Trabalho de Implementação 1 - Cifra de Vigenère

Ayssa Giovanna de Oliveira Marques
170100065

1 Introdução

A cifra de Vigenère é um método de criptografia que utiliza a substituição polialfabética para codificar um texto. Foi desenvolvida por Blaise de Vigenère durante o século XVI. O método utiliza uma palavra-chave para determinar a substituição de cada caractere do texto original - *plain text*. A palavra é então repetida ou truncada para corresponder ao comprimento do *plain text* e, em seguida, cada letra do texto é substituída pela letra correspondente no alfabeto de substituição - denominado *tabula recta*, consiste em um tabela que mostra o alfabeto deslocado 26 vezes - determinado pela letra da palavra-chave na mesma posição.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figura 1: *Tabula Recta*

Sendo uma extensão da cifra de César, se diferencia de sua antecessora por ser mais segura. Primeiramente por utilizar substituição polialfabética. A cifra emprega múltiplos alfabetos de substituição com base em uma palavra-chave. Isso torna a cifra de Vigenère mais resistente à análise de frequência do que a cifra de César, já que um mesmo caractere no *plain text* pode ser substituído por diferentes caracteres no texto cifrado, dependendo da posição da letra na palavra-chave.

Embora a cifra de Vigenère seja mais segura do que a cifra de César, ela ainda é vulnerável a ataques criptoanalíticos, especialmente quando o *plain text* é longo e a palavra-chave é curta ou possui um padrão reconhecível. Criptoanalistas podem usar técnicas como a análise de índice de coincidência e o teste Kasiski para identificar o comprimento da palavra-chave e, posteriormente, realizar análises de frequência em subsequências de texto cifrado.

2 Ferramentas

Para a implementação da cifra de Vigenère nesse trabalho foi utilizada a linguagem Python.

3 Parte I: Cifrador/Decifrador

3.1 Cifrador

A implementação de um cifrador Vigenère foi dada como mostrado a seguir.

```
def vigenere_cipher(text, key):
    cyphertext = ""
    key_length = len(key)
    key_as_int = [ord(i.upper()) - ord('A') for i in key]
    index = 0

    for letter in text:
        if letter.isalpha():
            offset = ord('A') if letter.isupper() else ord('a')
            cyphertext += chr((ord(letter) - offset +
                key_as_int[index % key_length]) % 26 + offset)
            index += 1
        else:
            cyphertext += letter

return cyphertext
```

A função *vigenere_cipher(text, key)* recebe dois parâmetros: o *text*, que é o *plain text*, e *key*, que é a palavra-chave utilizada no processo.

O resultado da cifra após a aplicação da palavra-chave é armazenado na variável *cyphertext*. Temos que *key_length* guarda o comprimento da palavra-chave, enquanto *key_as_int* armazena uma lista contendo a diferença entre o código ASCII das letras da palavra-chave e o código ASCII da letra 'A'. Essa lista será usada posteriormente para deslocar as letras do *plain text*.

Percorrendo cada caractere do texto original, se este for uma letra (*isalpha()*), o código calcula o deslocamento apropriado baseado na letra da palavra-chave correspondente. Para isso, é verificado se a letra é maiúscula ou minúscula com o método *isupper()*, e o valor ASCII de 'A' ou 'a' é armazenado na variável *offset*. Em seguida, é calculado o novo caractere criptografado, levando em conta o deslocamento, e adicionado à variável *cyphertext*.

```
(ord(letter) - offset + key_as_int[index % key_length]) % 26 + offset
```

Se o caractere não for alfabético, ele é adicionado ao *cyphertext* sem alterações. Ao final do processo, a função retorna o *cyphertext* contendo o texto criptografado.

3.2 Decifrador

A função *vigenere_decipher* é usada para decifrar um texto cifrado com a cifra de Vigenère, usando uma chave fornecida.

```
def vigenere_decipher(text, key, display):
cyphertext = ""
key_length = len(key)
key_as_int = [ord(i.upper()) - 65 for i in key]
index = 0

for letter in text:
    if letter.isalpha():
```

```
            offset = 65 if letter.isupper() else 97
            cyphertext += chr((ord(letter) - offset -
                key_as_int[index % key_length]) % 26 + offset)
            index += 1
        elif display:
            cyphertext += letter

    return cyphertext
```

A função aceita três argumentos: o *text*, que é o texto cifrado, e *key*, que é a chave usada para decifrar o texto; e retorna o texto decifrado.

Assim como em *vigenere_cipher*, também usa a variável *key_length* também é usada para armazenar o comprimento da chave e *key_as_int* é criada para armazenar a representação numérica da chave, onde cada letra é convertida em um número de 0 a 25 (letras maiúsculas). É criada uma variável *index* e inicializada com 0. Ela será usada para iterar pela chave durante o processo de decifração.

Ao iterar sobre cada caractere no *text*, se o caractere for uma letra (maiúscula ou minúscula), a função calculará o caractere decifrado usando a fórmula de decifração de Vigenère:

```
(ord(letter) - offset - key_as_int[index % key_length]) % 26 + offset
```

Nesta fórmula, *ord(letter)* retorna o valor Unicode do caractere, offset é 65 para letras maiúsculas e 97 para letras minúsculas (os valores Unicode de 'A' e 'a', respectivamente), e *key_as_int[index % key_length]* é o valor numérico correspondente da chave. O resultado é então convertido de volta em um caractere usando a função *chr()*.

Se o caractere no texto não for uma letra (por exemplo, um espaço, pontuação, etc.), e o parâmetro *display* for **True**, indicando que a função está sendo usada para mostrar o texto decifrado na sala, ele será adicionado diretamente ao texto decifrado sem modificação.

Após iterar por todo o texto, a função retorna o *plain text*.

4 Parte II: Ataque de recuperação de senha por análise de frequência e método qui-quadrado

A seguir, a estratégia usada para descobrir a senha de uma mensagem cifrada. O código desenvolvido suporta os idiomas inglês e português.

4.1 Limpeza da mensagem cifrada

Remove-se todos os caracteres não alfabéticos da mensagem cifrada e armazene as letras alfabéticas em uma nova mensagem limpa.

4.2 Definindo os parâmetros de ataque

Define-se os parâmetros do ataque, como os comprimentos mínimo e máximo de chave a serem testados e o número de chaves mais prováveis a serem avaliadas.

4.3 Encontrando os possíveis comprimentos de chave

Utiliza-se a função *find_key_lengths()* para encontrar os comprimentos de chave mais prováveis, considerando as estatísticas de frequência de letras na língua em que a mensagem foi escrita.

Letra	Frequência em Inglês	Frequência em Português
A	0.08167	0.14634
B	0.01492	0.01043
C	0.02782	0.03882
D	0.04253	0.04992
E	0.12702	0.12570
F	0.02228	0.01003
G	0.02015	0.01303
H	0.06094	0.00781
I	0.06966	0.06180
J	0.00153	0.00473
K	0.00772	0.00037
L	0.04025	0.02779
M	0.02406	0.04738
N	0.06749	0.04446
O	0.07507	0.09735
P	0.01929	0.02523
Q	0.00095	0.01201
R	0.05987	0.06530
S	0.06327	0.02613
T	0.09056	0.04630
U	0.02758	0.03698
V	0.00978	0.01574
W	0.02360	0.00037
X	0.00150	0.00281
Y	0.01974	0.00654
Z	0.00074	0.00470

Tabela 1: Frequência das letras do alfabeto em inglês e português

Os possíveis comprimentos da chave da cifra de Vigenère são encontrados por meio de um processo de análise de frequência das letras na mensagem cifrada limpa. O objetivo é encontrar o tamanho da chave que gera uma distribuição de frequência de letras semelhante à distribuição de frequência das letras na língua em que a mensagem foi escrita.

Para encontrar os possíveis comprimentos da chave, utiliza-se um método conhecido como "Índice de Coincidência", que mede a semelhança entre duas distribuições de frequência de letras. O índice de coincidência para uma língua dada é calculado utilizando-se uma grande amostra de texto naquela língua, e serve como um valor de referência para comparação com a mensagem cifrada.

Temos a equação:

$$IC = \frac{\sum_{c=0}^{25} letter\_frequencies[c] * letter\_frequencies[c] - 1}{text\_length * (text\_length - 1)} \quad (1)$$

onde *letter_frequencies[c]* é o dicionário que armazena a frequência da letra na posição c (de 0 a 25) no alfabeto e *text_lenght* é o comprimento do texto cifrado.

O resultado do índice de coincidência é um valor entre 0 e 1, que indica o grau de semelhança entre a distribuição de frequência de letras na mensagem cifrada e a distribuição de frequência de letras na língua em que a mensagem foi escrita. Quanto mais próximo de 1 for o valor do índice de coincidência, maior é a semelhança entre as duas distribuições.

Para encontrar os possíveis comprimentos da chave, o algoritmo testa todos os comprimentos possíveis de chave entre um valor mínimo e um valor máximo pré-definidos, e calcula o índice de coincidência para cada comprimento. Os comprimentos com os três maiores valores de índice de coincidência são considerados os mais prováveis, e são retornados pela função *find_key_length()*.

4.4 Dividindo a mensagem cifrada em blocos

A mensagem cifrada é dividida em blocos pela função *split_nth_letter()*, cada um correspondendo a cada letra cifrada pela mesma letra da chave.

4.5 Encontrando as letras mais frequentes em cada bloco com qui-quadrado

Utiliza-se a função *find_nth_letter()* para encontrar as letras mais frequentes em cada bloco, baseando-se nas estatísticas de frequência de letras na língua em que a mensagem foi escrita. O algoritmo usa o método de qui-quadrado para avaliar a adequação de cada letra do alfabeto em uma determinada posição. Para isso, ele usa as frequências das letras em um idioma específico (inglês ou português) e compara com as frequências das letras resultantes da decifração com cada letra do alfabeto nessa posição.

Assim, para cada letra do alfabeto, a função decifra o texto cifrado usando a letra como chave e depois calcula as frequências das letras no texto decifrado. Em seguida, é calculado o valor do qui-quadrado entre as frequências das letras no idioma selecionado (observadas) e as frequências encontradas na decifração (esperadas). Usando a fórmula em 2 onde O_i é o valor i nas frequências observadas e E_i é o valor i nas frequências esperadas, as letras que geram os valores de qui-quadrado mais baixos são então consideradas as melhores candidatas para aquela posição. O dicionário *chi_values* armazena esses valores para todas as letras do alfabeto e a função retorna a letra com o menor valor de qui-quadrado, que é a melhor candidata para aquela posição.

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i} \tag{2}$$

4.6 Formando as possíveis chave

Para cada comprimento de chave provável encontrado em 4.3, foram "adivinhadas"as possíveis chaves utilizando as letras mais frequentes encontradas em cada bloco, de acordo com 4.5. É utilizada a função *find_key* que recebe dois parâmetros: *language*, que especifica o idioma que deve ser usado na análise estatística, e *ciphertext*, que é o texto cifrado. O algoritmo começa removendo do texto cifrado todos os caracteres que não são letras e armazenando o resultado em *new_ciphertext*. Em seguida, ele define o comprimento mínimo e máximo da chave (*min_key_length* e *max_key_length*, respectivamente) e o número de comprimentos de chave possíveis que devem ser retornados (*top_n*).

Para encontrar possíveis comprimentos de chave, a função *find_key_lengths* é chamada, passando *new_ciphertext*, *min_key_length*, *max_key_length* e *top_n* como argumentos. Assim temos os n comprimentos de chave mais prováveis. Em seguida, para cada comprimento de chave retornado, o algoritmo chama a função *split_nth_letter* e, então, para cada grupo, a função *find_nth_letter* é chamada, passando o grupo e o idioma selecionado como argumentos. Com a letra mais provável de ser a chave para a posição correspondente no grupo, a função constrói todas as chaves possíveis para cada comprimento de chave e as armazena em *possible_keys*. Por fim, retorna uma lista com todas as chaves possíveis encontradas.

4.7 Teste as possíveis chaves

Fica a cargo do usuário selecionar uma das possíveis chaves e decifrar a mensagem cifrada.

4.8 Verifique se a mensagem decifrada faz sentido

O usuário então verifica se a mensagem decifrada faz sentido e se o ataque foi bem-sucedido. Se não, volta para 4.7 e tenta outra possível chave.

4.9 Fim do ataque

O ataque é concluído com sucesso quando a mensagem decifrada faz sentido e o conteúdo original da mensagem cifrada é recuperado.

Referências

[1] Wikipédia, "Cifra de Vigenère." [Online]. Disponível: `https://pt.wikipedia.org/wiki/Cifra_de_Vigen%C3%A8re`. Acesso em: 20/04/2023.

[2] Udacity, "Vigenere cipher." Youtube. Disponível em: `https://www.youtube.com/watch?v=SkJcmCaHqS0`. Acesso em: 06/05/2023., 2015.

[3] B. Veitch, "Cryptography - breaking the vigenere cipher." Youtube. Disponível em: `https://www.youtube.com/watch?v=P4z3jAOzT9I`. Acesso em: 06/05/2023., 2014.

[4] Theoretically, "Vigenere cipher - decryption (unknown key)." Youtube. Disponível em: https://www.youtube.com/watch?v=LaWp_Kq0cKs&t=916s. Acesso em: 28/04/2023., 2015.

[5] P. of Concept, "Cryptanalysis of vigenere cipher: not just how, but why it works." Youtube. Disponível em: https://www.youtube.com/watch?v=QgHnr8-h0xI. Acesso em: 28/04/2023., 2020.

[6] A. Sweigart, *Cracking Codes with Python: Chapter 19*. 2021.

[7] A. Sweigart, *Cracking Codes with Python: Chapter 20*. 2021.

[8] stine, "Cracking vigenère cipher." Medium. Disponível em: https://0xckylee.medium.com/cracking-vigen%C3%A8re-cipher-cee60db3a966. Acesso em: 04/04/2023., 2019.