

**Relatório – Cifra de Vigenère**

Alan Fontenele Véras – 200013335

Welder Cavalcante de Oliveira – 190039582

Dep. Ciência da Computação – Universidade de Brasília (UnB)

1. **Introdução**

Neste projeto, nosso objetivo principal é implementar completamente as operações de codificação, decodificação e análise de vulnerabilidades da Cifra de Vigenère, um método de criptografia histórica que utiliza múltiplas Cifras de César com deslocamentos determinados por uma chave específica. A Cifra de Vigenère é reconhecida por sua resistência a ataques de força bruta devido às chaves de comprimento variável. Estamos enriquecendo nossa implementação com recursos adicionais do curso e pesquisando fontes online confiáveis para garantir a excelência do projeto. Este esforço visa não apenas dominar a técnica, mas também compreender profundamente a importância da criptografia ao longo da história e em contextos contemporâneos.

Através deste projeto, apresentaremos uma abordagem abrangente para codificar, decodificar e explorar vulnerabilidades na Cifra de Vigenère.

1. **Codificação**

Inicialmente, definimos o alfabeto como uma string composta por 26 letras maiúsculas.

# Defina o alfabeto usado na cifra

alfabeto = 'abcdefghijklmnopqrstuvwxyz'

Em seguida, definimos uma função chamada "cifrar", que recebe como parâmetros uma mensagem a ser cifrada e uma chave. Esta função é responsável por percorrer cada caractere da mensagem e realizar uma série de verificações. Primeiro, ela verifica se a letra está presente na string "alfabeto" e, em seguida, executa cálculos para cifrar o caractere. Se a letra estiver na string "alfabeto", a função encontra o índice da letra atual na mensagem e o correspondente na chave. Em seguida, calcula o novo índice usando a Cifra de Vigenère e adiciona a letra cifrada à string que armazena o texto cifrado. Se o caractere não for uma letra, a função mantém o caractere original no texto.

# Função para cifrar uma mensagem

def cifrar(texto, chave):

texto\_cifrado = ''

i = 0

for i in range(len(texto)):

if texto[i] in alfabeto:

# Encontre o índice da letra atual na mensagem e da letra correspondente na chave

indice\_letra = alfabeto.find(texto[i])

indice\_chave = alfabeto.find(chave[i % len(chave)])

# Calcule o novo índice usando a cifra de Vigenère

novo\_indice = (indice\_letra + indice\_chave) % len(alfabeto)

# Adicione a letra cifrada ao texto cifrado

texto\_cifrado += alfabeto[novo\_indice]

i += 1

else:

# Se não for uma letra do alfabeto, mantenha o caractere original

texto\_cifrado += texto[i]

return texto\_cifrado

1. **Decodificação**

Para a decodificação, seguimos a mesma lógica da codificação. Implementamos um loop que percorre a mensagem cifrada e realiza verificações para cada caractere. Se o caractere não estiver presente na string "alfabeto", ele é mantido inalterado no texto decifrado. No entanto, se o caractere estiver na string "alfabeto", procedemos da seguinte maneira:

1. Encontramos o índice da letra cifrada atual na mensagem e o correspondente na chave.
2. Calculamos o novo índice usando a Cifra de Vigenère.
3. Se o resultado for negativo, adicionamos o tamanho do alfabeto ao índice para obter um índice positivo.
4. O caractere correspondente ao novo índice é adicionado a uma string que armazena os caracteres decifrados.

Essa abordagem garante que a mensagem seja decifrada com base na chave e na Cifra de Vigenère, mantendo os caracteres não cifrados inalterados no texto decifrado.

# Função para decifrar uma mensagem cifrada

def decifrar(frase\_cifrada, chave):

texto\_decifrado = ''

i = 0

for letra in frase\_cifrada:

# Verifique se a letra cifrada está no alfabeto

if letra in alfabeto:

# Encontre o índice da letra cifrada atual na mensagem e da letra correspondente na chave

indice\_letra\_cifrada = alfabeto.find(letra)

indice\_chave = alfabeto.find(chave[i % len(chave)])

# Calcule o novo índice usando a cifra de Vigenère

novo\_indice = (indice\_letra\_cifrada - indice\_chave) % len(alfabeto)

# Se o resultado for negativo, adicione o tamanho do alfabeto para obter um índice positivo

if novo\_indice < 0:

novo\_indice += len(alfabeto)

# Adicione a letra decifrada ao texto decifrado

texto\_decifrado += alfabeto[novo\_indice]

else:

# Se a letra cifrada não estiver no alfabeto, mantenha-a no texto decifrado

texto\_decifrado += letra

i += 1

return texto\_decifrado

1. **Ataque** 
   1. **Análise de Frequência de Letras (analisar\_frequencia(sequencia))**

Esta função analisa a frequência de letras em uma sequência de texto. Para fazer isso, ela calcula o Qui-Quadrado de frequências para cada letra do alfabeto, comparando a frequência observada na sequência com a frequência esperada em um idioma específico (português ou inglês). O Qui-Quadrado mede o quão bem as frequências observadas correspondem às frequências esperadas. A função retorna uma letra com base na análise de frequência ou uma sugestão se o usuário não fizer uma escolha.

* 1. **Obtenção de Chave (obter\_chave(mensagem, tamanho\_chave))**

Esta função recebe uma mensagem cifrada e um tamanho de chave suposto como entrada. Ela divide a mensagem em sequências de caracteres, uma para cada posição da chave. Em seguida, utiliza a função analisar\_frequencia para determinar a letra da chave de cada sequência. A função retorna a chave obtida.

* 1. **Cálculo do Índice de Coincidência (calcular\_indice\_coincidencia(sequencia))**

Calcula o índice de coincidência de uma sequência de texto. O índice de coincidência mede a probabilidade de duas letras escolhidas aleatoriamente na sequência serem iguais. Quanto mais próximo de 1, mais provável é que as letras sejam iguais. A função retorna o índice de coincidência.

* 1. **Encontrar Tamanho de Chave Provável (encontrar\_tamanho\_chave(mensagem, tamanho\_maximo=20))**

Analisa a mensagem cifrada para encontrar os possíveis tamanhos da chave de Vigenère. Ela itera por tamanhos supostos de chave e calcula o índice de coincidência médio para cada tamanho. Os melhores tamanhos de chave são aqueles com índices de coincidência médios mais altos. A função retorna uma lista dos melhores tamanhos de chave.

* 1. **Ataque de Vigenère (atacar(mensagem))**

Realiza um ataque de força bruta na mensagem cifrada, tentando encontrar a chave correta com base nos possíveis tamanhos de chave. Filtra caracteres inválidos da mensagem e encontra os tamanhos de chave mais prováveis. Permite ao usuário escolher o tamanho da chave ou usa o tamanho mais provável.

* 1. **Função Auxiliar: Filtrar Caracteres Inválidos (filtrar\_caracteres\_invalidos(mensagem))**

Remove caracteres inválidos da mensagem, mantendo apenas os caracteres que estão no alfabeto definido no código.

1. **Conclusão**

Embora a Cifra de Vigenère seja notoriamente complexa, ela se torna vulnerável quando as chaves de codificação são curtas e repetitivas [Wikipedia, 2023]. Portanto, é crucial optar por chaves longas e diversificadas. Conclui-se que, apesar de ser uma técnica antiga, a Cifra de Vigenère pode oferecer segurança e eficácia na criptografia, desde que sua chave seja robusta e sua implementação seja sólida.

Por meio deste projeto, ganhamos uma compreensão mais profunda da criptografia de cifras. Enfrentamos desafios significativos, especialmente ao lidar com a complexidade dos ataques à cifra. No entanto, nossa jornada nos permitiu explorar e compreender melhor a eficácia da criptografia de cifras, destacando a importância de abordagens seguras e chaves robustas.

**Referências**

[Wikipedia 2023] Wikipedia (2023). Cifra de Viginère. <https://pt.wikipedia.org/wiki/Cifra_de_Vigenère>

[WikiHow 2023] WikiHow (2023). Como Codificar e Decodificar Usando a Cifra de Vigènere. <https://pt.wikihow.com/Codificar-e-Decodificar-Usando-a-Cifra-de-Vig%C3%A8nere>

Cifra de Vigenère. Disponível em: <https://www.youtube.com/watch?v=qVgkytsdmvk>

Vigenere Cipher - Decryption (Unknown Key). Disponível em: <https://www.youtube.com/watch?v=LaWp_Kq0cKs>