

Федеральное государственное автономное образовательное учреждение высшего образования "Национальный исследовательский университет ИТМО"

Лабораторная работа №1
по дисциплине "Информационная безопасность"

Основы шифрования данных

Вариант 10

Выполнил: студент Саржевский И.А.

Группа: Р3402

Преподаватель: к.т.н., доцент
Маркина Т.А.

г. Санкт-Петербург

2021 г.

Лабораторная работа №1

Основы шифрования данных

Цель работы

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

Задание

Вариант: 10. Реализовать в программе шифрование и дешифрацию содержимого файла по методу Цезаря. Провести частотный анализ зашифрованного файла, осуществляя проверку по файлу с набором ключевых слов.

Метод Цезаря заключается в том, что весь алфавит циклически сдвигается на заданное количество шагов направо или налево, и исходное сообщение записывается при помощи алфавита, индексы которого сдвинуты. Для того, чтобы раскодировать сообщение необходимо провести обратную операцию - сдвиг индексов алфавита в обратном направлении на такое же количество шагов.

Помимо реализации метода Цезаря, в задании предлагается реализовать расшифровку закодированного файла используя частотный анализ. Для этого воспользуемся статистической информацией о распределении частоты использования букв английского алфавита и попробуем сопоставить эти данные с частотой использования букв в зашифрованном сообщении, таким образом восстановив изначальные символы.






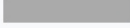







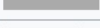














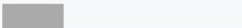





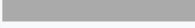
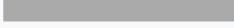
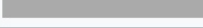
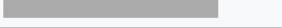














Letter ↕	Relative frequency in the English language			
	Texts ↕		Dictionaries ↕	
a	8.2%		7.8%	
b	1.5%		2%	
c	2.8%		4%	
d	4.3%		3.8%	
e	13%		11%	
f	2.2%		1.4%	
g	2%		3%	
h	6.1%		2.3%	
i	7%		8.6%	
j	0.15%		0.21%	
k	0.77%		0.97%	
l	4%		5.3%	
m	2.4%		2.7%	
n	6.7%		7.2%	
o	7.5%		6.1%	
p	1.9%		2.8%	
q	0.095%		0.19%	
r	6%		7.3%	
s	6.3%		8.7%	
t	9.1%		6.7%	
u	2.8%		3.3%	
v	0.98%		1%	
w	2.4%		0.91%	
x	0.15%		0.27%	
y	2%		1.6%	
z	0.074%		0.44%	

Рис. 1: Относительная частота встречаемости в текстах букв английского алфавита

Листинг разработанной программы

main.rs

```
1  extern crate getopt;
2  use getopt::Options;
3  use std::env;
4  use std::fs;
5
6  mod caesar;
7  mod freq_analysis;
8  // Just parsing cmd args and calling encode or decode function
9  fn main() {
10     let args: Vec<String> = env::args().collect();
11
12     let mut opts = Options::new();
13     opts.optopt("f", "file", "input file path", "input.txt");
14     opts.optopt("o", "offset", "the chipher offset", "4");
15     opts.optflag("l", "left", "left shift mode (default is right)");
16     opts.optflag("d", "decode", "decode mode");
17
18     let matches = match opts.parse(&args[1..]) {
19         Ok(m) => { m }
20         Err(f) => { panic!(f.to_string()) }
21     };
22
23     if !matches.opt_present("f") {
24         println!("You must specify the input file path!");
25         return;
26     }
27
28     if !matches.opt_present("o") && !matches.opt_present("d") {
29         println!("You must specify the offset!");
30         return;
31     }
32
33     let filename = matches.opt_str("f").unwrap();
34     let left = matches.opt_present("l");
35     let decode = matches.opt_present("d");
36
37     let contents = fs::read_to_string(filename)
38         .expect("Something went wrong reading the file");
39
40     if !decode {
41         let offset = matches.opt_str("o").unwrap().parse::<usize>().unwrap();
42         let encoded = caesar::encode_text(contents, offset, left);
43         println!("{}", encoded);
44     } else {
45         let decoded = freq_analysis::decode_text(contents);
46         println!("{}", decoded);
47     }
48 }
```

alphabet.rs

```
1  pub static ALPHABET: [char; 26] = [
2      'a', 'b', 'c', 'd', 'e', 'f', 'g',
3      'h', 'i', 'j', 'k', 'l', 'm', 'n',
4      'o', 'p', 'q', 'r', 's', 't', 'u',
5      'v', 'w', 'x', 'y', 'z'
6  ];
```

caesar.rs

```
1  #[path = "./alphabet.rs"] mod alphabet;
2  // calculate the correct index when shifting to the left
3  fn get_left_idx(idx: usize, offset: usize) -> usize {
4      if idx >= offset {
5          return idx - offset;
6      } else {
```

```

7         return 26 - (offset - idx);
8     }
9 }
10 // caesar chipher implementation
11 pub fn encode_text(input_text: String, offset: usize, left: bool) -> String {
12     let len = input_text.len();
13     let mut result = String::with_capacity(len);
14
15     for cur_char in input_text.chars() {
16         let lower_case = cur_char.to_ascii_lowercase();
17         // find the index of the letter in alphabet
18         let idx = alphabet::ALPHABET.iter().position(|&r| r == lower_case).unwrap_or(27);
19         // find the shifted index of the letter
20         let new_idx = if left {get_left_idx(idx, offset % 26)} else {(idx + offset) % 26};
21         // get the new character based on shifted index
22         let mut new_char = if idx == 27 {lower_case} else {alphabet::ALPHABET[new_idx]};
23         // preserve original case
24         if cur_char.is_ascii_uppercase() {
25             new_char = new_char.to_ascii_uppercase();
26         }
27         // store new character
28         result.push(new_char);
29     };
30
31     return result;
32 }

```

freq_analysis.rs

```

1  #[path = "./alphabet.rs"] mod alphabet;
2
3  const REL_FREQ: [f32; 26] = [
4      8.2, 1.5, 2.8, 4.3, 13.0, 2.2, 2.0,
5      6.1, 7.0, 0.15, 0.77, 4.0, 2.4, 6.7,
6      7.5, 1.9, 0.095, 6.0, 6.3, 9.1, 2.8,
7      0.98, 2.4, 0.15, 2.0, 0.074
8  ];
9  // find the closest by the probability english letter
10 pub fn get_closest_alphabet_idx(freq: f32) -> usize {
11     let mut min = f32::MAX;
12     let mut min_idx = 27;
13     for i in 0..26 {
14         let d = (REL_FREQ[i] - freq).abs();
15         if d < min {
16             min = d;
17             min_idx = i;
18         }
19     };
20
21     return min_idx;
22 }
23
24 pub fn decode_text(encoded_text: String) -> String {
25     // here we count all the occurrences of all the english letters
26     // and total letter count to count the relative freqs later
27     let mut letter_count: [usize; 26] = [0; 26];
28     let mut letters_sym_count = 0;
29     for c in encoded_text.chars() {
30         if c.is_ascii_alphabetic() {
31             letters_sym_count += 1;
32             let idx = alphabet::ALPHABET.iter().position(|&r| r == c.to_ascii_lowercase()).unwrap();
33             letter_count[idx] += 1;
34         };
35     };
36
37     // find the closest real letter for every letter in encoded text
38     let mut real_idx_mapping: [usize; 26] = [27; 26];
39     for (i, c) in letter_count.iter().enumerate() {
40         real_idx_mapping[i] = get_closest_alphabet_idx((*c as f32 / letters_sym_count as f32) * 100.0);
41     }
42
43     // allocate the space for the result string

```

```

44     let mut res = String::with_capacity(encoded_text.len());
45     // replacing the characters from encoded string with the
46     // assumed real characters
47     for c in encoded_text.chars() {
48         let idx = alphabet::ALPHABET.iter().position(|&r| r == c.to_ascii_lowercase()).unwrap_or(27);
49         let mut new_char = if idx == 27 {c} else {alphabet::ALPHABET[real_idx_mapping[idx]]};
50         if c.is_ascii_uppercase() {
51             new_char = new_char.to_ascii_uppercase();
52         }
53         res.push(new_char);
54     }
55
56     return res;
57 }

```

Результаты работы программы

```

keker (~/.code/itmo-4th-year/infosec/lab1/target/debug) master
↳ cat example.txt
I jumped in the river and what did I see?
Black-eyed angels swam with me
A moon full of stars and astral cars
All the things I used to see
All my lovers were there with me
All my past and futures
And we all went to heaven in a little row boat
There was nothing to fear and nothing to doubt
keker (~/.code/itmo-4th-year/infosec/lab1/target/debug) master
↳ ./lab1 -f example.txt -o 5 | tee encoded.txt
N ozruji ns ymj wnajw fsi bmfy ini N xjj?
Gqfhp-jdji fsljqx xbfr bnym rj
F rtts kzqq tk xyfwx fsi fxywfq hfwx
Fqq ymj ymnslx N zxji yt xjj
Fqq rd qtajwx bjwj ymjwj bnym rj
Fqq rd ufxxy fsi kzyzwjx
Fsi bj fqq bjsy yt mjfajs ns f qnyyqj wtb gtfy
Ymjwj bfx stymnsl yt kjfw fsi stymnsl yt itzgy

keker (~/.code/itmo-4th-year/infosec/lab1/target/debug) master
↳ ./lab1 -f encoded.txt -o 5 -l
I jumped in the river and what did I see?
Black-eyed angels swam with me
A moon full of stars and astral cars
All the things I used to see
All my lovers were there with me
All my past and futures
And we all went to heaven in a little row boat
There was nothing to fear and nothing to doubt

```

Рис. 2: Результат работы программы (обычный режим)

```

keker (~/.code/itmo-4th-year/infosec/lab1/target/debug) master
↳ head -c 400 input.txt
A merry little surge of electricity piped by automatic alarm from the mood organ beside his bedawakened
Rick Deckard. Surprised - it always surprised him to find himself awake without priornotic
e - he rose from the bed, stood up in his multicolored pajamas, and stretched. Now, in her bed,his wife
Iran opened her gray, unmerry eyes, blinked, then groaned and shut her eyes again.      "Y%
keker (~/.code/itmo-4th-year/infosec/lab1/target/debug) master
↳ ./lab1 -f input.txt -o 3 | tee encoded.txt | head -c 400
D phuub olwwoh vxujh ri hohfwulflwb slshg eb dxwrpdwlf dodup iurp wkh prrg rujdq ehvlgh klv ehgdzdnhqhg
Ulfh Ghfndug. Vxusulvhg - lw dozdbv vxusulvhg klp wr ilqg klpvhoi dzdnh zlwkrxw sulruqrwlf
h - kh urvh iurp wkh ehg, vwrrg xs lq klv pxowlfroruhg sdmddpv, dqg vwuhwfkhg. Qrz, lq khu ehg,klv zlih
Ludq rshqhg khu judb, xqphuub hbbv, eolqnhg, wkhq jurdqhg dqg vkxw khu hbbv djdlq.      "B%
keker (~/.code/itmo-4th-year/infosec/lab1/target/debug) master
↳ ./lab1 -f encoded.txt -d | head -c 600
A merrg dattde rcrge om edectracatg baber bg actomatac adarm mrom the moor organ berare har beragakener
Rack Reckarr. Rcrbrarer - at adgagr rcrbrarer ham to manr hamredm agake gathoct braornotac
e - he rore mrom the ber, rtoor cb an har mcdtacodorer bajamar, anr rtretcher. Nog, an her ber,har game
Aran obener her grag, cmerrg eger, bdanker, then groaner anr rhct her eger agaan.      "Goc ret gocr Benm
aedr too geak he raar to her. "A'dd reret at anr goc'dd be agake anr - "      "Keeb gocr hanr omm mg retta
ngr." Her koace hedr batter rharnerr. "A ron't gant to be agake."      He reater %

```

Рис. 3: Результат работы программы (частотный анализ)

Вывод

В результате выполнения данной лабораторной работы был имплементирован шифр Цезаря для шифрования файла и метод частотного анализа для его дешифровки. Нужно отметить, что полностью восстановить исходный текст не удалось, так как дешифрующий метод основан на статистических данных. Тем не менее, некоторые ключевые слова удалось распознать - organ, he, her, to, be. В случае с шифром Цезаря этого более чем достаточно, чтобы определить смещение и раскодировать все сообщение.