Relatório - Cifra de Vigenere

Rodrigo Pereira Couto - 190116510

Introdução

Este trabalho explora a cifra de Vigente onde é uma cifra de substituição polialfabética, o que significa que usa uma chave diferente para cada letra do texto a ser cifrado. A chave é uma sequência de letras que é usada para gerar uma tabela de substituição. A cifra de Vigenère foi inventada no século XVI pelo diplomata e criptoanalista francês Blaise de Vigenère. É considerada uma das cifras mais fortes do período pré-computacional.

Codificação

Começamos a codificação convertendo os caracteres e a chave para valores entre 0 a 25
No pegamos esses valores somamos e aplicamos módulo 26
Depois convertemos para caractere novamente

```
string cifraVigenere(string texto, string chave) {
    string textoCifrado = "";
    int tamanhoChave = chave.length();

    for (int i = 0; i < texto.length(); i++) {
        char caractereOriginal = texto[i];
        char caractereChave = chave[i % tamanhoChave];

        // Verifique se o caractere original é uma letra do alfabeto
        if (isalpha(caractereOriginal)) {
            char base = islower(caractereOriginal) ? 'a' : 'A';

            // Converta o caractere original e chave para valores de 0 a 25
            int valorOriginal = caractereOriginal - base;

            int valorChave = tolower(caractereChave) - 'a';

            // Aplique a cifração de Vigenère
            int valorCifrado = (valorOriginal + valorChave) % 26;

            // Converta o valor cifrado de volta para um caractere no mesmo caso (minúscula ou maiúscula)
            char caractereCifrado = valorCifrado + base;

            // Adicione o caractere cifrado à string de texto cifrado
            textoCifrado += caractereCifrado;
        } else {
            // Mantenha caracteres não alfabéticos inalterados
            textoCifrado += caractereOriginal;
        }
    }
    return textoCifrado;
}
```

Decodificação

A decodificação só muda da codificação no momento de aplicar a cifração, que em vez de somar os valores, nos subtraímos os valores e somamos 26 após a subtração e fazemos o módulo 26. Isso garante que o resultado seja um valor positivo dentro do intervalo correto.

```
string decifraVigenere(string textoCifrado, string chave) {
    string textoDecifrado = "";
    int tamanhoChave = chave.length();

    for (int i = 0; i < textoCifrado.length(); i++) {
        char caractereCifrado = textoCifrado[i];
        char caractereChave = chave[i % tamanhoChave];

        // Verifique se o caractere cifrado é uma letra do alfabeto
        if (isalpha(caractereCifrado)) {
            char base = islower(caractereCifrado) ? 'a' : 'A';

            // Converta o caractere cifrado e chave para valores de 0 a 25
            int valorCifrado = caractereCifrado - base;

            int valorChave = tolower(caractereChave) - 'a';


            // Aplique a decifração de Vigenère
            int valorDecifrado = (valorCifrado - valorChave + 26) % 26;

            // Converta o valor decifrado de volta para um caractere no mesmo caso (minúscula ou maiúscula)
            char caractereDecifrado = valorDecifrado + base;

            // Adicione o caractere decifrado à string de texto decifrado
            textoDecifrado += caractereDecifrado;
        } else {
            // Mantenha caracteres não alfabéticos inalterados
            textoDecifrado += caractereCifrado;
        }
    }

    return textoDecifrado;
}
```

Ataque
Descobrindo Tamanho da chave

A função **findKeySize** é responsável por tentar encontrar o tamanho da chave utilizada na cifra de Vigenère com base no texto cifrado. O método utilizado é identificar repetições de sequências de três caracteres no texto. Aqui está uma explicação passo a passo do que a função faz:

Ela inicializa um vetor factors de tamanho 19 com zeros. Esse vetor será usado para contar quantas vezes diferentes comprimentos de chave são encontrados.
A função entra em um loop que varre o texto, começando do primeiro caractere até o terceiro a partir do final (para evitar índices fora do limite).
Em cada iteração do loop externo, ela pega uma sequência de três caracteres do texto, começando na posição i.
Em seguida, ela entra em um segundo loop interno que começa três posições após o primeiro loop. Isso é feito para procurar repetições da sequência em outras partes do texto, a fim de identificar um possível período da chave.
Se uma repetição da sequência é encontrada, a função calcula a distância entre as duas ocorrências e, em seguida, verifica se essa distância é divisível por valores de 2 a 20 (representados pelo loop k). Se for divisível, isso sugere que esse comprimento pode ser o tamanho da chave.
A cada vez que um possível tamanho de chave é encontrado, o vetor factors é atualizado de acordo com o tamanho encontrado.

Após percorrer todo o texto, a função imprime a contagem de cada tamanho de chave encontrado.
O usuário é solicitado a selecionar o tamanho da chave a ser usado com base nas contagens exibidas.
A função retorna o tamanho da chave selecionado pelo usuário.

```
int findKeySize(const string &text)
{
    vector<int> factors(19, 0); // Criar um vetor de 19 elementos
inicializados com 0

    for (int i = 0; i < text.length() - 2; i++)
    {
        string seq = text.substr(i, 3);

        for (int j = i + 3; j < text.length() - 2; j++)
        {
            if (seq == text.substr(j, 3))
            {
                int dist = j - i;
                // cout << seq << " " << dist << endl;
                for (int k = 2; k <= 20; k++)
                {
                    if (dist % k == 0)
                    {
                        factors[k - 2]++;
                    }
                }
                break;
            }
        }
    }

    cout << "Tamanhos de chaves e suas quantidades encontradas:" << endl;

    for (int i = 0; i < factors.size(); i++)
    {
        cout << "\t" << i + 2 << " : " << factors[i] << endl;
    }

    int selected;
    cout << "Selecione o tamanho da chave: ";
    cin >> selected;

    return selected;
}
```

A função **shiftMap** é usada para fazer um deslocamento circular nas letras de um mapa (utilizado para representar as frequências das letras). Isso é útil para realizar a análise de frequência com diferentes deslocamentos, tentando encontrar a chave correta.

```
void shiftMap(map<char, double> &myMap, int shift)
{
    // Cria um novo map para armazenar o resultado do shift
    map<char, double> shiftedMap;

    // Itera sobre o map original
    for (auto pair : myMap)
    {
        // Calcula o novo indice da letra
        int newIndex = (pair.first - 'a' + shift) % 26;

        // Adiciona a letra no novo map
        shiftedMap['a' + newIndex] = pair.second;
    }

    // Copia o novo map para o map original
    myMap = shiftedMap;
}
```

A função **divideEmBlocos** divide uma string em vários blocos de tamanho igual. No contexto da cifra de Vigenère, ela é usada para dividir o texto cifrado em blocos para análise.

```
vector<string> divideEmBlocos(const string &texto, int tamanhoBloco)
{

    vector<string> blocos(tamanhoBloco);

    for (int i = 0; i < texto.length(); i++)
    {
        blocos[i % tamanhoBloco] += texto[i];
    }

    return blocos;
}
```

A função **aplicaAnaliseDeFrequencia** é responsável por aplicar a análise de frequência em cada bloco do texto cifrado. Ela calcula a frequência das letras em cada bloco, compara com as frequências esperadas da língua (português ou inglês) e tenta encontrar o deslocamento que minimiza as diferenças entre as frequências.

```
string aplicaAnaliseDeFrequencia(const vector<string> &sequencias, map<char, double>
frequenciaLingua)
{
    string chave = "";
    for (const string &sequencia : sequencias)
    {
        map<char, int> frequenciaLetras;
        for (char letra : sequencia)
        {
            // lower case
            letra = tolower(letra);
            frequenciaLetras[letra]++;
        }
        // letras que não aparecem na sequencia tem frequencia 0
        for (char letra = 'a'; letra <= 'z'; letra++)
        {
            if (frequenciaLetras.find(letra) == frequenciaLetras.end())
            {
                frequenciaLetras[letra] = 0;
            }
        }
        // deixar em %
        map<char, double> frequenciaLetrasPercentual;

        for (auto par : frequenciaLetras)
        {
            frequenciaLetrasPercentual[par.first] = (double)par.second /
(double)sequencia.length();
        }

        vector<double> dif;
        for (int i = 0; i < 26; i++)
        {
            double soma = 0.0;

            for (auto par : frequenciaLetrasPercentual)
            {
                map<char, double> freqLetrasAux = frequenciaLingua;
                shiftMap(freqLetrasAux, i);
                soma += abs(freqLetrasAux[par.first] / 100 - par.second);
            }
            dif.push_back(soma);
        }

        int shift = min_element(dif.begin(), dif.end()) - dif.begin();
        chave += 'a' + shift;
    }
    return chave;
}
```

conclusão

Em resumo, a cifra de Vigenère é uma técnica histórica e importante na criptografia, que ilustra como a segurança pode ser aumentada através do uso de chaves e da complicação do processo de codificação. É um exemplo valioso de criptografia polialfabética e ainda é estudada hoje em aulas de criptografia e segurança da informação. Durante o desenvolvimento do trabalho sobre a cifra de Vigenère, enfrentei algumas dificuldades, mas consegui superá-las e aprender valiosas lições sobre criptografia. Uma das principais dificuldades que encontrei ao trabalhar com essa cifra foi a análise de frequência, que, embora seja uma técnica poderosa, pode ser mais complicada de aplicar na cifra de Vigenère.