Decifrando a cifra de Vigenère

Arthur Henrique Henz[1], **Pablo Miguel Chong Alles**[1]

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
Porto Alegre – RS – Brasil

arthur.henz@edu.pucrs.br, p.chong@edu.pucrs.br

Resumo. *Este relatório descreve uma solução para o primeiro problema proposto na disciplina de Segurança de Sistemas no semestre 2023/2, que trata do desenvolvimento de um programa, que dado um texto cifrado com a cifra de Vigenère, deve encontrar o texto claro. Com isso, é preciso primeiramente descobrir o tamanho da chave que criptografou o texto claro. Assim, é apresentada uma implementação que utiliza o método de Índice de Coincidência para encontrar o tamanho da chave e análise de frequência para encontrar a possível chave que encriptou o texto.*

Abstract. *This report describes a solution to the first problem proposed in the Systems Security course in semester 2023/2, which deals with the development of a program that, given a text encrypted with the Vigenère cipher, must find the plain text. To do this, it must first find out the size of the key that encrypted the plain text. Thus, an implementation is presented that uses the Coincidence Index method to find the size of the key and frequency analysis to find the possible key that encrypted the text.*

Introdução

A cifra de Vigenère é um método de criptografia que se baseia no uso de diversas cifras de César em sequência, tendo diversos valores de deslocamento no texto determinados por uma ”palavra-chave”. Para o processo de cifragem se faz uso de uma tabela de alfabetos escrito 26 vezes (considerando o alfabeto romano) em linhas diferentes em que ao cifrar o texto cada um é deslocado ciclicamente à esquerda se comparado com o alfabeto anterior, a fim de corresponder às 26 cifras de César. Dessa forma, quando o texto é criptografado, a cifra usa um alfabeto diferente de uma das linhas sendo que o alfabeto a ser utilizado em cada um dos pontos é com base em uma palavra-chave, a qual é repetida se o texto a ser cifrado é maior que ela [dos Reis 2016].

Entretanto, devido a ténicas conhecidas atualmente como o cálculo de índice de coincidência e análise de frequência das letras do texto cifrado que são utilizadas para encontrar a palavra-chave esse método de criptografia se tornou obsoleto. Ou seja, é uma cifra fácil de quebrar, dado que apesar de embaralhar o texto original, ainda se mantém características do texto original. Dito isso, a seguir será apresentado como foi feita a implementação e suas etapas para a quebra da cifra de Vigenère. Por fim, também serão apresentados os resultados obtidos de palavras-chave para os casos de teste propostos e um link para uma apresentação da implementação feita.

Descobrindo o tamanho da chave

Primeiramente, para decifrar o texto é necessário descobrir a chave com que ele foi cifrado. Para isso, iniciou-se tentando descobrir o tamanho da chave utilizando o cálculo de Índice de Coincidência de Friedman. Dado que, apesar do texto estar cifrado ele ainda mantém uma frequência de caracteres razoável a ponto de apresentar resultados através desse cálculo que representem uma determinada língua. Esse cálculo é feito pela função *coincidenceIndex* na implementação, onde é feito o somatório de *f * (f - 1)* em que *f* é a frequência de cada letra dividido por *n * (n - 1)* sendo *n* o tamanho do texto.

O único problema é que a cifra de Vigenère não mantém as mesmas frequências do texto claro para o cifrado por conta da chave fazer com que os caracteres tenham deslocamentos variados. Entretanto, como para a cifra de Vigenère se utiliza uma chave com comprimento *m* temos que o mesmo caracter da chave *k* é usada de *m* em *m* caracteres podendo-se utilizar disso para quebrá-la. Com o propósito de aproveitar essa fraqueza, utilizou-se uma outra função *getDividedStrings* que divide o texto até um limite de tamanho de chave pré-determinado. Além de realizar a divisão essa função também calcula o Índice de Coincidência para cada trecho de texto dividido e armazena a letra mais frequente que será utilizada na próxima etapa da decifragem.

Dessa forma, ao dividir os textos pelo tamanho da chave garante-se que os trechos divididos de *m* em *m* caracteres mantém características da língua no texto claro e assim podemos trabalhar com os resultados cálculo do Índice de Coincidência. Assim, se for feita a média dos valores resultantes para um tamanho de chave *m* e ele corresponder aos valores indicados pela língua teremos uma estimativa de que possivelmente *m* é o tamanho da chave. A função *checkCoincidenceIndex* calcula essa média no algoritmo implementado com base nas informações passadas pela função *getDividedStrings* e retorna qual o provável tamanho da chave com base nas características da língua. Para os casos de teste sugeridos o tamanho da chave se deu pelo maior valor encontrado nesse conjunto de médias, sendo que valores mais baixos indicam que o texto é aleatório, portanto não identificando as características da língua mencionados anteriormente.

Descobrindo os caracteres da chave

Tendo o tamanho da chave, a língua do texto cifrado e o caracter mais frequente de cada trecho dividido do texto, pode-se partir para a descoberta dos caracteres da chave por meio da análise de frequencia. Dado que existe uma frequência característica de caracteres dentro de cada língua, pode-se comparar o caracter mais frequente encontrado com o caracter mais frequente da língua. Ou seja, a distância do caracter mais frequente da língua até o caracter mais frequente naquela divisão do texto é o índice da letra da chave no alfabeto. A função *LetterDistance* recebe as duas letras e o alfabeto para assim calcular esse índice e retornar o caracter que corresponde a chave daquela sequência do texto.

Contudo, como para o português a porcentagem de frequência do caracter mais frequente para o segundo é muito próxima, em alguns casos, o mais frequente do texto corresponde ao segundo e não ao primeiro. Por isso, esse processo de cálculo de distância e identificação de elementos da chave é feito para as 26 letras, porém na implementação feita é apresentado quais seriam os caracteres com mais frequência na linguagem selecionada. Ainda assim, alguns casos a chave encontrada trocava um ou dois caracteres. Afinal, para todas sequências se utilizava a mesma comparação.

Por isso, implementou-se uma segunda solução para descoberta dos caracteres utilizando uma técnica estatística para verificar quantitativamente a relação entre deslocamentos possíveis dos caracteres e a frequência desses mesmos caracteres esperada no texto da língua. O nome da técnica utilizada é a distribuição Qui-Quadrado [Chambers 2014]. Dessa forma, para cada sequencia do texto cifrado verifica-se cada deslocamento possível e para cada um calcula-se a margem de erro das frequências a fim de atribuir uma pontuação para cada deslocamento. No entanto, como a frequência pode estar acima ou abaixo do estimado, essa margem varia para cada letra e se for feito o somatório desses resultados eles irão se anular. Logo, a fim de ter uma pontuação significativa elevou-se esses valores ao quadrado e dividiu-se pela frequência esperada da língua para poder efetuar o somatório com valores normalizados. Sendo assim, o deslocamento que tiver o menor valor desse cálculo é um candidato para ser o caracter que cifrou aquela sequência do texto. Assim, na implementação feita, a função *freqAnalysis* realiza essa análise recebendo uma sequência, o alfabeto e as frequências da língua. Por fim, obtendo-se uma outra possibilidade de chave com os caracteres descobertos por essa função.

Apresentação da implementação

Para executar a implementação basta ter *Python 3.11* instalado estar em um terminal no mesmo diretório do arquivo `t1.py` e seguir o seguinte padrão de execução e parametrização:

```
python t1.py <texto.txt> <limite_da_chave> <opcional:lingua>
```

Abaixo também o link para o vídeo de apresentação do programa:

`https://www.youtube.com/watch?v=2qs59oQzln8&ab_channel=ArthurHenz`

Resultados

A tabela a seguir apresenta qual o arquivo do texto cifrado, tamanho da chave e a própria palavra-chave encontrados, para cada caso de teste proposto após sua execução no algoritmo implementado em *Python 3.11*:

Tabela 1. Resultados dos casos de teste obtidos para o algoritmo implementado.

Caso de teste	Tamanho da chave	Chave
cipher1.txt	8	cristian
cipher2.txt	10	daviddavid
cipher3.txt	10	diegodiego
cipher4.txt	7	eduardo
cipher5.txt	6	felipe
cipher6.txt	7	girotto
cipher7.txt	7	gregory
cipher8.txt	8	hercilio
cipher9.txt	6	maurer
cipher10.txt	6	rangel
cipher11.txt	9	jerusalem

cipher12.txt	8	software
cipher13.txt	8	igorigor
cipher14.txt	9	joaopedro
cipher15.txt	10	steinstein
cipher16.txt	7	schuler
cipher17.txt	7	marcelo
cipher18.txt	6	mateus
cipher19.txt	7	matheus
cipher20.txt	7	mathias
cipher21.txt	10	paulopaulo
cipher22.txt	6	ritter
cipher23.txt	10	companhoni
cipher24.txt	7	cadaval
cipher25.txt	6	renata
cipher26.txt	7	ricardo
cipher27.txt	7	rodrigo
cipher28.txt	6	branco
cipher29.txt	10	krothkroth
cipher30.txt	9	virgilius
cipher31.txt	10	vitorvitor

Conclusões

Na história, assim como a criptografia evoluiu ao longo dos séculos, também evoluíram as técnicas de decifragem, reforçando a necessidade de constante inovação nas ténicas de cifragem. Este trabalho reforça o quão importante é se colocar no lugar de um criptoanalista para que, ao realizar o processo de cifragem tenhamos em mente que qualquer informação passada do texto original ao texto cifrado pode ser usada para decifrar o mesmo. Assim sendo, foram descritas o que algumas das principais funções de uma implementação para quebra da cifra de Vigenère realizam, dessa maneira reiterando esse aprendizado.

Referências

Chambers, A. (2014). Using chi squared to crack codes. https://ibmathsresources.com/2014/06/15/using-chi-squared-to-crack-codes/. [online: acesso em 10-setembro-2023].

dos Reis, F. (2016). Criptografia cifra de vigenere. http://www.bosontreinamentos.com.br/seguranca/criptografia-cifra-de-vigenere/. [online: acesso em 10-setembro-2023].