Relatório Projeto 1
Segurança Computacional 2023/2

Cifra de Vigenère

Gabriel Nazareno Halabi - 150010290

INTRODUÇÃO

O objetivo deste projeto é a implementação de uma Cifra de Vigenère e funções de ataque capazes de quebrar a cifra de forma automatizada. A Cifra de Vigenère é um cifra polialfabética que utiliza uma combinação de mútiplas cifras de césar para para intervalos regulares da mensagem, de forma que decifrar manualmente o código fique significativamente mais trabalhoso. Se um atacante souber, porém, o tamanho da chave utilizada e a linguagem na qual a mensagem foi escrita, torna-se simples a quebra da cifra por meio da análise de frequência dos caracteres individualmente e de suas combinações.

IMPLEMENTAÇÃO

O cifrador implementado é uma classe VigenereCipher desenvolvida em python, durante a sua inicialização é gerado um dicionário de listas contendo um par de tabelas para a cifração e decifração para cada valor de deslocamento da chave, desta maneira economizando processamento após a inicialização do objeto da classe durante a operação. Cada índice do dicionário possui um par de tabelas equivalentes para cifrar e decifrar o texto, respectivamente.

```
    def __init__(self, alphabet, case_sensitive = False):
        self.alphabet = alphabet
        self.tables = {}
        self.case_sensitive = case_sensitive

        for i in range(len(self.alphabet)):
            shift = self.alphabet[i:]+self.alphabet[:i]
            enc = {}
            dec = {}

            for j in range(len(self.alphabet)):
                enc[self.alphabet[j]] = shift[j]
                dec[shift[j]] = self.alphabet[j]

            self.tables[self.alphabet[i]] = (enc, dec)
```

Dado esse dicionário de tabelas de deslocamento, a cifração e decifração é realizada da mesma forma, passando caractere a caractere do string de input junto do caractere equivalente da sequência da chave como índices e o charactere devidamente deslocado é retornado, sendo adicionado à string de output.

```
    def vigenere(self, text_in, key, decode = False):
        keystring = self.keystring(key, len(text_in))
        text_out = ''

        if not self.case_sensitive:
            text_in = text_in.upper()

        j = 0
        for i in range(len(text_in)):
            c = text_in[i]
            k = keystring[j]

            if c in self.alphabet and k in self.alphabet:
                text_out += self.tables[k][int(decode)][c]
                j += 1
            else:
                text_out += c

        return text_out
```

ATAQUE

A quebra da cifra é realizada por outra classe definida como VigenereBreaker, e é feita em duas etapas. Primeiro é necessário definir o tamanho da chave e portanto a função percorre o texto cifrado buscando a repetição de combinações de caracteres e calcula a distância entre essas ocorrências pois há alta probabilidade de a distância ser um mútiplo do comprimento da chave, retornando assim uma lista contendo os tamanhos mais prováveis.

```
    def keysize(self, cipher, lang, size):
        chars = ''.join(list(self.alphabets[lang].keys()))
        cipher = re.sub('[^'+chars+']','',cipher.upper())
        factors = {}

        for i in range(len(cipher)-(size-1)):
            for j in range(i+size, len(cipher)-(size-1)):
                if cipher[j:j+size] == cipher[i:i+size]:
                    for k in range(2,51):
                        if (j-i)%k == 0:
                            factors[k] = 1+factors.get(k,0)
                    break

        if len(factors):
            return dict(sorted(factors.items(), key=lambda item: item[0]))
        else:
            return self.keysize(cipher, lang, size-1) if size else {}
```

A segunda etapa é a análise de frequência da ocorrência de cada caractere em cada substring cifrada pelo mesmo caractere da chave utilizada, agora que se sabe seu comprimento. A forma que isto é feito é com a separação do texto cifrado em substrings para cada caractere da chave, que por sua vez tem um dicionário criado com a contagem de ocorrências de cada caractere que ocorre e esstas frequências são comparadas com as frequências conhecidas da linguagem passada como argumento da função, o que pode ser facilmente realizado considerando que não um embaralhamento de caracteres mas apenas um deslocamento destes.

```
    def breakVigenere(self, cipher, lang, keysize = 4):
        chars = ''.join(list(self.alphabets[lang].keys()))
        freqs = list(self.alphabets[lang].values())

        cipher = re.sub('[^'+chars+']','',cipher.upper())

        key = ''
        for i in range(keysize):
            subset = cipher[i::keysize]
            offset = self.freqAnalysis(self.textFreq(subset, chars), freqs)
            key += chr(ord('A')+offset)

        return self.splitKey(key)

    def freqAnalysis(self, text, lang):
        match = []

        for _ in range(len(lang)):
            match.append(sum([text[i]*lang[i] for i in range(len(lang))]))
            text.append(text.pop(0))

        return match.index(max(match))

    def textFreq(self, text, chars):
        freq = dict([(c,0) for c in chars])

        for c in text:
            freq[c] += 1

        for c in chars:
            if freq[c] != 0:
                freq[c] /= len(text)

        return list(freq.values())
```

CONCLUSÃO

Embora a cifra de Vigenère seja uma evolução significativa da cifra de César, tornando um ataque manual trabalhoso, com acesso a algumas informações como frequência de ocorrência de caracteres na linguagem e acesso a poder computacional, a quebra desta cifra se torna trivial. O desenvolvimento do módulo de ataque foi efetivo, porém foi identificada uma dificuldade em fazer com que o programa identificasse sozinho o valor mais adequado para o comprimento da chave, tornando necessário um pouco mais de input do usuário para identificar padrões para um ser humano são mais claros.