УНИВЕРСИТЕТ ИТМО

ФГАОУ ВО «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ

Лабораторная работа №1

Основы шифрования данных

Лабушев Тимофей

Группа P3402

Санкт-Петербург

2021

Цель работы

Изучение основных принципов шифрования информации, знакомство с широко известными алгоритмами шифрования, приобретение навыков их программной реализации.

Задание

Вариант 7. Реализовать в программе шифрование и дешифрацию файла с использованием перестановочного шифра с ключевым словом. Ключевое слово вводится.

Описание шифра

В перестановочном шифре с ключевым словом символы исходного текста помещаются в матрицу с числом столбцов, равным длине ключа, затем столбцы переставляются в соответствии с алфавитным порядков символов в ключе, и полученная матрица считывается в зашифрованный текст по столбцам. Частоты встречаемости символов не изменяются, что может помочь отличить перестановочный шифр от подстановочного.

Интерфейс взаимодействия с программой шифрования

Режим: Зашифровать

Ключ: тайнключ

Удалить пробелы и знаки пунктуации:

Файл: Choose File aaiw-excerpt

Результат работы программы

Длина ключа: 8

Длина входного текста: 236

Длина результата: 240

Результат: Download... result.txt

Исходный код реализации

```
begin
    function encrypt(text::String, key::String)
        keychars::Array{Char,1} = collect(key)
        textchars::Array{Char,1} = collect(text)
        # Text is arranged into `numcols` columns
        numcols = length(keychars)
        # Is the length of the text divisible by the number of columns?
        modchars = length(textchars) % numcols
        # If not, we need to pad it
        if modchars > 0
            # Use the first symbol in the text as the padding symbol
            padding = repeat([textchars[1]], numcols - modchars)
            append!(textchars, padding)
        end
        # Arrange text into columns in row-major order
        T = permutedims(reshape(textchars, numcols, :))
        # Reorder columns based on the order of characters in the key
        T = T[:, sortperm(keychars)]
        # Collapse the matrix into a vector of characters
        ciphertextchars = reshape(T, 1, :)
        # Convert the vector to a string
        join(ciphertextchars)
    end

    function decrypt(ciphertext::String, key::String)
        keychars::Array{Char,1} = collect(key)
        ciphertextchars::Array{Char,1} = collect(ciphertext)
        # Compute the number of columns the source text was arranged into
        numcols = length(keychars)
        # If the ciphertext length is not divisible by `numcols`, the key is incorrect
        if length(ciphertext) % numcols > 0
            throw(ArgumentError("Длина зашифрованного текста не кратна длине ключа"))
        end
        # Arrange the ciphertext into columns in row-major order
        T = permutedims(reshape(ciphertextchars, :, numcols))
        # Find the initial order of columns
        initkeyperm = sortperm(sortperm(keychars))
        # Put the columns of ciphertext into the initial order
        T = T[initkeyperm, :]
        # Collapse the matrix into a vector of characters
        textchars = reshape(T, 1, :)
        # Convert the vector to a string
        join(textchars)
    end

    md"## Исходный код реализации"
end
```

Тестирование алгоритма

```
Test Summary: | Pass  Total
encryption    |    2      2
Test Summary: | Pass  Total
key mismatch  |    1      1
```

```
with_terminal() do
    @testset "encryption" begin
        # arrange abcdefgh into 4 columns:
        # a b c d
        # e f g h
        # next, rearrange the columns as 4321 (DCBA -> ABCD):
        # d c b a
        # h g f e
        # finally, flatten the columns into one in column-major order:
        # d h c g b f a e
        @test encrypt("abcdefgh", "DCBA") == "dhcgbfae"
        @test decrypt("dhcgbfae", "DCBA") == "abcdefgh"
    end

    @testset "key mismatch" begin
        @test_throws ArgumentError decrypt("1234", "123")
    end
end
```

Вскрытие алгоритма

Для вскрытия перестановочного шифра с ключевым словом может применяться перебор с целевой функцией на основе анализа частоты встречаемости n-грамм.

Рассмотрим пример для сообщения на английском языке длиной 109 символов:

IttakesalltherunningyoucandotokeepinthesameplaceIfyouwanttogetsomewhereelseyoumustrunatle

Воспользуемся **таблицей весов биграмм для стандартного текста**.

Вывод программы вскрытия для ключа "eightcrs":

```
Key: [2, 5, 3, 4, 8, 1, 6, 7], text: IttakesalltherunningyoucandotokeepinthesameplaceIfyouwantt
ogetsomewhereelseyoumustrunatleasttwiceasfastasthatIII
Key: [5, 3, 4, 8, 1, 6, 7, 2], text: ttakesaltherunlingyoucnndotokeapinthesemeplaceafyouwanIto
getsotewhereemseyoumultrunatlsasttwiceasfastaethatIIIs
Key: [5, 3, 4, 8, 1, 2, 6, 7], text: ttakeIsaltherluningyonucndotoakepintheesmeplaacefyouwIanto
gettsoewhermeeseyoulmutrunastlasttweicasfasetathatIsII
```

```
with_terminal() do
    breaktest_ciphertext = encrypt(breaktest, "eightcrs")
    results = bruteforce_top_n(breaktest_ciphertext, en_bigram_probabilities, 8, 3)
    for (text, key) in results
        print("Key: $key, text: $text\n")
    end
end
```

Сложность простого перебора составляет *O(n!)*.

Время вскрытия при длине ключа *n* для выбранного сообщения составляет:

- *n* = 8: 0.096062791 с
- *n* = 9: 0.378447856 с
- *n* = 10: 3.546032419 с
- *n* = 11: 48.993470253 с

```
begin
    els_8 = @elapsed bruteforce_top_n(encrypt(breaktest, "testTEST"),
en_bigram_probabilities, 8, 3)
    els_9 = @elapsed bruteforce_top_n(encrypt(breaktest, "testTEST1"),
en_bigram_probabilities, 9, 3)
    els_10 = @elapsed bruteforce_top_n(encrypt(breaktest, "testTEST13"),
en_bigram_probabilities, 10, 3)
    els_11 = @elapsed bruteforce_top_n(encrypt(breaktest, "testTEST313"),
en_bigram_probabilities, 11, 3)

    md"""
    Сложность простого перебора составляет _O(n!)_.

    Время вскрытия при длине ключа _n_ для выбранного сообщения составляет:
    * _n = 8_: $els_8 с
    * _n = 9_: $els_9 с
    * _n = 10_: $els_10 с
    * _n = 11_: $els_11 с
    """
end
```

bruteforce_top_n (generic function with 1 method)

```
function bruteforce_top_n(ciphertext::String, bigram_probs::Matrix{Float32},
maxkeylen::Int, top_n::Int)
    ciphertextchars::Array{Char,1} = collect(ciphertext)

    # Keep a list of 'top_n' deciphered texts with the highest score
    top_n_scores::Array{Int, 1} = []
    top_n_candidates::Array{Tuple{String, Array{Int, 1}}} = []

    # Lock for accessing the list of candidates from multiple threads
    top_n_lock = ReentrantLock()

    # Test each key length from 1 to 'maxkeylen'
    for keylen in 1:1:maxkeylen
        if length(ciphertextchars) % keylen != 0
            continue
        end

        # Arrange the ciphertext into 'keylen' columns in row-major order
        T = permutedims(reshape(ciphertextchars, :, keylen))

        # Calculate the number of key permutations as the factorial of 'keylen'
        col_indexes = [1:1:keylen;]
        numperms = factorial(keylen)

        # Try each permutation in parallel
        Threads.@threads for p in 1:1:numperms
            cols = nthperm(col_indexes, p)
            candidate = T[cols, :]

            score = score_candidate(candidate, bigram_probs)

            lock(top_n_lock) do
                if length(top_n_scores) < top_n
                    push!(top_n_scores, score)
                    push!(top_n_candidates, (join(candidate), cols))
                else
                    (min_score, min_idx) = findmin(top_n_scores)
                    if min_score < score
                        top_n_scores[min_idx] = score
                        top_n_candidates[min_idx] = (join(candidate), cols)
                    end
                end
            end
        end
    end

    top_n_candidates[sortperm(top_n_scores, rev=true)]
end
```

score_candidate (generic function with 1 method)

```
function score_candidate(textmatrix::Matrix{Char}, en_probs::Matrix{Float32})::Int
    sum::Float32 = 0
    for i in 1:1:length(textmatrix)-1
        c1::Char = uppercase(textmatrix[i])
        c2::Char = uppercase(textmatrix[i+1])

        if c1 < 'A' || c1 > 'Z' || c2 < 'A' || c2 > 'Z'
            continue
        end

        bigram_prob::Float32 = en_probs[Int(c1 - 'A') + 1, Int(c2 - 'A') + 1]
        sum += bigram_prob
    end
    round(sum * 10)
end
```

Вывод

В ходе выполнения работы был изучен перестановочный шифр с ключевым словом, реализованы алгоритмы шифрации и дешифрации текстового файла. Также рассмотрена процедура вскрытия шифра с использованием метода простого перебора.