Universidade de Brasília

Departamento de Ciência da Computação

Cifra de Vigenère

Relatório sobre implementação

Segurança Computacional 2022/1

João Pedro Correia Nogueira Mota 17/0106144

Brasília
31 de julho de 2022

1 Objetivos do projeto

Este projeto tem por finalidade a automação da cifração e decifração de Vigenère, além da implementação de um possível ataque a uma cifra através de análise de frequência de letras para obter uma chave provável e compatível.

2 Observações importantes

- O sistema foi totalmente desenvolvido utilizando Java 17 LTS.
- O sistema não possui uma interface gráfica. Portanto, deve ser executado e utilizado através de linhas de comando.
- Importante ter cuidado com CLTR-C e CLTR-V nos textos de cifra, quebras de linhas significam o fim da entrada do usuário, e isso pode deixar a interface perdida.
- Se possível, inserir textos de cifras por apenas uma linha do terminal.
- O sistema não possui muitas validações de entrada, portanto, qualquer teste de caractere não válido nas entradas poderá causar uma exceção e a interrupção do programa.
- Qualquer caractere diferente de letra ou "espaço"na cifra será descartado.

3 Estrutura do projeto

A implementação deste projeto possui a seguinte estrutura de arquivos:

1. **VigenereLauncher:** classe na qual o método main está inserido, portanto, torna-se a principal classe do sistema. É a responsável pela entrada e saída de dados, além de fazer o controle dos métodos que deverão ser chamados em cada situação.
2. **VigenereCore:** classe que mantém os métodos de cifração e decifração de mensagens. Quando chamados, ambos retornam uma String de conteúdo cifrado ou decifrado para a classe VigenereLauncher.
3. **VigenereCrack:** classe mais complexa do sistema, tem por responsabilidade apenas a quebra de uma mensagem cifrada. Armazena também uma sequência de métodos auxiliares para a obtenção de uma chave a partir da cifra.

4. **VigenereUtils:** classe auxiliar, armazena métodos úteis no decorrer de todo o projeto, além de guardar também a frequência de letras das línguas especificadas.

4 Principais métodos

O sistema possui quatro métodos essenciais para o cumprimento de suas especificações. Cada método será citado abaixo pelo seu nome original, seguido de sua função.

4.1 Encryt (Cifrador)

Recebe como parâmetro uma mensagem e uma chave, na qual é feito um tratamento para que a chave recebida seja alongada até possuir o mesmo comprimento da mensagem. Em seguida, a mensagem recebida é formatada para que sejam conservados apenas caracteres de letras, se houver algum espaçamento, os mesmos terão suas posições armazenadas em uma lista de inteiros.

Posteriormente, para cada caractere da mensagem já tratada e a partir de seus respectivos valores ASCII, calculamos **(caractere atual + caractere respectivo da chave) % 26**, o resultado corresponde ao caractere já cifrado.

Após a iteração por todos os caracteres da mensagem, é inserido novamente os espaços retirados da mensagem original, e a mensagem cifrada é retornada ao final da função.

4.2 Decryt (Decifrador)

Possui a mesma lógica da função **encrypt**, porém, recebe como parâmetro uma mensagem já cifrada. A partir da chave já prolongada e de cada caractere da mensagem cifrada, calculamos **(caractere atual - caractere respectivo da chave) % 26**, o que corresponde ao inveso do que foi aplicado na função **encrypt**.

O método é encerrado retornado a mensagem decifrada, com todos os espaços inseridos (se existirem).

4.3 GetProbableKeyLenghts (calcula o tamanho provavel da chave)

Recebe como parâmetro apenas uma mensagem cifrada, na qual é repartida em uma lista contendo Strings de tamanho *N (parametrizável no código)*,

originalmente $N = 3$, chamaremos cada String dessa de *n-grama*.

Para cada *n-grama* dentro da lista, é feito uma busca na mensagem cifrada, contando as repetições e guardando as suas respectivas posições. Se o *n-grama* se repetir mais de uma vez, guardamos ele em uma nova lista de *n-gramas* significantes.

Posteriormente, para todos *n-gramas* e suas respectivas posições, procouramos o máximo divisor em comum entre todas elas. O valor que mais se repetir, provavelmete corresponderá ao tamanho da chave da cifra. Neste ponto, a função retorna uma com os prováveis tamanhos da chave, juntamente com a quantidade de números divisores, ordenados dos mais prováveis para o menos provável.

4.4 FindKey (retorna a provável chave da cifra)

Recebe como parâmetro uma mensagem cifrada, um tamanho de chave e um inteiro sinalizando a língua da cifra (Inglês ou Portugês). Importante lembrar que neste ponto já possuímos as informações de frequência de cada letra em cada língua.

Inicialmente tratamos a cifra recebida retirando espaços e caracteres diferentes de letras, e com a cifra já tratada podemos separar os caracteres em grupos a partir do cálculo **(tamanho da cifra / tamanho da chave)**. Feito isso, calculamos também um percentual em comum de letras por grupo **(100 / caracteres por grupo)**.

Para cada posição dentro da provável chave, calculamos o percentual de ocorrência de cada caractere em um grupo de caracteres, e armazenamos em um vetor contendo o caractere e seu respectivo valor. Feito isso, percorremos todas as posições do alfabeto calculando o possível valor de "shift"entre uma letra e outra a partir de menor diferença entre a frequência de letras da cifra e a frequência de letras na língua escolhida.

Após todas as posições da provável chave terem sido preenchidas, a função retorna uma String contendo a sequencia de caracteres prováveis, ou seja, a chave.