

**Segurança de Sistemas**

**Trabalho 1**

**Prof. Avelino Francisco Zorzo**

**Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)**

**CEP 90619-900 – Porto Alegre – RS – Brasil**

Christian Schmidt

{christian.schmidt}@acad.pucrs.br

***Resumo:***

Trabalho de criptoanálise de Vigenere. Este trabalho 1 tem por objetivo o desenvolvimento de uma aplicação *Java*​ para decifrar um texto cifrado. Ao ler um texto cifrado a aplicação consegue descobrir o tamanho da chave, qual é a chave, e decifrar o texto.

# 1 Utilização do programa

Para a utilização da aplicação basta executar em uma IDE. A classe ***Main*** é chamada de ***Game.*** Outra opção é executar no terminal o arquivo SST1.jar através do comando:

***java -jar SST1.jar***

Vídeo demonstrativo do trabalho dentro da pasta: ***Video\_Demonstrativo.mp4***

Link do repositório no github contendo todos os arquivos: <https://github.com/Christian2019/Seguranca-de-Sistemas_T1>

# 2 Desenvolvimento

## 2.1 Game (Main)

Primeiramente ele escolhe o índice de coincidência da língua portuguesa já que o texto cifrado utilizado é o **"Arquivo cifrado (plaintext em Português)"**. Ele então cria uma lista de **AlphabetValue**, que é um objeto que contém uma String (letter) e um int(value). Esta lista contém todas as letras do alfabeto e seus valores. Após lê o arquivo “entrada.txt” através do método ***load()*** dentro da classe **Leitura**. O texto cifrado está então armazenado na variável String texto.

Agora está tudo pronto para começar. O método ***discover\_Key\_Size()*** da classe**Key\_Size** descobre o tamanho da chave. O discovery\_key() da classe Key descobre todas as combinações de chaves possíveis. O decifrar() da Decipher decifra o texto utilizando todas as combinações de chaves encontradas e descobre qual a chave e o texto corretos.

É impresso na tela todos os indicies de coincidência calculados para cada tamanho de chave, todas as combinações possíveis de chaves, a chave correta e os 100 primeiros caracteres do texto claro.

## 2.2 Key\_Size

Um loop testa tamanhos de chave de 1 até 100. O texto então é dividido em substrings em quantidade igual ao tamanho da chave que está sendo testada. Para cada substring é calculada seu indicie de coincidência. Caso seu indicie de coincidência esteja dentro de um intervalo entre o índice de coincidência da língua -0.01 e o índice de coincidência da língua +0.01, o método para e considerada que o tamanho de chave foi encontrado.

## 2.3 Key

O texto cifrado é dividido em n colunas, sendo n o tamanho da chave encontrado. Ele então descobre qual a letra de maior frequência de cada coluna. Na língua inglesa a letra “e” é a letra de maior frequência e fica fácil descobrir a chave. Na língua portuguesa temos um problema, as letras “a” e “e” possuem frequências muito próximas. Dessa forma a letra que mais se repete em cada coluna pode ser “a” ou “e” do texto claro. Para resolver isso foi criado 2 chaves, key e key2. A key considera que sempre a letra de maior frequência é a letra “a”, e resultou na chave “miyrsqe”. A key2 considera que sempre a letra de maior frequência é a letra “e”, e resultou na chave “ieunoma”. Sabemos que a chave correta é uma das combinações possíveis entre a key e a key2. Então o método ***allkeyCombinations()*** cria uma lista com todas as combinações possíveis entre as 2 chaves. O tamanho da lista é de 2^(tamanho da chave).

## 2.4 Decipher

O texto cifrado é então decifrado através do método ***dec()*** para todas as combinações de chaves encontradas. O método ***decifrar()*** verifica qual dos textos decifrados contém a menor frequência de “y”, ”w” e ”k”, já que são letras pouco usadas na língua portuguesa.

O que conter a menor frequência é o texto correto e, portanto contém a chave correta. O resultado que ele encontrou foi a chave: ”**meunome**”.

Primeiras letras do texto decifrado: quemhacincoentaannostivesseacoragemdepublicarumlivrocomoodesumn