Trabalho 1 - Segurança de Sistemas

Octavio Carpes

[1]Pontifícia Universidade Católica do Rio Grande do Sul

octavio.carpes@edu.pucrs.br

Resumo. *Este trabalho contempla a 1ª tarefa da disciplina de Segurança de Sistemas ministrada pelo Prof. Avelino Francisco Zorzo. Nesta atividade devemos implementar um programa o qual deverá ser capaz de decifrar um texto cifrado através da cifra de Viginere.*

1. Introdução

Com o objetivo de aprimorar e aplicar os conteúdos estudados na disciplina, nos foi proposto o desenvolvimento de um programa o qual fosse capaz de decrifrar um código cifrado pela cifra de Vigenere. Nos próximos tópicos serão explicados as instruções de uso, estrutura do código, algoritmos, e uma conclusão.

2. Instruções de uso

Como o programa foi escrito na linguagem Node.js, existem algumas configurações necessárias para que o programa possa ser executado. É necessário ter a ferramenta Node.js instalada no computador, para isso recomendamos acessar o link oficial do Node.js encontrado em https://nodejs.dev/. Após instalar a linguagem em seu computador, recomendamos que o gerenciador de pacotes *yarn*, seja instalado também, para que os pacotes utilizados no programa sejam instalados de acordo com os arquivos de configuração. Para instalar o *yarn* basta seguir o guia de instalação oficial encontrado em https://yarnpkg.com/. Após as configurações realizadas em sua máquina, o usuário pode optar por utilizar o git, para baixar o código fonte encontrado em https://github.com/octaviocarpes/vigenere-cipher, utilizando o comando *git clone https://github.com/octaviocarpes/vigenere-cipher* ou simplesmente baixar o zip do código fonte, também presente no mesmo link. Com o programa baixado e extraído, basta navegar para a pasta onde se encontra o software, através do uso de um terminal, e executar os seguintes comandos:

yarn install
yarn start

que o programa irá começar. É necessário a atenção a uma parte crucial do programa na qual será necessário a interação com ele, para que seja possível decifrar o texto corretamente o usuário deve escolher a opção **A E E E E E A**, mais será explicado ao longo do relatório.

3. Definições e organização do programa

O programa é composto apenas por dois arquivos *index.ts* e *portuguese.ts*. No arquivo *portuguese.ts* existem 3 constantes que são respectivamente o texto a ser decifrado, uma

lista contendo as letras do alfabeto com os índices de coincidência para a língua portuguesa e uma lista contendo todas as letras do alfabeto. Já o arquivo *index.ts* é o arquivo principal do programa, que é responsável pela execução do algoritmo que decifra o texto. Dentro do arquivo *index.js* existem funções as quais, em conjunto, executa o algoritmo que decifra o texto. Primeiramente vale lembrar que estamos decifrando um texto cifrado em Português e estamos utilizando o método de índice de coincidência, adotando o valor do índice para português como 0.072723.
Como citado anteriormente, dentro do *script* existem várias funções que executam o algoritmo de decifragem, estas funções são:

3.1. findKeySize

Esta função é responsável por encontrar o tamanho da chave da cifra, para isso são utilizadas duas funções *calculateCoincidenceIndexForText* e *isThisAgoodIndex*. A primeira calcula o índice de coincidência de qualquer texto e retorna um valor do tipo *float* enquanto a segunda verifica se um índice calculado é um bom chute para usar no cálculo do tamanho da chave. Esta função retorna o tamanho da chave após descobrir o índice de coincidência.

3.2. calculateCoincidenceIndexForText

Nesta função um texto é passado como parâmetro e é feita a contagem da frequência de todas as letras do alfabeto que estão no texto, após isso são feitas operações matemáticas para calcular o índice de frequência do texto passado como parâmetro.

3.3. isThisAgoodIndex

O método *isThisAgoodIndex* recebe por parâmetro um valor de um índice de coincidência e compara ele com o índice da língua portuguesa, que adotamos como 0.072723, caso a diferença seja pequena, esta função retorna *true*, caso o contrário *false*.

3.4. findEachLetterKey

Descoberto o tamanho da chave, agora é necessário descobrir a chave em si, para isso a função, *findEachLetterKey*, transforma a cifra em uma matriz com o número de colunas igual ao tamanho da chave, procura pelo caractere mais frequente em cada coluna e então monta a chave a partir deles.

3.5. transformCipherIntoMatrix

Este método transforma o texto cifrado em uma matriz com o número de colunas igual ao número de caracteres da chave e então retorna a matriz.

3.6. getMatrixColumns

Esta função tem como objetivo montar strings a partir de cada coluna da matriz e retornar uma lista contendo o texto de cada coluna.

3.7. getMostFrequentCharacterOfColumn

Retorna o caractere mais frequente de uma coluna da matriz.

3.8. parseColumnWithCharacters

Dado o texto da coluna e as chaves, esta função é responsável por transformar o texto cifrado pelo texto decifrado na ordem das colunas.

3.9. getMostFrequentLettersForPortuguese

Este método pede para o usuário qual a combinação de letras mais frequentes do português que o usuário gostaria de usar. Esta função existe pois como as a letra **A** e a letra **E** tem um índice de coincidência muito próximo, muitas vezes o programa não vai conseguir decifrar o texto utilizando apenas uma delas. Para resolver este problema o usuário necessita escolher a combinação correta destas letras, que no caso deste programa é [A, E, E, E, E, E, A].

3.10. calculatePaddingForLetter

Esta função calcula o deslocamento entre a letra mais frequente da língua portuguesa e a letra da chave, retornando o entao qual o deslocamento que deve ser utilizado ao realizar o *shift* das letras das colunas da matriz.

3.11. shiftLetter

Função que realiza o deslocamento das letras e retorna o caractere decifrado.

3.12. decryptText

Esta função monta o texto decifrado através da matriz, ou seja, junta todas as colunas da matriz sequencialmente e retornar o texto decifrado.

3.13. begin

A função que começa o script e executa todas as funções necessárias, ela executa todos os passos do algoritmo

1. Calcula o índice de coincidência até encontrar um bom candidato;
2. Escolhe um bom candidato a tamanho da chave;
3. Descobre todas as letras da chave;
4. Transforma o texto em matriz;
5. Para cada coluna da matriz, executa o deslocamento;
6. Imprime na tela o texto remontado;

4. Conclusão

Realizar esta tarefa foi um desafio muito difícil porém recompensador, no início acreditava que não conseguiria traduzir o texto porém o resultado deu certo para o texto cifrado. O programa não é viável para qualquer texto justamente porque tivemos dificuldades de transformar a chave em texto claro. Então o usuário deve ficar tentando todas as possibilidades de chave até conseguir decifrar. Apesar do resultado não ser muito satisfatório, acreditamos que conseguimos compreender o funcionamento de Vigenere e como aplicá-lo em casos de uso, tornando a experiência final melhor. Acreditamos que o trabalho conseguiu atingir o objetivo de aumentar e aplicar os conhecimentos de Segurança de Sistemas em código real.

5. Exemplos do código e funcionamento

```ts
const begin = async () => {
    const keySize = findKeySize(PORTUGUESE_CIPHER)
    const key = await findEachKeyLetter(keySize)
    const cipherMatrix: string[][] = transformCipherIntoMatrix(keySize)
    const columns = getMatrixColumns(cipherMatrix)
    const shiftedColumns: string[] = await parseColumnWithCharacters(columns, key)
    const parsedText = decryptText(shiftedColumns)
    console.log(parsedText)
}
```

Figura 1. Trecho da função *begin*

Figura 2. Imagem mostrando o resultado