Segurança computacional: Cifra de Vigenère

1st Gustavo Pierre Starling - (202006420)
Ciência da computação
Universidade de Brasília)
Brasília, Brasil
gugapierrestarling@gmail.com

2nd Bruno Vargas de Souza - (202006564)
Ciência da computação
Universidade de Brasília
Brasília, Brasil
brunovargas7899@gmail.com

***Abstract*—This project aims to implement a cipher and decoder using the Vigenère cipher. In addition, it also performs a key recovery attack by frequency analysis, making it possible to decrypt a message without knowing the key.**

***Index Terms*—cipher decoder, Vigenère**

I. Resumo

Este é o projeto 1 de Segurança computacional, o qual têm como objetivo a implementação de um cifrador e de um decifrador. Além disso, também realiza-se um ataque de recuperação de chave por análise de frequência, sendo possível decifrar uma mensagem sem o conhecimento da chave.

II. Introdução

A criptografia no âmbito da segurança computacional consiste na conversão de dados de um formato legível em um formato codificado, a fim de evitar que seu conteúdo seja revelado em casos de interceptação. Assim, utiliza-se um cifrador para realizar tal conversão. Além disso, com o intuito de ler/processar os dados criptografados, faz-se necessário o uso de um discifrador, o qual converte a mensagem no formato codificado na mensagem original.

A cifra de Vigenère constitui um método de criptografia, o qual é escolhido uma chave e que cada letra da mensagem se relaciona com a letra da chave da sua respectiva posição, correspondendo a um determinado valor na tabela de vigenère (Figura 1). Por exemplo, para a mensagem: ”attackatdown” e a chave ”lemonlemonle”, o resultado será ”lxfopvefrnhr”.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Fig. 1. Tabela de Vigenère

III. Metodologia

Esta secção detalha os procedimentos realizados na implementação no projeto. Nesse sentido, utiliza-se a linguagem de programação C++ para realizar o cifrador de vigenere com chave e o decifrador de vigenere com e sem chave.

A. Cifrador

A priori, faz-se necessário escolher uma chave e concatenar a chave com ela mesma até igualar o seu tamanho com o tamanho da mensagem que deseja codificar, como observado na figura 2. Após isso, aplica-se a fórmula da figura 3 com a chave atualizada e a mensagem para a cifração [1]. Nota-se na figura 4 que optou-se por manter os caracteres especiais, acentuação e pontuação inalterados. [2]

```
string equal_key(string message, string key){
    string aux = "";
    int j = 0;
    for(int i=0; i<message.size(); i++){
        if(message[i] < 'a' or message[i] > 'z') aux += message[i];
        else{
            aux += key[j];
            j++;
            j %= key.size();
        }
    }
    return aux;
}
```

Fig. 2. Igualar tamanho da chave

$$C_i \equiv P_i + K_i \pmod{26}$$

Fig. 3. Fórmula do cifrador de vigenère

```
void cifrador(string key, string message){
    key = equal_key(message, key);
    // cifra de acordo com a tabela de vigenere
    string msg = "";
    for(int i=0; i<key.size(); i++){
        if(message[i] < 'a' or message[i] > 'z') msg += message[i];
        else msg += (((message[i]-'a') + key[i]-'a') % 26) + 'a';
    }
    cout << GRN << msg << NC << endl;
}
```

Fig. 4. Implementação do cifrador de vigenère

B. Decifrador com chave

Primeiramente, repete-se o mesmo processo inicial do cifrador de igualar o tamanho da chave, mostrado na figura 2. Após isso, aplica-se a fórmula da figura 5 com a chave atualizada e a mensagem para a decifração [1]. Verifica-se na figura 6 que optou-se por manter os caracteres especiais, acentuação e pontuação inalterados. [2]

$$P_i \equiv C_i - K_i + 26 \pmod{26}$$

Fig. 5. Fórmula do decifrador de vigenère

```
void decifrador(string key, string message){
    key = equal_key(message, key);
    // decifra de acordo com a tabela de vigenere
    string msg = "";
    for(int i=0; i<message.size(); i++){
        if(message[i] < 'a' or message[i] > 'z') msg += message[i];
        else msg += ((((message[i]-'a')-(key[i]-'a')) + 26) % 26) + 'a';
    }
    cout << GRN << msg << NC << endl;
}
```

Fig. 6. Implementação do decifrador de vigenère

C. Decifrador sem chave (ataque)

A princípio, realiza-se uma tentativa em encontrar o tamanho da chave por meio das trigramas repitidas na mensagem codificada. Nesse contexto, calcula-se a distância de trigramas repetidas, após isso determina a quantidade de cada divisor de cada distância. Dessa forma, estima-se que o tamanho da chave consiste no divisor que se repete mais vezes. Observa-se também que se trata de uma estimativa e que em alguns casos, o tamanho ideal da chave será um múltiplo do valor estimado. [3]

```
int key_size(vector<string> data){
    int count=0;
    string text = clean_text(data);

    map<string, set<int>> trigramas;
    for(int i=0; i<text.size()-2; i++){
        string tri_a = text.substr(i, 3);
        for(int j=i+1; j<text.size()-2; j++){
            string tri_b = text.substr(j, 3);
            if(tri_a == tri_b){
                trigramas[tri_a].insert(j-i);
            }
        }
    }

    map<int, int> divs;
    for(auto trig: trigramas){
        set<int> distances = trig.second;
        for(auto d: distances){
            for(int i=2; i<=20; i++){
                if(d%i == 0){
                    divs[i]++;
                }
            }
        }
    }
```

Fig. 7. Implementação da estimativa do tamanho da chave

Após encontrar o tamanho da chave, itera-se para cada letra da chave, tentando encontrar a probabilidade mais próxima, tendo como referência as letras mais utilizadas no idioma da mensagem. [4]

```
string get_key(int key_size, string message, string language){
    string key = "";
    for(int i=0; i<key_size; i++){
        int total = 0;
        vector<int> freq(26);
        for(int j=i; j<message.size(); j+= key_size){
            freq[message[j]-'a']++;
            total++;
        }
        vector<double> probabilities;
        for(int c=0; c<26; c++){
            double x = (freq[c]/(double)total);
            probabilities.push_back(x);
        }
        key += get_letter(probabilities, language);
    }
    return key;
}
```

Fig. 8. Frequência da letra parte 1

```
char get_letter(vector<double> probabilities, string lang){
    vector<double> en = {0.08167,0.01492,0.02782,0.04253,0.12702, 0.02228, 0.0
    vector<double> pt = {0.1463, 0.0104, 0.0388, 0.0499, 0.1257, 0.0102, 0.013

    vector<double> prob = (lang == "en" ? en : pt);

    double menor = 0x3f3f3f3f;
    char c;
    // shiftando e pegando a menor diferenca
    for(int i=0; i<26; i++){
        double diff = 0;
        for(int j=0; j<26; j++){
            diff += abs(probabilities[(i+j)%26] - prob[j]);
        }
        if(diff < menor){
            menor = diff;
            c = i + 'a';
        }
    }
    return c;
}
```

Fig. 9. Frequência da letra parte 2

IV. Conclusão

Verifica-se que é possível decifrar uma mensagem codificada pela cifra de Vigenère sem o uso da chave, porém o algorítmo não têm acurácia absoluta, uma vez que pode-se falhar em amostras muito pequenas ou ao estimar o tamanho da chave errado.

References

[1] "Vigenere cipher," https://www.youtube.com/watch?v=SkJcmCaHqS0.
[2] "Cifra de vigenère," https://pt.wikipedia.org/wiki/Cifra$_{d}e_{V}igenère$.
[3] "Cryptography - breaking the vigenere cipher," https://www.youtube.com/watch?v=P4z3jAOzT9I.
[4] "Frequência de letras," https://pt.wikipedia.org/wiki/Frequência$_{d}e_{l}etras$.