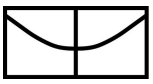

Universidade de Brasília

Instituto de Ciências Exatas
Departamento de Ciência da Computação

Trabalho 1

Segurança Computacional 1

Guilherme Braga Pinto | 170162290
Gabriel Carvalho Moretto | 150154917

Brasília
2021

Introdução

Este trabalho tem por objetivo dar suporte à quebra de uma Cifra de Vigenère, e também implementar a cifração e a decifração da mesma. A quebra da cifra teve a maior automatização possível, porém é necessário que o usuário siga as instruções dadas pelo terminal, e também que sejam respeitadas regras relacionadas ao que o programa processa ou não. Por conta da análise de frequência de letras, acentos e pontuações não foram levados em consideração. Todavia, o programa aceita espaços, que serão conservados porém ignorados no deslocamento letra a letra. Letras também são tratadas todas como sendo letras minúsculas.

Em relação à quebra da cifra, a frequência das letras do alfabeto da língua Portuguesa e da língua inglesa foram retirados da Wikipédia.

O programa se divide em:

- *atacar.py*
- *cifrar.py*
- *decifrar.py*
- *freq.py*
- *gerador_chave.py*
- *salvar.py*
- *main.py*

O programa main.py é o principal, e chama funções de cifrar mensagem, decifrar cifra, salvar resultado em arquivo de texto e atacar cifra sem chave. As funções de cifrar e decifrar usam a função de gerador de chave para fazer o *keystream* corretamente. A função de ataque chama a função de análise de frequências. Utilizou-se apenas a biblioteca *time* para gerar nomes únicos ao se salvar o resultado de uma execução do programa. O programa pode ser executado com:

python3 main.py

Implementação

A implementação do processo de cifrar ocorre após a criação da *keystream*, que é a repetição da key de tamanho igual à mensagem. Para cada letra na mensagem, é executada a soma da letra equivalente na string da chave, com módulo de 26 aplicado para que, por exemplo, a soma de “z” com o número “1” retorne a próxima posição possível, a letra “a”. O processo de decifrar ocorre de maneira exatamente oposta, com a subtração da cifra dada com a *keystream* gerada.

Exemplo (frase do Guia do Mochileiro das Galáxias, de Douglas Adams):

Mensagem: *existe uma teoria que diz que se um dia alguem descobrir exatamente para que serve o universo e por que ele esta aqui ele desaparecera instantaneamente e*

sera substituido por algo ainda mais estranho e inexplicavel existe uma segunda teoria que diz que isso ja aconteceu

Key: *DOUGLAS*

Cifra: *hlcyee mpo nkzrad eok oir tiy yp ue gwu gwgmha xkdcgefcx pxswogkytw solg buw vslbp o mqwpkcsg h dix buw hzy kdts deoo plw gsmgaajhqyxl ifvhuteafhogkytw h gyxl smegnoeuagc juc adjc uoyds pocy psluohnz e aqsrvwiudjyr pxavhy axa khuotoa lhclol qmh rcf buw lgmu ua sfchzpcwx*

Esse resultado está de acordo com outras cifras geradas por sites de terceiros. O programa feito para este trabalho também é capaz de recuperar com sucesso a mensagem dada a key e a cifra, comprovando o funcionamento.

A parte mais elaborada está na parte do ataque. Primeiro se analisa a repetição de vários grupos de 3 letras da cifra (logo, espaços são desconsiderados tanto ao se escolher letras como para contar distâncias entre repetições) e se determina os números pelos quais aquela distância encontrada é divisível. E, por exemplo, se é muito recorrente valores como “7”, “8” e “5”, esses possíveis tamanhos de key devem ser priorizados em cima de valores como “2” e “3”, que são comuns porém pequenos para uma key real. Essa análise funciona por conta do fato de que as frequências em grupos separados por tamanho “x” de key é preservado, apenas com letras alteradas.

Após a escolha de um tamanho de key realista, e determinada qual a língua sendo usada no ataque (o que influencia nas frequências originais do alfabeto), se compara a frequência das letras daquela posição “y” da key de “x” em “x” sendo esse o tamanho da key. Dessa forma, analisa-se sempre uma letra que foi deslocada com um mesmo valor. Comparando as frequências recuperadas da cifra com as frequências originais, o programa sugere quais letras têm mais chance de representar aquela posição. Tentando quebrar o exemplo anterior, as letras que realmente representam a *key* original aparecem como sugestões, em 6 de 7 posições do tamanho da key correta. Na análise das repetições de grupos de 3 letras, o número “7” é bastante recorrente, sendo o tamanho da *key* “DOUGLAS”.

A implementação foi feita em Python 3 e está disponível no Github e foi entregue junto deste relatório.

Conclusão

A implementação é funcional e obedece aos requisitos propostos de acordo com os testes realizados e disponíveis no relatório, apesar de limitações relacionadas ao tratamento de caracteres que não são letras.