Министерство образования Республики Беларусь

Учреждение образования

“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Факультет информационных технологий и управления

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №2

по курсу “Средства и методы защиты информации в интеллектуальных системах”

|  |  |
| --- | --- |
| Выполнил:  студент гр. 221702 | Юргилевич Е. В. |
| Проверил: | Крищенович В.А. |

Минск 2024

Лабораторная работа №2

Простейшие криптографические преобразования

**Задание:**

1) Реализовать в виде программы шифр (зашифрование и расшифрование) в соответствии с вариантом. Язык исходного текста русский или английский по выбору исполнителя.

2) Реализовать в виде программы атаку полным перебором ключа,

используя для оценки правильности выбора ключа визуальный метод или исходный текст для автоматического сравнения результата дешифрования.

3) Оценить криптографическую стойкость реализованного шифра.

4) Предложить варианты усложнения шифра. Предложенные варианты оформить в виде алгоритма.

**Вариант 1**

Вариант шифра: шифр Цезаря.

**Описание шифра:**

Шифр Цезаря – это один из шифров подстановки, в котором каждый

символ открытого текста заменяется символом, находящимся на некотором, определяемом ключом, постоянном числе позиций левее или правее него в алфавите.

**Шифрование:** Каждая буква в исходном сообщении заменяется на другую букву, сдвинутую на фиксированное количество позиций в алфавите. Например, при сдвиге на 3, буква "A" заменяется на "D", "B" на "E" и так далее. Алфавит считается цикличным, поэтому после "Z" снова идёт "A". Для дешифрования достаточно выполнить сдвиг в обратную сторону на то же количество позиций. Шифр Цезаря легко подвержен атаке полным перебором, так как существует всего 25 возможных сдвигов (для английского алфавита).

**Реализация программы шифрования и дешифрования:**

import java.util.Scanner;  
  
public class CaesarEncryption {  
 private String alphabet;  
 private String sourceText;  
 private int shift;  
 private String encryption;  
  
 public void startProgram() {  
 chooseAlphabet();  
 setSourceText();  
 processEncryption();  
 System.*out*.println("Result: " + this.encryption);  
 }  
  
 public void chooseAlphabet() {  
 System.*out*.println("Choose the alphabet type:");  
 System.*out*.println("1 - Русский\n2 - English");  
 Scanner scanner = new Scanner(System.*in*);  
 int n = scanner.nextInt();  
 if (n > 2 || n < 1) {  
 System.*out*.println("Incorrect data");  
 System.*exit*(0);  
 }  
 if (n == 1) {  
 this.alphabet = "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ";  
 } else {  
 this.alphabet = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";  
 }  
 }  
  
 public void setSourceText() {  
 System.*out*.println("Enter the text that you want to encrypt:");  
 Scanner scanner = new Scanner(System.*in*);  
 this.sourceText = scanner.nextLine();  
 }  
  
 public void processEncryption() {  
 this.setShift();  
 String encryptionText = "";  
 for (char c : this.sourceText.toCharArray()) {  
 if (c == ' ') {  
 encryptionText += c;  
 continue;  
 }  
 int index = 0;  
 for (int i = 0; i < this.alphabet.length(); i++) {  
 if (c == this.alphabet.charAt(i)) {  
 index = (i + this.shift) % this.alphabet.length();  
 break;  
 }  
 }  
 encryptionText += this.alphabet.charAt(index);  
 }  
 this.encryption = encryptionText;  
 }  
  
 public void setShift() {  
 System.*out*.print("Enter shift: ");  
 Scanner scanner = new Scanner(System.*in*);  
 this.shift = scanner.nextInt();  
 while (this.shift > this.alphabet.length() - 1) {  
 System.*out*.println("Going beyond the boundaries of the alphabet");  
 this.shift = scanner.nextInt();  
 }  
 }  
  
 public void decoding() {  
 for (int i = this.alphabet.length(); i > 0; i--) {  
 String decode\_str = "";  
 for (int j = 0; j < this.encryption.length(); j++) {  
 if (this.encryption.charAt(j) == ' ') {  
 decode\_str += ' ';  
 continue;  
 }  
 int index\_alphabet = this.alphabet.indexOf(this.encryption.charAt(j));  
 decode\_str += this.alphabet.charAt((index\_alphabet + i - 1)%this.alphabet.length());  
 }  
 System.*out*.println(decode\_str);  
 if (decode\_str.equals(this.sourceText)) {  
 System.*out*.println("Shift: " + (this.alphabet.length() - (i - 1)));  
 return;  
 }  
 }  
 }  
}

**Описание основных функций программы:**

1. **public void startProgram()**

Начинает запуск программы, выполняет все основные функции.

1. **public void chooseAlphabet()**

Предоставляет возможность выбора алфавита. В случае некорректного выбора завершает работу программы.

1. **public void setSourceText()**

Запрашивает исходный текст, который необходимо зашифровать.

1. **public void setShift()**

Запрашивает у пользователя сдвиг шифра.

1. **public void processEncryption()**

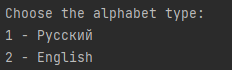
Проходит по каждому символу исходной строки и определяет позицию данного символа в алфавите. Затем добавляет сдвиг, в случае выхода его за пределы, отбрасывает предел, до заданного диапазона. Определяет новый символ алфавита и записывает его в новую строку. В случае пустого символа, переносит его в новую строку.

1. **public void decoding()**

Дешифрует шифр, до тех пор, пока не получит исходный текст. Выводит возможные результаты расшифровки, пока не дойдёт до исходного текста. А в случае его отсутствия, пока не обойдёт все варианты.

**Пример работы:**

Зашифруем фразу “ CAESAR CODE

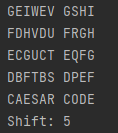


Выбираем алфавит, который предпочитаем пользователь.



Вводим сообщение, которое желаем зашифровать.

  
Задаём сдвиг шифра.  
  
Получаем зашифрованное сообщение.

  
Варианты декодирования программы, до тех пор, пока не встретится исходный текст. Определение сдвига.

**Криптографическая стойкость: сильные и слабые стороны  
Сильные стороны:**

* **Простота реализации:** Шифр Цезаря легко реализуется и понимается. Его алгоритм прост и требует минимальных вычислительных ресурсов.
* **Небольшая вычислительная нагрузка:** Поскольку шифр основан на простом сдвиге, его выполнение требует очень мало времени и ресурсов.

**Слабые стороны:**

* **Малое количество ключей:** Шифр Цезаря использует ограниченное количество сдвигов, что делает его уязвимым к атакам методом полного перебора. Взломать его можно за считанные секунды.
* **Отсутствие защиты от частотного анализа:** Шифр Цезаря сохраняет частотные характеристики исходного текста. Это позволяет злоумышленнику использовать частотный анализ для расшифровки сообщения, выявляя наиболее часто встречающиеся символы и их соответствующие замены.

Для улучшения стойкости метода шифрования можно включить элемент перестановки (транспозиции) для перемешивания символов после применения шифра Цезаря. Это добавляет дополнительный уровень сложности.

**Алгоритм:**

1. **Применение шифра Цезаря:** Применяем шифр Цезаря для исходного текста.
2. **Определение таблицы перестановки:**
   * Создаём таблицу, в которой каждый столбец представляет собой новый порядок символов. Например, если длина текста 5, таблица может быть [4, 1, 3, 0, 2], что означает, что символы на позициях 0 и 1 будут перемещены на места 4 и 1 соответственно.
3. **Расшифрование:**
   * Сначала восстановите порядок символов в [0,1,2,3,4].
   * Затем примените метод полного перебора до получения корректного результата.