**Университет ИТМО**

Факультет программной инженерии и компьютерной техники

**Отчет по лабораторной работе № 1**  
Информационная безопасность

Вариант 8

Выполнил студент группы P34302

Ким Даниил Кванхенович

Проверил преподаватель

Рыбаков Степан Дмитриевич

Санкт-Петербург 2023

Содержание

[Цель работы 3](#_Toc148450965)

[Постановка задачи: 3](#_Toc148450966)

[Порядок выполнения работы: 3](#_Toc148450967)

[Выполнение: 4](#_Toc148450968)

[Описание алгоритма: 4](#_Toc148450969)

[Описание разработанного алгоритма: 5](#_Toc148450970)

[Алфавит: 5](#_Toc148450971)

[Вспомогательные методы модуля *AffineCipher*: 6](#_Toc148450972)

[Основные методы модуля AffineCipher: 6](#_Toc148450973)

[Листинг разработанного модуля: 7](#_Toc148450974)

[Вспомогательные методы модуля AffineCipher: 7](#_Toc148450975)

[Основные методы модуля AffineCipher: 8](#_Toc148450976)

[Вывод программы 10](#_Toc148450977)

[Частотный анализ 11](#_Toc148450978)

[Вывод 12](#_Toc148450979)

Цель работы**:**

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

# Постановка задачи:

**Вариант – 8:**

Реализовать в программе шифрование и дешифрацию файла с использованием аффинной криптосистемы. Провести частотный анализ зашифрованного файла, осуществляя проверку по файлу с набором ключевых слов.

# Порядок выполнения работы:

1. Ознакомьтесь с теоретическими основами шифрования данных.
2. Получите вариант задания у преподавателя.
3. Напишите программу согласно варианту задания.
4. Отладьте разработанную программу и покажите результаты работы программы преподавателю.
5. Составьте отчет по лабораторной работе.

# Выполнение:

## Описание алгоритма:

Аффинное шифрование заключается в замещении символов открытого текста в соответствие с определёнными правилами. Алгоритм можно разделить на несколько шагов:

* Символу открытого текста сопоставляется число – номер символа в алфавите.
* Число шифруется с помощью простой математической функции  
   – функции шифрования ***E(x)***.
* Получившееся число преобразуется обратно в букву.

Шифрование производится с помощью ключей **A** и **Б**. На ключ А накладывается ограничение:   
**А** и **М** должны быть взаимно простыми числами, где **М** – мощность используемого алфавита.

**Функция шифрования**:

**Функция расшифрования**:

– **обратное к A число по модулю M**, т.е. оно удовлетворяет уравнению:

## Описание разработанного алгоритма:

Модуль разработан на языке программирования Java и состоит из основного класса *AffineCipher* и вспомогательного типа перечисления *AlphabetType*.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

### Алфавит:

Т.к. результат работы непосредственно связан с тем, как будет задан алфавит, была реализована возможность шифрования и расшифрования текста с использованием настраиваемого алфавита.

Алфавит представлен в виде динамического массива. В зависимости от параметров *AlphabetType* поданных при инициализации экземпляра *AffineCipher*, массив будет дополнен соответствующим набором символов.



Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

### Вспомогательные методы модуля *AffineCipher*:

* **E** и **D** – функции шифрования и расшифрования соответственно
* **areComprime** – функция проверки взаимно простоты двух чисел
* **getModularMultiplicativeInverse –** функция расчета числа обратного по модулю

### Основные методы модуля AffineCipher:

* **encrypt** – публичный метод для преобразования открытого текста в шифротекст
* **decrypt –** публичный метод для преобразования зашифрованного текста в исходный текст
* **encryptSerial** & **decryptSerial -** аналогичны обычным **encrypt** и **decrypt**, но шифрование и расшифровывание производится последовательно

## Листинг разработанного модуля:

### Вспомогательные методы модуля AffineCipher:

* Функция шифрования *E(x)*:

private int E(int x, int A, int B) {  
 return (A\*x + B) % M;  
}

* Функция расшифрования *D(x)*:

private int D(int x, int A, int B) {  
 return ((x+ M - B) \* A) % M;  
}

* Функция проверки взаимно простоты двух чисел:

public boolean areCoprime(int digit1, int digit2) {  
 int lowerDigit = Math.*abs*(Math.*min*(digit1, digit2));  
 int greaterDigit = Math.*abs*(Math.*max*(digit1, digit2));  
  
 if (lowerDigit == 0 || greaterDigit == 0)  
 return false;  
  
 if (greaterDigit % lowerDigit == 0 && lowerDigit != 1)  
 return false;  
  
 for (int i = 2; i < lowerDigit; i++) {  
 if (lowerDigit % i == 0 && greaterDigit % i == 0)  
 return false;  
 }  
  
 return true;  
}

* Функция расчета числа обратного по модулю:

private int getModularMultiplicativeInverse (int x, int modular) {  
 int result = 0;  
 for (int i = 1; i <= modular; i++) {  
 if (x\*i % modular == 1) {  
 result = i;  
 break;  
 }  
 }  
 return result;  
}

### Основные методы модуля AffineCipher:

* Публичный метод для преобразования открытого текста в шифротекст:

public String encrypt(String text, int keyA, int keyB) {  
 */\* STEP 0 \*/* if (null == text) {  
 return null;  
 }  
  
 if (!areCoprime(keyA, M)) {  
 return null;  
 }  
  
 */\* STEP 1 \*/* List<Integer> indexes = text.chars()  
 .mapToObj(o -> (char) o)  
 .filter(ALPHABET::contains)  
 .map(ALPHABET::indexOf)  
 .toList();  
  
 */\* STEP 2 \*/* List<Integer> encryptedIndexes = indexes.stream()  
 .map(index -> E(index, keyA, keyB))  
 .toList();  
  
 */\* STEP 3 \*/* StringBuilder stringBuilder = new StringBuilder();

encryptedIndexes.stream()  
 .map(ALPHABET::get)  
 .forEach(stringBuilder::append);

String cipherText = stringBuilder.toString();  
  
 return cipherText;  
}

* Публичный метод для преобразования зашифрованного текста в исходный текст:

public String decrypt(String text, int keyA, int keyB) {  
 */\* STEP 0 \*/* if (null == text) {  
 return null;  
 }  
  
 if (!areCoprime(keyA, M)) {  
 return null;  
 }  
  
 */\* STEP 1 \*/* List<Integer> indexes = text.chars()  
 .mapToObj(o -> (char) o)  
 .filter(ALPHABET::contains)  
 .map(ALPHABET::indexOf)  
 .toList();;  
  
 */\* STEP 2 \*/* int keyAModularMultiplicativeInverse = getModularMultiplicativeInverse(keyA, M);

List<Integer> decryptedIndexes = indexes.stream()  
 .map(index -> D(index, keyAModularMultiplicativeInverse, keyB))  
 .toList();  
  
 */\* STEP 3 \*/* StringBuilder stringBuilder = new StringBuilder();

decryptedIndexes  
 .forEach(index -> stringBuilder.append(ALPHABET.get(index)));

String openText = stringBuilder.toString();  
  
 return openText;  
}

Вывод программы**:**

* Шифрация и дешифрация строки *“attack at dawn”* с использованием ключа ***A*** *= 3* и ключа ***B*** *= 4*:  
    
  **Изображение выглядит как текст, снимок экрана, Шрифт, число

  Автоматически созданное описание**
* Шифрация и дешифрация строки *“hello world”* с использованием ключа ***A*** *= 3* и ключа ***B*** *= 4*:  
    
  **Изображение выглядит как текст, снимок экрана, Шрифт, число

  Автоматически созданное описание**
* Шифрация и дешифрация строки *“telecommunications”* с использованием ключа ***A*** *= 1* и ключа ***B*** *= 0*:

**Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание**

Частотный анализ**:**

Открытый текст:

Частотный анализ – это один из методов криптоанализа,  
основывающийся на предположении о существовании нетривиального статистического распределения  
отдельных символов и их последовательностей как в открытом тексте,  
так и шифрованном тексте, которое с точностью до замены символов будет сохраняться в процессе  
шифрования и дешифрования. Кратко говоря, частотный анализ предполагает,  
что частота появления заданной буквы алфавита в достаточно длинных текстах одна и та же для разных текстов одного языка.  
При этом в случае моно алфавитного шифрования,  
если в шифрованном тексте будет символ с аналогичной вероятностью появления,  
то можно предположить, что он и является указанной зашифрованной буквой.  
Аналогичные рассуждения применяются к биграммам (двухбуквенным последовательностям),  
триграммам в случае поли алфавитных шифров.  
Метод частотного анализа известен с еще IX-го века и связан и именем Ал-Кинди.  
Но наиболее известным случаем применения такого анализа является дешифровка египетских иероглифов  
Ж.-Ф. Шампольоном в 1822 году.  
Данный вид анализа основывается на том, что текст состоит из слов,  
а слова из букв. Количество различных букв в каждом языке ограничено и буквы могут быть просто перечислены.  
Важными характеристиками текста являются повторяемость букв, пар букв (биграмм) и вообще m-ок (m-грамм),  
сочетаемость букв друг с другом, чередование гласных и согласных и некоторые другие.  
Идея состоит в подсчете чисел вхождений каждой nm возможных m-грамм в достаточно длинных открытых текстах,  
составленных из букв алфавита a1, a2, ..., an. При этом просматриваются подряд идущие m-граммы текста:

Зашифрованный текст:

сгбзQзJfh гJгvaЩ – tзQ QюaJ aЩ CЕзQюQр o4aXзQгJгvaЩг,  
QбJQрfргAЧahбH Jг X4ЕюXQvQТЕJaa Q боЧЕбзрQргJaa JЕз4aрaгvmJQчQ бзгзaбзaЙЕбoQчQ 4гбX4ЕюЕvЕJaH  
QзюЕvmJfь бaCрQvQр a aь XQбvЕюQргзЕvmJQбзЕh oгo р Qзo4fзQC зЕoбзЕ,  
згo a Рaх4QргJJQC зЕoбзЕ, oQзQ4QЕ б зQЙJQбзmA юQ ЩгCЕJf бaCрQvQр йоюЕз бQь4гJHзmбH р X4QГЕббЕ  
Рaх4QргJaH a юЕРaх4QргJaH. Х4гзoQ чQрQ4H, ЙгбзQзJfh гJгvaЩ X4ЕюXQvгчгЕз,  
ЙзQ ЙгбзQзг XQHрvЕJaH ЩгюгJJQh йоoрf гvхгрaзг р юQбзгзQЙJQ юvaJJfь зЕoбзгь QюJг a зг ТЕ юvH 4гЩJfь зЕoбзQр QюJQчQ HЩfoг.  
y4a tзQC р бvоЙгЕ CQJQ гvхгрaзJQчQ Рaх4QргJaH,  
Ебva р Рaх4QргJJQC зЕoбзЕ йоюЕз бaCрQv б гJгvQчaЙJQh рЕ4QHзJQбзmA XQHрvЕJaH,  
зQ CQТJQ X4ЕюXQvQТaзm, ЙзQ QJ a HрvHЕзбH оoгЩгJJQh ЩгРaх4QргJJQh йоoрQh.  
OJгvQчaЙJfЕ 4гббоТюЕJaH X4aCЕJHAзбH o йaч4гCCгC (юроьйоoрЕJJfC XQбvЕюQргзЕvmJQбзHC),  
з4aч4гCCгC р бvоЙгЕ XQva гvхгрaзJfь Рaх4Qр.  
dЕзQю ЙгбзQзJQчQ гJгvaЩг aЩрЕбзЕJ б ЕЧЕ ЭЖ-чQ рЕoг a брHЩгJ a aCЕJЕC Ov-ХaJюa.  
kQ JгaйQvЕЕ aЩрЕбзJfC бvоЙгЕC X4aCЕJЕJaH згoQчQ гJгvaЩг HрvHЕзбH юЕРaх4Qрoг ЕчaXЕзбoaь aЕ4QчvaхQр  
ъ.-7. шгCXQvmQJQC р cZjj чQюо.  
ёгJJfh рaю гJгvaЩг QбJQрfргЕзбH Jг зQC, ЙзQ зЕoбз бQбзQaз aЩ бvQр,  
г бvQрг aЩ йоoр. ХQvaЙЕбзрQ 4гЩvaЙJfь йоoр р oгТюQC HЩfoЕ Qч4гJaЙЕJQ a йоoрf CQчоз йfзm X4QбзQ XЕ4ЕЙaбvЕJf.  
2гТJfCa ьг4гoзЕ4aбзaoгCa зЕoбзг HрvHAзбH XQрзQ4HЕCQбзm йоoр, Xг4 йоoр (йaч4гCC) a рQQйЧЕ Д-Qo (Д-ч4гCC),  
бQЙЕзгЕCQбзm йоoр ю4оч б ю4очQC, ЙЕ4ЕюQргJaЕ чvгбJfь a бQчvгбJfь a JЕoQзQ4fЕ ю4очaЕ.  
ЗюЕH бQбзQaз р XQюбЙЕзЕ ЙaбЕv рьQТюЕJah oгТюQh КД рQЩCQТJfь Д-ч4гCC р юQбзгзQЙJQ юvaJJfь Qзo4fзfь зЕoбзгь,  
бQбзгрvЕJJfь aЩ йоoр гvхгрaзг pc, pj, ..., pК. y4a tзQC X4QбCгз4aргAзбH XQю4Hю aюоЧaЕ Д-ч4гCCf зЕoбзг:

|  |  |
| --- | --- |
| **Слов в открытом тексте** | **Слов в шифротексте** |
| 223 | 223 |

Наиболее часто встречающиеся слова:

|  |  |  |  |
| --- | --- | --- | --- |
| **Слово в открытом тексте** | **Кол-во** | **Слово в шифротексте** | **Кол-во** |
| и | 11 | a | 11 |
| в | 10 | р | 10 |
| букв | 6 | йоoр | 6 |
| из | 4 | aщ | 4 |
| m | 4 | б | 4 |
| c | 4 | д | 4 |
| тексте | 3 | гjгvaщг | 3 |
| … | | | |

Наиболее часто встречающиеся словосочетания:

|  |  |  |  |
| --- | --- | --- | --- |
| **В открытом тексте** | **Кол-во** | **В шифротексте** | **Кол-во** |
| в достаточно длинных | 2 | г щжшnюnжмэж щжхээьф | 2 |
| достаточно длинных | 2 | щжшnюnжмэж щжхээьф | 2 |
| в случае | 2 | г шжчмюа | 2 |
| при этом | 2 | aнх бnжс | 2 |
| шифрованном тексте | 2 | ехйнжгюээжс nаншnа | 2 |
| из букв | 2 | хк ичнг | 2 |
| в достаточно | 2 | г щжшnюnжмэж | 2 |
| … | | | |

Вывод**:**

В ходе данной лабораторной работы был закреплен материал об алгоритмах симметричного шифрования, о влиянии ключа на криптостойкость алгоритмах простой замены и об их криптостойкости.

Был создан программный модуль на языке программирования Java реализующий алгоритм аффинного шифрования.

Так же получен опыт использования частотного анализа.