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:

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 **А** и **Б**. На ключ А накладывается ограничение:
А и **М** должны быть взаимно простыми числами, где **М** – мощность используемого алфавита.

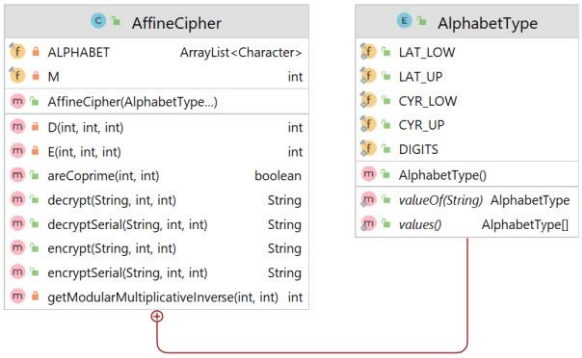
Функция шифрования: $E(x) = (Ax + B) \bmod M$

Функция расшифрования: $D(x) = A^{-1}(x - B) \bmod M$

A^{-1} – **обратное к А число по модулю М**, т.е. оно удовлетворяет уравнению: $1 = AA^{-1} \bmod M$

Описание разработанного алгоритма:

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



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.

```
private final ArrayList<Character> ALPHABET;

public AffineCipher (AlphabetType ... alphabetTypes) {
    this.ALPHABET = new ArrayList<>();

    for (AlphabetType type : alphabetTypes) {
        switch (type) {
            case LAT_LOW -> {
                for (char a = 'a'; a <= 'z'; a++) ALPHABET.add(a);
            }
            case LAT_UP -> {
                for (char a = 'A'; a <= 'Z'; a++) ALPHABET.add(a);
            }
            case CYR_LOW -> {
                for (char a = 'а'; a <= 'е'; a++) ALPHABET.add(a);
                ALPHABET.add('ё');
                for (char a = 'ж'; a <= 'я'; a++) ALPHABET.add(a);
            }
            case CYR_UP -> {
                for (char a = 'А'; a <= 'Е'; a++) ALPHABET.add(a);
                ALPHABET.add('Ё');
                for (char a = 'Ж'; a <= 'Я'; a++) ALPHABET.add(a);
            }
            case DIGITS -> {
                for (char a = '0'; a <= '9'; a++) ALPHABET.add(a);

            }
        }
    }
```

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*:

```
[INIT] ALPHABET:          [a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z]
                           0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[MAIN] Input the key A: 3
[MAIN] Input the key B: 4

[ENCRYPTION] OPEN TEXT:         [a  t  t  a  c  k  a  t  d  a  w  n]
[ENCRYPTION] MAPPED LETTERS:    [0  19 19 0  2  10 0  19 3  0  22 13]
[ENCRYPTION] ENCRYPTED LETTERS: [4  9  9  4  10 8  4  9  13 4  18 17]
[ENCRYPTION] CIPHER TEXT:       [e  j  j  e  k  i  e  j  n  e  s  r]

[DECRYPTION] CIPHER TEXT:       [e  j  j  e  k  i  e  j  n  e  s  r]
[DECRYPTION] MAPPED LETTERS:    [4  9  9  4  10 8  4  9  13 4  18 17]
[DECRYPTION] DECRYPTED LETTERS: [0  19 19 0  2  10 0  19 3  0  22 13]
[DECRYPTION] OPEN TEXT:         [a  t  t  a  c  k  a  t  d  a  w  n]
```

- Шифрация и дешифрация строки *“hello world”* с использованием ключа ***A*** = *3* и ключа ***B*** = *4*:

```
[INIT] ALPHABET:          [a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z]
                           0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[MAIN] Input the key A: 3
[MAIN] Input the key B: 4

[ENCRYPTION] OPEN TEXT:         [h  e  l  l  o  w  o  r  l  d]
[ENCRYPTION] MAPPED LETTERS:    [7  4  11 11 14 22 14 17 11 3]
[ENCRYPTION] ENCRYPTED LETTERS: [25 16 11 11 20 18 20 3  11 13]
[ENCRYPTION] CIPHER TEXT:       [z  q  l  l  u  s  u  d  l  n]

[DECRYPTION] CIPHER TEXT:       [z  q  l  l  u  s  u  d  l  n]
[DECRYPTION] MAPPED LETTERS:    [25 16 11 11 20 18 20 3  11 13]
[DECRYPTION] DECRYPTED LETTERS: [7  4  11 11 14 22 14 17 11 3]
[DECRYPTION] OPEN TEXT:         [h  e  l  l  o  w  o  r  l  d]
```

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

```
[INIT] ALPHABET:          [a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z]
                           0  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[MAIN] Input the key A: 1
[MAIN] Input the key B: 0

[ENCRYPTION] OPEN TEXT:         [t  e  l  e  c  o  m  m  u  n  i  c  a  t  i  o  n  s]
[ENCRYPTION] MAPPED LETTERS:    [19 4  11 4  2  14 12 12 20 13 8  2  0  19 8  14 13 18]
[ENCRYPTION] ENCRYPTED LETTERS: [19 4  11 4  2  14 12 12 20 13 8  2  0  19 8  14 13 18]
[ENCRYPTION] CIPHER TEXT:       [t  e  l  e  c  o  m  m  u  n  i  c  a  t  i  o  n  s]

[DECRYPTION] CIPHER TEXT:       [j  q  l  q  k  u  o  o  m  r  c  k  e  j  c  u  r  g]
[DECRYPTION] MAPPED LETTERS:    [9  16 11 16 10 20 14 14 12 17 2  10 4  9  2  20 17 6]
[DECRYPTION] DECRYPTED LETTERS: [9  16 11 16 10 20 14 14 12 17 2  10 4  9  2  20 17 6]
[DECRYPTION] OPEN TEXT:         [j  q  l  q  k  u  o  o  m  r  c  k  e  j  c  u  r  g]
```

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 rJrvaЩ – tзQ QюaJ aЩ CEзQюQp o4aXзQrJrvaЩr,
QбJQpfprAЧahбH Jr X4EюXQvQTEJaa Q боЧЕбзрQргJaa JEз4арагvmJQчQ бзгзабзаЙЕбоQчQ
4гбX4EюEvEJaH
QзюEvmJfь баСрQvQр а аь XQбvEюQргзEvmJQбзEh ого р Qзо4fзQС зЕобзЕ,
зго а Рах4QргJJQС зЕобзЕ, оQзQ4QЕ б зQЙJQбзmА юQ ЩгСЕJf баСрQvQр йоюЕз бQь4гJНзmбН р
X4QГЕббЕ
Рах4QргJaH а юЕРах4QргJaH. X4гзоQ чQрQ4H, ЙгбзQзJfh rJrvaЩ X4EюXQvгчгЕз,
ЙзQ ЙгбзQзг XQHpvEJaH ШгюгJJQh йоорf rvxгразг р юQбзгзQЙJQ ювaJJfь зЕобзгь QюJг а зг ТЕ
ювН 4гЩJfь зЕобзQр QюJQчQ НЩfог.
у4а tзQС р бvoЙгЕ CQJQ rvxгразJQчQ Рах4QргJaH,
Ебva р Рах4QргJJQС зЕобзЕ йоюЕз баСрQv б rJrvQчаЙJQh рЕ4QHзJQбзmА XQHpvEJaH,
зQ CQTJQ X4EюXQvQTазm, ЙзQ QJ а HpvHЕзбН оогЩгJJQh ЩгРах4QргJJQh йоорQh.
OJrvQчаЙJfЕ 4гббоТюЕJaH X4aCEJHАзбН о йач4гССгС (юроьйоорEJJfC XQбvEюQргзEvmJQбзНС),
з4ач4гССгС р бvoЙгЕ XQva rvxгразJfь Рах4Qр.
dЕзQю ЙгбзQзJQчQ rJrvaЩr аЩрЕбзЕJ б ЕЧЕ ЭЖ-чQ рЕог а брНЩгJ а аCEJEC Ov-ХаJюа.
kQ JгайQvЕЕ аЩрЕбзJfС бvoЙгЕС X4aCEJEJaH згоQчQ rJrvaЩr HpvHЕзбН юЕРах4Qрог ЕчаХЕзбоаь
аЕ4QчvахQр
ъ.-7. шгСXQvmQJQС р cZjj чQюо.
ёгJJfh раю rJrvaЩr QбJQpfprЕзбН Jr зQС, ЙзQ зЕобз бQбзQаз аЩ бvQр,
г бvQрг аЩ йоор. XQvaЙЕбзрQ 4гЩvaЙJfь йоор р огТюQС НЩfоЕ Qч4гJаЙЕJQ а йоорf CQчоз йfзm
X4QбзQ ХЕ4ЕЙабvEJf.
2гТJfСа ьг4гозЕ4абзаогСа зЕобзг HpvHАзбН XQрзQ4НЕСQбзm йоор, Хг4 йоор (йач4гСС) а рQQйЧЕ
Д-Qо (Д-ч4гСС),
бQЙЕзгЕСQбзm йоор ю4оч б ю4очQС, ЙЕ4ЕюQргJaЕ чvгбJfь а бQчvгбJfь а JЕоQзQ4fЕ ю4очаЕ.
ЗюЕН бQбзQаз р XQюбЙЕзЕ ЙабЕv рьQТюЕJah огТюQh КД рQЩCQTJfь Д-ч4гСС р юQбзгзQЙJQ ювaJJfь
Qзо4fзfь зЕобзгь,
бQбзгрvEJJfь аЩ йоор rvxгразг рс, рj, ..., рК. у4а tзQС X4QбСгз4аргАзбН XQю4Ню аюоЧаЕ Д-
ч4гССf зЕобзг:
```

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	а	11
в	10	р	10
букв	6	йоор	6
из	4	ащ	4
m	4	б	4
с	4	д	4
тексте	3	гjгvащг	3
…			

Наиболее часто встречающиеся словосочетания:

В открытом тексте	Кол-во	В шифротексте	Кол-во
в достаточно длинных	2	г щжшпюпжмэж щжхээьф	2
достаточно длинных	2	щжшпюпжмэж щжхээьф	2
в случае	2	г шжчмюа	2
при этом	2	анх бпжс	2
шифрованном тексте	2	ехйнжгюээжс паншпа	2
из букв	2	хк ичнг	2
в достаточно	2	г щжшпюпжмэж	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.