UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"

FCT – FACULDADE DE CIÊNCIAS E TECNOLOGIA
DMC – DEPARTAMENTO DE MATEMÁTICA E COMPUTAÇÃO
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

JUAN CARDOSO DA SILVA - 171257138

SEGURANÇA DA INFORMAÇÃO

ATIVIDADE 3 – RELATÓRIO DE PROGRAMA/CÓDIGO – HASHING MD5 E SHA1

Presidente Prudente, 18/08/2022

Sumário

Introdução

O objetivo desse relatório é demonstrar a capacidade do programa desenvolvido em python, de seguir os requisitos dos algoritmos de encriptação hash, como também ajudar a expandir o conhecimento sobre esses assuntos.

Execução do programa

Foi utilizado Python 3.10.6 para o desenvolvimento de ambos programas, o encriptador (main.py) e o "cracker" (brute_force.py)

A execução do main.py segue como abaixo:

```
[py main.py -sha1 "abcd"] ou [py main.py -md5 "abcd"] sem os [ ]
```

Também é possível passar arquivos de diversos tamanhos para cálculo das hash

```
[py main.py -sha1 -file path] ou [py main.py -md5 -file path] sem os [ ]
```

A execução do cracker segue como:

```
[py brute_force.py -sha1 "abcd"] ou [py brute_force.py -md5 "abcd"] sem os [ ]
```

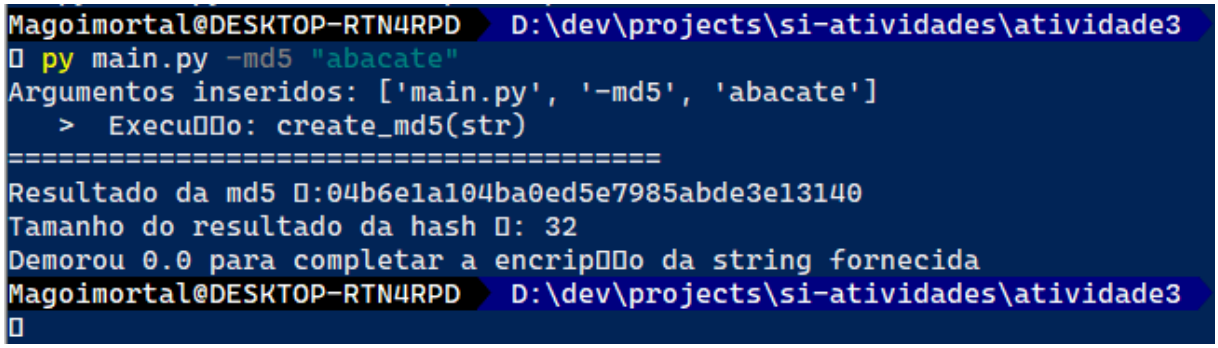
Como utiliza pool bits para realizar a quebra de hash, não foi possível realizar o cracking da hash do arquivo devido a quantidade de arquivos necessários e o poder computacional necessário para calcular todos.

Tamanho de entrada de variável

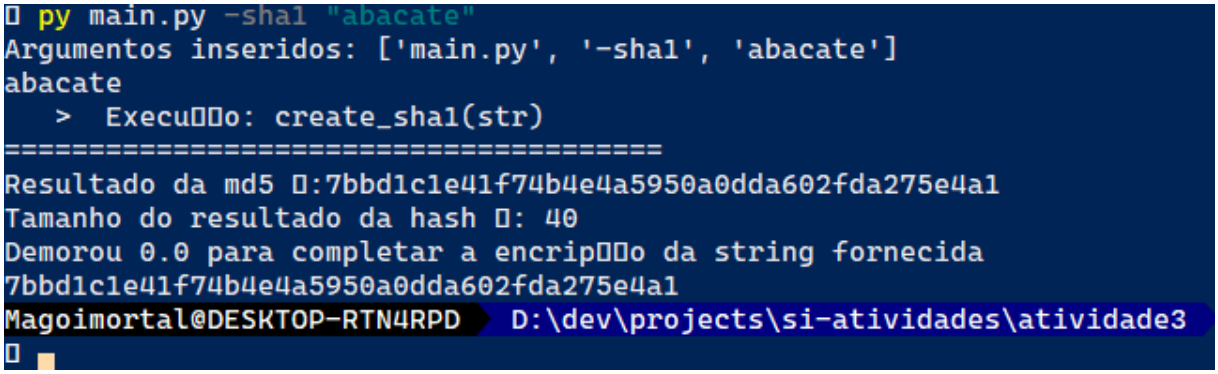
Ambos algoritmos no programa aceitam tamanhos quaisquer de entrada d variável, lembrando que a entrada maior, resulta em um calculo mais demorado e por consequência, o digest demora mais também.

Tamanho de saída fixa

A saída do MD5 é sempre fixada em 128 bits



A saída do SHA1 é sempre 160 bits de saída



Eficiência

SHA, neste caso SHA1 foi criado para ser melhor e arrumar vulnerabilidades do MD5, entretanto o cálculo não é tão rápido quanto um MD5, já o MD5 possuí a vantagem da velocidade e a vulnerabilidade pode ser trata com o método de "salting" nas hashes, entretanto, não garante que um hacker com acesso a hash não consiga obter a chave se ele souber como o salting foi feito, sendo mais um método de mitigação

Resistencia a preimage

Não existiu ocorrências (tanto da MD5 quanto da SHA1) aonde dado um x e um y o processo de hashing resultou no caso $H(x) = y$.

Resistencia a segunda preimage

Não ouve casos aonde dado um específico m e calcular o $H(m) e\ H('m)$ e depois fazer a comparação $H(m) == H('m)$, não ocorreu ocorrência de True no programa.

Resistencia a colisão forte

Não houve ocorrências de colisão forte nas execuções dos algoritmos, devido ao uso do pool de bits no MD5, no SHA1, ao fazer a exaustão de hash, não ocorreu colisões.

Pseudoaletoriedade

A MD5 utiliza pseudoaleatoriedade com constantes de números primos, neste caso chamando no código de s[64] e os pseudonúmeros, também constantes, t[64], eles são utilizados como um "pool de bits" para serem escolhidos na formação da MD5.

```
# tabela de constantes de números primos para ser utilizado no algoritmo.
s = [7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 5,  9, 14,
20, 5,  9, 14, 20, 5,  9, 14, 20, 5,  9, 14,
  20, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 6, 10, 15,
21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21]

# tabela de constantes de pseudo números primos para ser utilizado no
algoritmo.
t = [
    0xd76aa478, 0xe8c7b756, 0x242070db, 0xc1bdceee, 0xf57c0faf,
    0x4787c62a, 0xa8304613, 0xfd469501, 0x698098d8, 0x8b44f7af,
    0xffff5bb1, 0x895cd7be, 0x6b901122, 0xfd987193, 0xa679438e,
    0x49b40821, 0xf61e2562, 0xc040b340, 0x265e5a51, 0xe9b6c7aa,
    0xd62f105d, 0x2441453, 0xd8a1e681, 0xe7d3fbc8, 0x21e1cde6,
    0xc33707d6, 0xf4d50d87, 0x455a14ed, 0xa9e3e905, 0xfcefa3f8,
    0x676f02d9, 0x8d2a4c8a, 0xfffa3942, 0x8771f681, 0x6d9d6122,
    0xfde5380c, 0xa4beea44, 0x4bdecfa9, 0xf6bb4b60, 0xbebfbc70,
    0x289b7ec6, 0xeaa127fa, 0xd4ef3085, 0x4881d05, 0xd9d4d039,
    0xe6db99e5, 0x1fa27cf8, 0xc4ac5665, 0xf4292244, 0x432aff97,
    0xab9423a7, 0xfc93a039, 0x655b59c3, 0x8f0ccc92, 0xffeff47d,
    0x85845dd1, 0x6fa87e4f, 0xfe2ce6e0, 0xa3014314, 0x4e0811a1,
    0xf7537e82, 0xbd3af235, 0x2ad7d2bb, 0xeb86d391
]
```

SHA1 não utiliza pseudoaleatoriedade de números para geração da sua hash, seus sucessores como SHA256 utilizam.