido como Rijndael
- **RSA** - Rivest-Shamir-Adleman
- **SHA-256** - Secure Hashing Algorithm

Para a análise, os três algoritmos foram avaliados contra ficheiros de diversos tamanhos. Para o AES e o SHA, os tamanhos dos ficheiros utilizados foram de 8, 64, 512, 4096, 32768, 262144 e 2047152 bytes, e para o RSA, que conta com uma limitação mais estrita em relação ao tamanho da mensagem que pode ser encriptada, os tamanhos foram de 2, 4, 8, 16, 32, 64 e 128 bytes.

1.1 Setup

Os algoritmos foram implementados com a utilização da biblioteca Python **cryptography**, nomeadamente seu módulo hazmat. Os testes foram realizados em sequência, e o programa foi separado em três diferentes ficheiros: generator, processor e plotting. O programa generator tem como função gerar os ficheiros de teste e enviar seus nomes ao processor, que faz os testes e mede o tempo médio de execução de cada um. Por fim, o programa plotting recolhe os dados e distribui-os em gráficos para uma simplificada visualização.
Todo o programa foi testado no sistema operacional Windows 11, na versão 22H2. A máquina utilizada conta com uma memória RAM de 16GB, um processador Intel Core i7-9750H 2.60GHz. A versão utilizada do Python foi a 3.10.2, e, além da biblioteca já citada, foram também utilizadas **numpy** e **matplotlib** para manipulação e visualização dos dados.

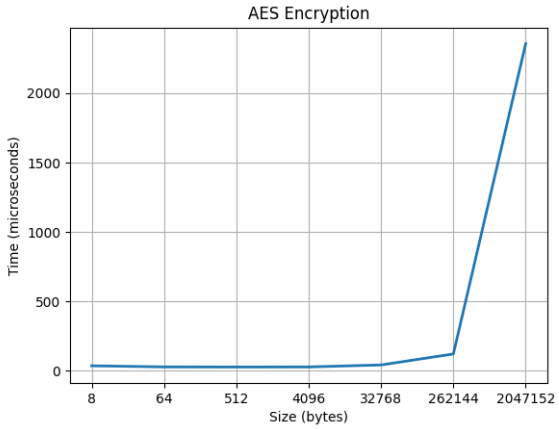
Chapter 2

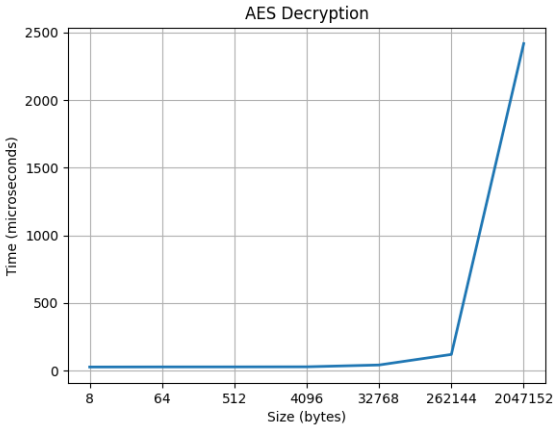
Resultados

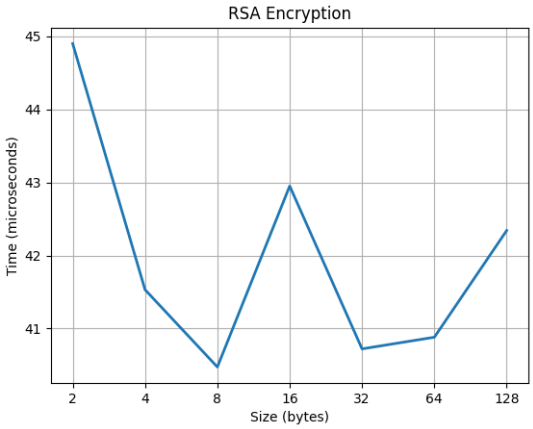
2.1 Obtenção dos Resultados

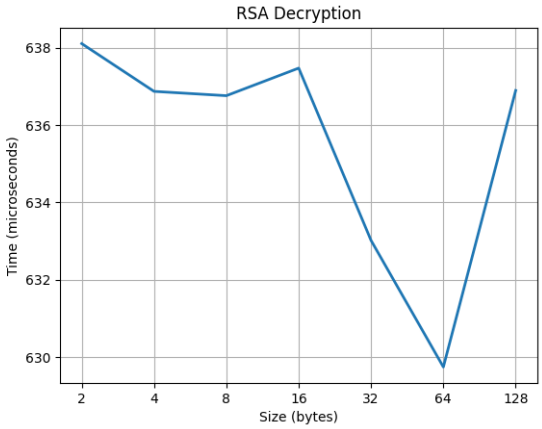
Para garantir uma maior fiabilidade e valores estatisticamente relevantes, o tempo foi medido correndo o algoritmo contra 50 diferentes ficheiros de cada tamanho para cada algoritmo. Este número é controlado pela variável TEST_LIMIT dentro do programa generator.py (que cumpre a tarefa de gerar os ficheiros de teste e fornecer seus nomes para o programa que corre os algoritmos), e é implementado a partir de um simples ciclo "for".
Além disso, cada um dos ficheiros foi corrido 100 vezes contra cada algoritmo, número controlado pela varíavel RUN_TIMES dentro do programa processor.py, que é utilizada na função 'timeit' do Python. Assim, os valores finais são as médias de ambos os processos, o que produz um resultado mais fiável e próximo do real.

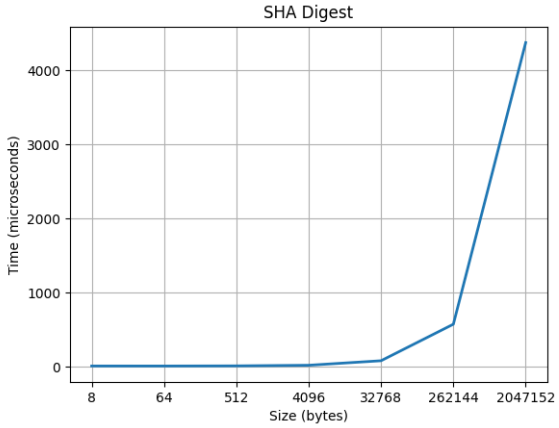
2.2 Gráficos


AES Encryption
Time (microseconds)
0
500
1000
1500
2000
8
64
512
4096
32768
262144
2047152
Size (bytes)


AES Decryption
Time (microseconds)
0
500
1000
1500
2000
2500
8
64
512
4096
32768
262144
2047152
Size (bytes)


RSA Encryption
Time (microseconds)
41
42
43
44
45
2
4
8
16
32
64
128
Size (bytes)


RSA Decryption
Time (microseconds)
630
632
634
636
638
2
4
8
16
32
64
128
Size (bytes)


SHA Digest
Time (microseconds)
0
1000
2000
3000
4000
8
64
512
4096
32768
262144
2047152
Size (bytes)

Chapter 3

Análise dos Resultados

3.1 AES

Um dos principais resultados a se notar na análise dos dados a respeito do algoritmo AES é o facto de tanto a encriptação quanto a decriptação serem extremamente similares em tempo de execução. Devido à estrutura do AES, que, em sua essência, utiliza operações XOR, byte-shifts, lookup-tables e multiplicações de matrizes num domínio limitado e fechado, todos os processos de encriptação possuem um equivalente com uma próxima (ou igual) complexidade durante a decriptação. Em posse da chave, ambas as operações são simples e eficientes, tanto a nível de software quanto hardware. Este fato explica a proximidade dos resultados.

Outra análise a ser feita diz respeito ao aumento do tempo com o aumento dos tamanhos dos arquivos. Nota-se, no gráfico, o formato semelhante à uma função exponencial. Em conjunto com a análise de que o eixo X cresce em proporções próximas à logarítmica, é possível concluir um aumento de complexidade muito próximo ao linear, o que vai de encontro ao esperado. Por ser uma cifra de blocos, o AES leva o mesmo tempo para qualquer tamanho de input menor do que um bloco. Ou seja, espera-se uma complexidade O(1) em relação ao tamanho deste tipo de input, que será tratado igualmente como um bloco de 128 bits. Assim, para cada bloco encriptado, leva-se o mesmo tempo, e, ao fim de n blocos, vemos uma complexidade temporal de n * O(1) = O(n).

3.1.1 Encriptação AES vs Hash SHA

Logo em primeira análise, nota-se uma clara semelhança entre os gráficos de tempo para a encriptação AES e a digestão do algoritmo SHA-256. Ambos com uma curva exponencial em um gráfico logarítmico, o que indica complexidade temporal linear. Entretanto, nota-se que o hashing do SHA-256 leva quase o dobro do tempo para ficheiros de mesmo tamanho. A diferença neste factor constante deve-se ao simples fato de que os algoritmos de hashing produzem output

de maneira mais lenta do que algoritmos de cifra em "stream", o que, para este propósito, inclui o AES em modo CTR, que possibilita a paralelização do processamento dos blocos.

3.2 RSA

A princípio, os gráficos do algoritmo RSA parecem não seguir um padrão relacionado ao tamanho. Entretanto, isto se deve ao facto do RSA possuir um limite para o tamanho da mensagem (que depende do tamanho da chave). Como os tamanhos dos ficheiros utilizados eram muito pequenos e relativamente muito próximos entre si, as diferenças nos tempos de execução para tamanhos diferentes de ficheiros se deve à pequenas inconsistências e mínimas diferenças na execução do programa, no acesso à memória e outras nuances ligadas ao software e ao hardware. A grande conclusão, entretanto, conecta-se à extrema diferença entre o tempo de encriptação e o tempo de decriptação, de um factor de mais de 10x, com a encriptação por volta dos 40 microssegundos e a decriptação por volta dos 630.

3.2.1 Encriptação vs Decriptação RSA

A grande diferença nos tempos de encriptação e decriptação do algoritmo RSA pode ser facilmente explicada ao se analisar o funcionamento deste. Durante o processo de encriptação, a mensagem, transformada em um número, deve ser levantada a um expoente público **e** pequeno, que, no caso do programa, é 65537, um número de 17 bits, e depois dividido por um **n** também público, operação cujo módulo é a mensagem cifrada. Durante a decriptação, entretanto, a mensagem cifrada deve ser levantada a uma chave privada **d**, que é um número diversas vezes maior do que **e**, para então ser também dividido por **n**. No caso do programa, essa chave é um número de 2048 bits. A enorme diferença de tamanho entre estes dois números é o que explica a grande variação de tempo entre a execução dos algoritmos.

3.2.2 Encriptação RSA vs AES

Como mencionado anteriormente, o algoritmo do AES leva um tempo que cresce linearmente com a quantidade de blocos da mensagem, independentemente do tamanho da chave. Isto acontece porque o envolvimento da chave no algoritmo é reduzido a, praticamente, operações XOR, que são extremamente optimizadas nos hardwares. Já no caso do RSA, assumindo que a mensagem já está traduzida a um inteiro M, é necessário realizar a exponenciação deste inteiro por uma chave pública **e** e então realizar também a operação de módulo por outro número **n**. Ambas as operações dependem da quantidade de bits em cada número envolvido. Para realizar a encriptação, a complexidade de tempo total é $O(n^2)$, no qual n é a quantidade de bits do maior número envolvido na operação. A análise dessa diferença entre os dois algoritmos torna se complexa devido ao facto

de os ficheiros processados pelo RSA terem um tamanho tão pequeno, com um máximo de 128 bytes contra os 2047152 do AES. Esta limitação é causada pelo tamanho da chave utilizada pelo RSA, e a diferença de tempo pode ser observada com maior facilidade no gráfico da decriptação, onde o algoritmo do RSA leva cerca de 600 microssegundos em ficheiros que são processados quase que instantaneamente pelo AES. Por este motivo, o RSA é comumente utilizado apenas para a geração de chaves simétricas para o uso de algoritmos de criptografia simétricos, como o AES, que são usualmente mais eficientes.

Chapter 4

Conclusão

O grupo notou que, entre os três algoritmos analisados, o mais veloz em puros termos de processamento de input era o AES, o que é de se esperar do algoritmo que tornou-se padrão na maior parte do mundo. Entretanto, todos os três algoritmos são utilizados para situações diferentes, e, em conjunto, determinam procedimentos que permitem proteger dados quase que infalivelmente. Por sua velocidade, o AES é o preferido para a encriptação de blocos de mensagem, mas exige que ambas as partes, receptor e emissor, possuam uma chave idêntica. Para isso, o RSA é utilizado por ser simples e robusto, permitindo partilhar a chave de um modo seguro. Por fim, o SHA, como algoritmo de hashing, é essencial para a garantia da autenticidade de diversas mensagens e é utilizado em diversas áreas da criptografia, permitindo a não-repudiação e, ao mesmo tempo, a privacidade dos usuários.