E-fólio A | Folha de resolução para E-fólio

UNIDADE CURRICULAR: Segurança em Redes e Computadores

CÓDIGO: 21181

DOCENTE: José Henrique S. Mamede

A preencher pelo estudante

NOME: Mário Pedro Capela Rodrigues Carvalho

N.º DE ESTUDANTE: 2000563

CURSO: Licenciatura em Engenharia Informática

DATA DE ENTREGA: 26 de Novembro 2022

TRABALHO / RESOLUÇÃO:

Este relatório tem como objectivo de explicar o trabalho realizado em paralelo, até ter sido alcançado o objetivo final do e-fólio em questão. Este, demonstra assim, o que fora pensado em cada étapa e ponderado, para realizar cada uma dela com sucesso. Na sua concretização, passou pela construção de um projeto Web, (com três ficheiros: main.js, main.html, e main.css), de modo a facilitar a sua consulta, avaliação e comportamento, encontrando-se este disponível em : E-fólio A, com possibilidade de verificação dos documentos no repositório do aluno: https://github.com/MP-C/UaB.segurancaRedesEfolioA que pode ser acedido, desde a data de entrega do respectivo E-fólio.

Assim, antes de se iniciar qualquer parte de código, ou reproduzir as condições necessárias, teve-se em atenção em criar a arquitetura daquilo que foi pedido, tal como:

1 – uma interface de interação com o utilizador;

2 – zona para escrever a mensagem;

3 – zona para visualizar a mensagem criptada;

4 – zona que mostra a chave utilizada;

5 – zona que mostra a mensagem desencriptada;

Ao ser criado cada uma destas zonas, foi tido em conta cada parte e cada função, de modos que, no próprio código foram surgindo comentários (em Português) apropriados à função em questão, sempre com o objetivo de a esclarecer, recorrendo a nomes (em Inglês) indicando o seu objetivo principal.

Quanto à criptografia em si, teve-se o cuidado de diferênciar a simétrica da assimétrica, tanto código implementado, como na apresentação da página e como no relatório redigido.

Cifra Simétrica : Cifra de César

Particularidades : Para a realização desta função, foi tido em conta a referência feita em Stallings, W (2015), secção 2.2 página 25, quanto à cifra de César.

Por ser uma cifra de substituição, que envolve substituir cada letra do alfabeto por uma nova posição ao longo do mesmo alfabeto. No entanto, foi encontrada uma dispersidade na bibliografia encontrada, quando ao número de posições que poderiam ser alteradas para a sua transposição (Williams K., 2021). É conhecido também que este tipo de encriptação se torna facilmente descodificada por força bruta, dado que a sua linguagem é facilmente reconhecida pela constituição de palavras, espaços, e acentuação (em função da língua), além de apenas permitir uma transposição ao longo do mesmo abcedário. Isto é, para 25 letras apenas se tem 25 chaves possíveis, se se for alterando na sua encriptação, sistematicamente por mais um valor/ posição acima do anterior

utilizado. Podendo-se encriptar com mais uma posição : a= b, ou a = c, e assim sucessivamente. Com isto, tal como referido por Samuele (2020), existindo várias possibilidades com diferentes complexidades de para implementar um código em JavaScript, para evitar plágio, ideias e implementar algo diferenciado que permitisse : uma implementação com grande número de caracteres, com várias possibilidades de caracteres alfabetos e especiais, implementou-se uma nova função, um pouco mais evoluída. Esta função, tem como objectivo permitir a encriptação de parágrafos longos, que incluí esconder o espaço. O que torna a mesma transformação de "Cesar" ligeiramente mais complexa, mas seguindo o mesmo princípio. É claro, que poderia apenas ser feita a transformação simples até como indica Hovhannisyan (2022), no entanto, perder-se-ia toda a reversibilidade da mensagem para se obter a original.

Para efeitos de demonstração de várias possibilidades segundo este método, utilizaram-se três maneiras para transformar os caracteres da mesma mensagem, em sentidos ou posições diferentes. Um para os caracteres especiais (com uma posição a menos), outro para as letras maíusculas (com um avançar de duas posições), e outro para as letras minúsculas (com um avançar exagerado de 6 posições). Procedeu-se também a uma contagem de transformações para cada tipo, que fora demonstrado na zona resultados da Chave utilizada. Para a contagem de posições, recorreu-se à tabela ASCII, para enumerar cada caracter.

Resultado: criando funções por partes, foi tido em conta cada transformação, e portanto, criou-se uma fase de controlo, para informar o utilizador sobre o que deve fazer, e procedendo da mesma maneira, no fim de cada fase (ou sobre a forma de "popUp", ou com introdução de mensagem na zona específica do HTML). Uma zona de leitura do input inserido pelo utilizador, de modo a que cada caracter fosse lido de forma continua, alterado sob a forma de "charCodeAt", para se obter o seu código ASCII, e lhe fora imposta a transposição indicada (para especiais, maísusculas e minúsculas), guardando numa variável única. Assim que este processo tenha terminado, esta variável que continha a mensagem codificada em ASCII é convertida novamente para o abcedário. Assim que esta fase é resolvida, a mensagem, a chave utilizada e a indicação do próximo passo, é indicada no texto de interação.

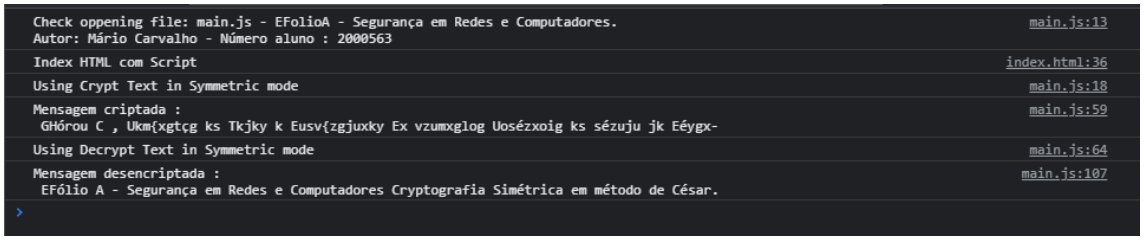
Dificuldades: foram encontradas duas dificuldades maiores durante a implementação, quanto à condição do espaço entre palavras e sinais de pontuação. Conseguindo-se contornar estas duas dificuldades com a transposição para indexes inferiores, e assim, se conseguir percorer todos os valores, sem cair nos valores alfabetas.

Testes: - texto claro e desencriptado: "*EFólio A - Segurança em Redes e Computadores Cryptografia Simétrica em método de César.".*

- texto encriptado: “*GHórou C , Ukm{xgtçg ks Tkjky k Eusv{zgjuxky Ex• vzumxglog Uosézxoig ks sézuju jk Eéygx-*”

- chave: 2 especiais e números, moveram-se "-1" posições; 58 minúsculas, moveram-se "+6" posições; 9 maiúsculas, moveram-se "+2" posições;

Para efeitos de garantia do que se passa em cada momento, a informação pertinente encontra-se diretamente no browser, quanto ao ficheiro que está a ser usado, ao autor e ao resultado final para cada encriptação / desencriptação.



Cifra Assimétrica: Rivest-Shamir-Adleman (RSA)

Particularidades: para a concretização desta cifra, optou-se pela implementação do algoritmo RSA, em que os conceitos teóricos assentam em Stallings, (2015), secção 9.2, mas também consolidados graças à leitura paralela de artigos em GeeksForGeeks (2020, 2021, 2022), com particular atenção a Just Cryptography (2022).

Neste caso, consiste em criar para cada utilizador um par de chaves (privada e pública), que recorre a um processo de transformações com recurso a números primos, (“p”, e “q”, normalmente aleatórios), que são na realidade a base para toda restante parte do cálculo estabelecido. Passando a explicar: com estes, calcula-se n (com n=p x q), e de seguida a função totiente (formúla matemática que assenta no princípio das congruências e módulos, que garante que a partir de um, se possa calcular o outro, desde que e somente quando, ambos os números são divisíveis um pelo outro. Sabendo que, para isso se tem de recorrer ao Máximo Divisor Comum e ao cálculo do seu inverso (modulo). Como se trata de números primos, e segundo o princípio do Algorítmo de Euclides, através de sucessivas divisões, obtendo um resto igual a 1, se garante que ambos (“p”, e “q”,) são primos. E portanto, o processo de criação de chaves torna-se único, a cada criação de pares. Pois, sabendo que, estes afetam uma mensagem (transformada em dígitos por conversão, da mais uma vez, tabela ASCII), toda a mensagem passa a ser um conjunto de cálculos que a transformam numa especie de mensagem encriptada. Esta mensagem, é no entanto encriptada de forma muito particular, pois para o fazer recore à chave pública, e para a desencriptar, recorre à chave privada. Garante assim, que, se precisa sempre de duas coisas entre as três (chave pública, privada e mensagem encriptada). Na realidade, recomenda-se a utilização de um mínimo de 2048-bits, para garantir que ao ser suficientemente grande para se conseguir decifrar, torna o processo por força bruta mais complicado e lento.

Simultâneamente, não é o recomendado para efeitos de transmissão de informação sensível ou “big data”. Para efeitos deste e-fólio, recorreu-se a números primos mais baixos (para garantir a possibilidade de verificação de resultados mais facilmente), mas com um processo de transformação para um texto claro mais reduzido.

Resultado: Assim, num processo mais matemático, mas o qual se recorreu para criar as funções no código foram os passos seguintes :

1) p=11, com q =17 (números primos inseridos no código); n= p x q = 11 x 17 = 187;
2) em que, ϕ(pq) = (p-1)(q-1), que e = (11-1) * (17-1) = 160. Procedendo ao cálculo do máximo divisor comum (para “gcd” no código implementado), a partir de um valor, também número primo: e=7, garante-se que mdc(160,7)=1 e que 1<7<160. Mas, o número de “e” que é encontrado pode eventualmente não respeitar as condições e ser coprimo de ϕ(n), a menos que ϕ(n), via mdc e sofrendo comparação, num ciclo while até encontrar o primeiro coprimo de ϕ(n) que seja menor que ϕ(n), até que surga um número que satisfaça estas duas condições.
3) Por este meio, pode-se obter a chave privada {23,187} e a chave publica {7,187}. Por questões de segurança, “p”, “q” e “d”, devem ser mantidos em privado.

O código implementado passa então, pela construção de várias funções tais como:

a) A principal, que é ativada assim que o utilizador clica em encriptar, que chama de maneira sequênciada cada outra função, necessária para a construção do resultado final. Onde recolhe ao texto claro, o transforma, e devolve a mensagem encriptada;
b) Função que gera a chave publica, associada à função do máximo divisor comum;
c) Função que gera a chave privada, associada à função do inverso do máximo divisor comum;
d) Função que encripta a mensagem;
e) Função que desencripta a mensagem;
f) Para garantir que o processo é realizado de forma correta, a função de desencriptar recorre ao que já fora calculado anteriormente, para apenas transformar a mensagem uma vez mais, e devolver a mensagem inical desencriptada;

Dificuldades: na transformação do texto claro para texto encriptado, passando no alfabeto, dada a limitação e passos sequênciados que a linguagem JavaScript obriga, garantindo que a mensagem é desencriptada corretamente. Mesmo que que a mensagem intermediária apresente caracteres estranhos (não se recorre a node.JS).

Testes: - texto claro e desencriptado: “*EFolio A - Seguranca em Redes e Computadores Cryptografia Simetrica em metodo de Cesar.”*

- texto encriptado: “*‰• ’ • W¯W±W¬vV M WvŽWž v› vQ WvWC• Ž` I› • v QWCB`I• Vª W¬ Ž vI WvŽWŽvI• › • W› vWCvQ*”.

- chave pública: 13 ; chave privada: 37.

Referências:

ASCII table(<unknown>).*ASCII table*.Retrieved from:https://www.asciitable.com/

GeeksForGeeks, (2022). *RSA Algorithm in Cryptography.* Retrieved from: https://www.geeksforgeeks.org/rsa-algorithm-cryptography/.

GeeksForGeeks, (2021). *RSA Algorithm using Multiple Precision Arithmetic Library*. Retrieved from: https://www.geeksforgeeks.org/rsa-algorithm-using-multiple-precision-arithmetic-library/?ref=rp

GeeksForGeeks, (2020). *How to solve RSA Algorithm Problems?* Retrieved from: https://www.geeksforgeeks.org/how-to-solve-rsa-algorithm-problems/?ref=rp/.

Just Cryptography, (2020). *How to generate RSA key pairs*. Retrieved from: https://justcryptography.com/rsa-key-pairs/

Hovhannisyan (2022), *Implementing the Caesar Cipher in JavaScript*. Retrieved from: https://www.aleksandrhovhannisyan.com/blog/caesar-cipher/

Stallings, W. (2017). *Cryptography and Network Security* (7 (Global Edition) ed.). Harlow, England: Pearson Education Limited.

Stallings, W (2015). *Criptografia e segurança de redes: princípios e práticas*; tradução Daniel Vieira; (6. ed.). São Paulo: Pearson Education do Brasil.

Williams K., (2021), *Caesar Cipher Solution in JavaScript,* Retrieved from: https://keithwilliams-91944.medium.com/caesar-cipher-solution-in-javascript-d8221984d61.

Samuele (2020), *How To Code a Caesar Cipher in JavaScript.*Retrieved from: https://blog.stranianelli.com/how-to-code-a-caesar-cipher-in-javascript/