





Relatório Breaking Enigma

Trabalho realizado:

1191830 Rui Marques

Novembro, 2021

**Docente(s)/Orientador(es)**

Marcelo Santos (MEF)

Vladir Vicente (VLV)

**Unidade Curricular**

Segurança de Aplicações e Dados

Indice

[1. Introdução 3](#_Toc90136403)

[2. Desenvolvimento 4](#_Toc90136404)

[3. Programa 4](#_Toc90136405)

[4. Tabela de Figuras 6](#_Toc90136406)

[5. BIBLIOGRAFIA 7](#_Toc90136407)

# Introdução

O objetivo deste trabalho foi criar um programa que atacasse uma “hash” com o auxílio de uma “wordlist” e “plugboard” fornecidas.

Essa “hash” correspondia a uma das palavras fornecidas na “wordlist”, misturada com um “salt” de 2 caracteres de ‘A’ até ‘M’ consecutivos, que poderiam aparecer no início ou final da palavra e misturada com uma versão da Enigma[[1]](#footnote-1) . Esta versão da Enigma consistia na troca de caracteres mediante a “plugboard” fornecida, um sistema de rotores, calculados em função do índex da letra(i), uma rotação(r) e um shift(s) ( i\*s+r).

# Desenvolvimento

Durante o desenvolvimento do programa, realizei duas versões do mesmo, a versão escolhida (A), e outra (B) que, encontrava as palavras com o nosso R e/ou S diferentes. Tendo em conta que a verdadeira enigma funcionava mediante o fornecimento diário das configurações, para que todas as máquinas pudessem comunicar, optei pela versão A que obtém os resultados de R e S pretendidos.

A diferença entre as duas é que a versão B utiliza a palavra e ambos os “salts” para todo o processo, ou seja, tomemos como exemplo a palavra ANIMAS e o “salt” AS, na versão B teremos ‘ASANIMASAS’ a ir para a “plugboard”, “rotores” e “plugboard” novamente sendo separada em ‘ASANIMAS’ e ‘ANIMASAS’ apenas na fase de encriptação, isto faz com que nos rotores o índex das letras das palavras com “salt” no fim seja alterado, o que faz com que a palavra seja encontrada, mas com configuração diferente. Com esta versão encontrei a palavra em 5.22 minutos. Já na versão final consegui reduzir este tempo para 4.01 minutos, sendo que com um algoritmo mais simples como MD5 este tempo é pouco mais de 1 minuto, o que o programa faz é passar cada palavra pela plugboard e rotor apenas uma vez, e depois vai fazendo esse processo com cada salt e no final junta ambos. Uma vez que as palavras são todas do mesmo tamanha, é desnecessário cada vez que mudamos de salt fazer novamente o processo todo, visto que a palavra com o mesmo rotor e shift vai acabar sempre igual. Também só passamos a palavra uma vez pelo primeiro plugboard. Caso o shift e o rotor sejam 0 também é desnecessário fazer o processo pois a palavra fica igual.

# Programa

O programa roda com 3 argumentos, sendo o 1º a hash a encontrar, o 2º a plugboard e o 3º o txt como a lista de palavras a “correr”.

Caso não sejam introduzidos 3 argumentos o programa avisa e encerra.

O algoritmo de Hash é calculado em função do tamanho da String introduzida como argumento 0. O programa verifica que a String apenas contém letras e numeros e depois mediante o tamanho aplica o algoritmo correspondente (128=SHA-512, 64=SHA-256, 32=MD5).

A plugboard terá que ser recebida no formato char:char,char:char, podendo os caracteres terem ou não pelicas. Só aceita letras como valores, no caso de ser introduzido um numero o programa avisa do erro e encerra, caso sejam inseridos 2 caracteres seguidos, apenas o primeiro é aceite e caso não seja inserido nada é ignorado.

Por fim o ficheiro com a lista de palavras terá que ser em formato txt e estar no dirétório do ficheiro java. Não é necessário indicar o caminho, mas se for tentado inserir outro caminho o programa encerra, caso seja tentado inserir algo como ./././wordlist.txt o programa automaticamente assume o dirétorio atual e ignora os ‘./’, neste caso se no dirétório atual houver um ficheiro wordlist.txt este é usado.

Em baixo podemos ver testes, feitos com uma lista de palavras reduziada, com os 3 diferentes algoritmos presentes no programa:

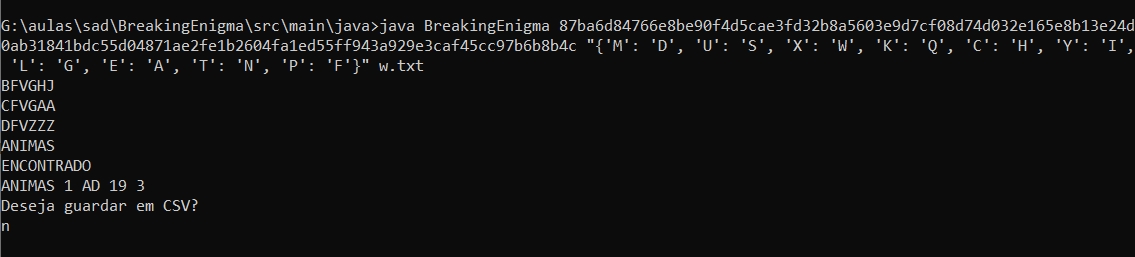


Figura 1 - SHA 512

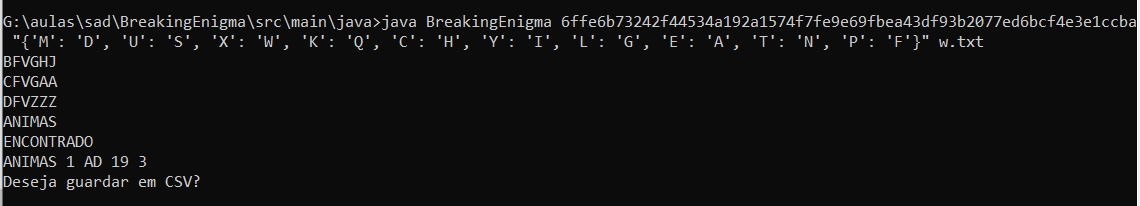


Figura 2 - SHA 256

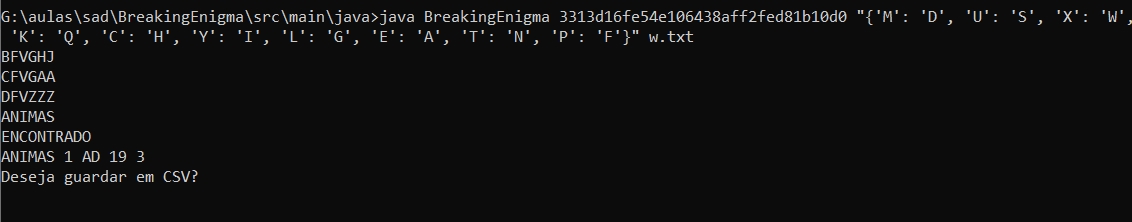


Figura 3 - MD5

No final, ao encontrar a palavra, é possísel guardar os resultados em CSV (password, Salt , Posição do Salt, Rotação, Shift, algoritmo de hash e hash).

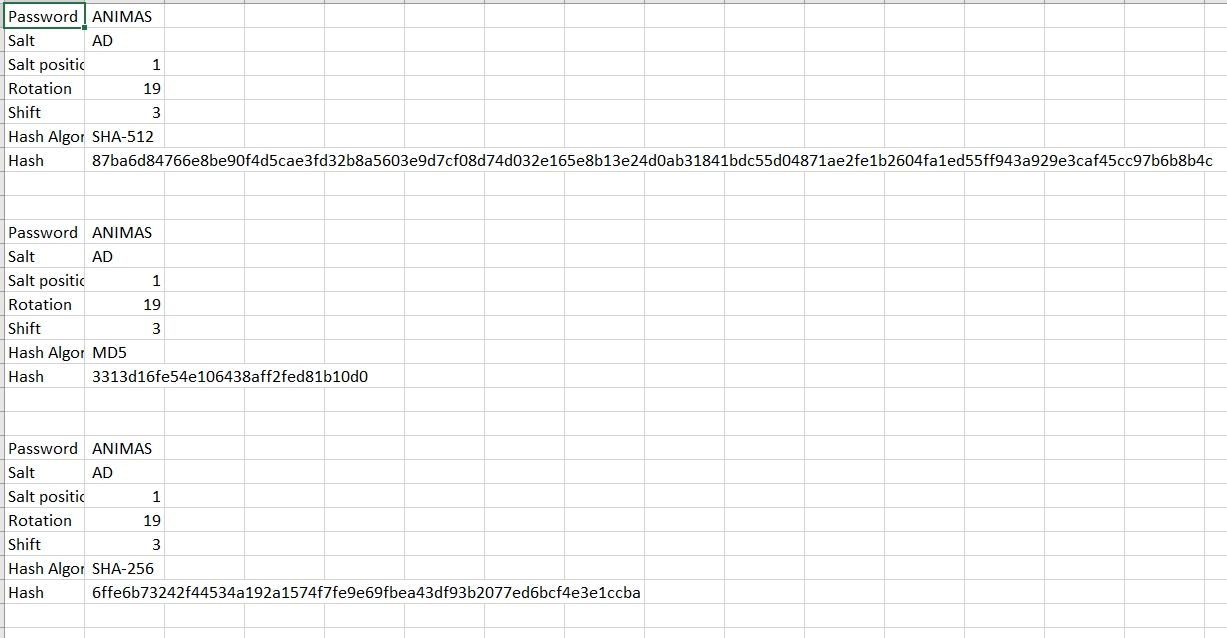


Figura 4 - Ficheiro CSV

# Tabela de Figuras

[Figura 1 - SHA 512 5](#_Toc90136335)

[Figura 2 - SHA 256 5](#_Toc90136336)

[Figura 3 - MD5 5](#_Toc90136337)

[Figura 4 - Ficheiro CSV 5](#_Toc90136338)

# BIBLIOGRAFIA

*How to Compile and Run Java Program from Command Prompt*. (2014, April 11). YouTube. https://www.youtube.com/watch?v=zBF1M8dTftk

*Java Command Line Arguments - Javatpoint*. (n.d.). Www.Javatpoint.Com. https://www.javatpoint.com/command-line-argument

*Online Tools*. (n.d.). Online Tools. https://emn178.github.io/online-tools/index.html

Sinha, A. (2015, October 12). *How to hash a password with SHA-512 in Java?* Stack Overflow. https://stackoverflow.com/questions/33085493/how-to-hash-a-password-with-sha-512-in-java

Wikipedia contributors. (2021, November 29). *Enigma machine*. Wikipedia. https://en.wikipedia.org/wiki/Enigma\_machine

1. https://pt.wikipedia.org/wiki/Enigma\_(m%C3%A1quina) [↑](#footnote-ref-1)