****

**Московский авиационный институт**

**(Национальный исследовательский университет)**

**Институт № 3**

**Кафедра 311**

**Информационная безопасность**

**Лабораторная работа № 4**

**«Шифр Виженера (взлом)»**

**Выполнил студент  
Максимов Евгений Викторович**

**Группа М3З-302-БК**

**Дата 19.10.2022 г.**

**Принял преподаватель  
Кос Оксана Игоревна**

Оглавление

[Цель лабораторной работы 3](#_Toc89793446)

[Глава 1. Принцип Шифра Виженера. 4](#_Toc89793447)

[Глава 2. Взлом Шифра Виженера. 5](#_Toc89793448)

[Глава 2. Нахождение ключа Методом Касиски 7](#_Toc89793449)

[Глава 3. Листинг программы нахождения длины ключа. 8](#_Toc89793450)

[Глава 4. Нахождение ключа по его длине. 10](#_Toc89793451)

[Глава 5. Листинг программы поиска ключа. 11](#_Toc89793452)

[Глава 6. Расшифровка текста. 13](#_Toc89793453)

[Итоги лабораторной работы 15](#_Toc89793454)

[Список литературы 16](#_Toc89793455)

# Цель лабораторной работы

Изучить:

* 1. Взлом Шифра Виженера.

# Глава 1. Принцип Шифра Виженера.

Шифр Виженера — это метод полиалфавитного шифрования буквенного текста с использованием ключевого слова. Шифр Виженера состоит из последовательности нескольких шифров Цезаря с различными значениями сдвига. Для зашифровывания может использоваться таблица алфавитов, называемая tabula recta или квадрат (таблица) Виженера. Применительно к латинскому алфавиту таблица Виженера составляется из строк по 26 символов, причём каждая следующая строка сдвигается на несколько позиций.

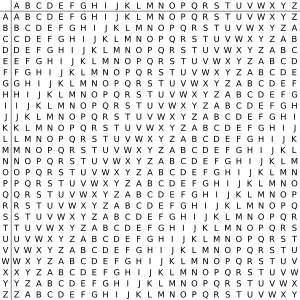


Рисунок 1.1 – Квадрат Виженера

# Глава 2. Взлом Шифра Виженера.

**Задача**: *найти ключ и взломать зашифрованный текст.*

**Зашифрованный текст:** «»

# Глава 2. Нахождение ключа Методом Касиски

Идея метода основана на том, что ключи являются периодическими, а в естественном языке существуют часто встречающиеся буквосочетания: биграммы и триграммы. Это наводит на мысль, что повторяющиеся наборы символов в шифротексте — повторения популярных биграмм и триграмм исходного текста.

Метод Касиски заключается в поиске групп символов, которые повторяются в зашифрованном тексте. Группы должны состоять из не менее чем трех символов. Тогда расстояния между последовательными возникновениями групп, вероятно, будут кратны длине ключевого слова. Предполагаемая длина ключевого слова кратна наибольшему общему делителю всех расстояний.

Расстояние между двумя повторяющимися подстроками в зашифрованном тексте g. Ключевое слово длиной k повторяется, чтобы заполнить длину зашифрованного текста, расстояние g кратно длине ключевого слова k. Таким образом, если мы видим две повторяющиеся подстроки с расстоянием g, то один из делителей g может быть длиной ключевого слова. Например, если расстояние равно g = 18, поскольку делители g равны 2, 3, 6, 9 и 18, один из них может быть длиной неизвестного ключевого слова. (Рисунок 2.1)

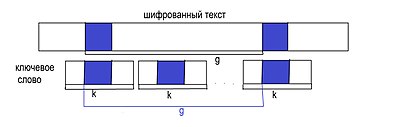


Рисунок 2.1 – идея метода Касиски

# Глава 3. Листинг программы нахождения длины ключа.

*Поиск повторяющихся подстрок*

Set<Integer> repeatsStarts = new TreeSet<>();  
Map<String, Set<Integer>> substring2repeats = new TreeMap<>();  
String uncheckedText = encryptedString;  
int maxRepeatedSubstringLength = 0;  
int repeatIndx;  
int checkedTextLength = 0;  
  
for (int substringLength = 10; substringLength >= 3; substringLength--) {  
 int substringStart = 0;  
 int substringEnd = substringLength;  
 while (substringEnd != encryptedString.length()) {  
 String substring = encryptedString.substring(substringStart, substringEnd);  
 int repeatInUncheckedTextIdx = uncheckedText.indexOf(substring);  
 if (isSubstringExist(repeatInUncheckedTextIdx)) {  
 repeatIndx = repeatInUncheckedTextIdx + checkedTextLength;  
 if (isNewRepeatIndx(repeatsStarts, repeatIndx)) {  
 repeatsStarts.add(repeatIndx);  
 uncheckedText = uncheckedText.substring(repeatInUncheckedTextIdx + substringLength);  
 checkedTextLength = repeatIndx + substringLength;  
 } else {  
 substringStart++;  
 substringEnd++;  
 }  
 } else {  
 substringStart++;  
 substringEnd++;  
 uncheckedText = encryptedString;  
 checkedTextLength = 0;  
 if (repeatsStarts.size() > 1) {  
 substring2repeats.put(substring, new TreeSet<>(repeatsStarts));  
 maxRepeatedSubstringLength = max(maxRepeatedSubstringLength, substringLength);  
 }  
 repeatsStarts.clear();  
 }  
 }  
}

*Расчет длины между повторами*

Map<Integer, Map<Integer, List<Integer>>> length2factors = new TreeMap<>(reverseOrder());  
Set<Integer> distances = new TreeSet<>();  
substring2repeats.forEach((key, value) -> {  
 Integer[] repeatStarts = value.toArray(new Integer[0]);  
 int substringLength = key.length();  
 Map<Integer, List<Integer>> distance2factors = new HashMap<>();  
 List<Integer> factors = new ArrayList<>();  
 for (int i = 1; i < repeatStarts.length; i++) {  
 int current = repeatStarts[i];  
 int prev = repeatStarts[i - 1];  
 int distance = current - prev;  
 for (int factor = 3; factor <= 20; factor++) { // Полагаем, что ключ, длиной 1 и 2 слишком мал и не больше 20  
 if (distance % factor == 0) {  
 factors.add(factor);  
 }  
 }  
 distances.add(distance);  
 distance2factors.put(distance, factors);  
 }  
 if (length2factors.containsKey(substringLength)) {  
 Map<Integer, List<Integer>> existingDistance2Factors = length2factors.get(substringLength);  
 distance2factors.forEach((distance, newFactors) -> {  
 if (existingDistance2Factors.containsKey(distance)) {  
 existingDistance2Factors.get(distance).addAll(distance2factors.get(distance));  
 existingDistance2Factors.put(distance, existingDistance2Factors.get(distance).stream().distinct().collect(toList()));  
 } else {  
 existingDistance2Factors.put(distance, newFactors);  
 }  
 });  
 length2factors.put(substringLength, existingDistance2Factors);  
 } else {  
 length2factors.put(substringLength, new TreeMap<>(distance2factors));  
 }  
});

*Нахождение НОД*

private static Integer findNod(Set<Integer> distances) {  
 Integer[] distArray = distances.toArray(new Integer[0]);  
 List<Integer> pair = asList(distArray[0], distArray[1]);  
 int nod = findPairNod(pair);  
 for (int i = 2; i < distArray.length; i++) {  
 if (1 == findPairNod(asList(nod, distArray[i]))) {  
 continue;  
 }  
 nod = findPairNod(asList(nod, distArray[i]));  
 }  
  
 return nod;  
}  
  
private static int findPairNod(List<Integer> pair) {  
 int nod;  
 int first = max(pair.get(0), pair.get(1));  
 int second = min(pair.get(0), pair.get(1));  
 int remainder;  
 while (first % second != 0) {  
 remainder = first % second;  
 first = second;  
 second = remainder;  
 }  
 nod = second;  
  
 return nod;  
}

*Найденная длина ключа: 5.*

# Глава 4. Нахождение ключа по его длине.

Зная длину ключа (n), необходимо разделить шифротекст (x) на n-групп:

x1…

xn+1…

xn+2…

xn…

Затем каждую группу необходимо подвергнуть частотному анализу, полагая, что самая частая буква будет равна самой частовстречающейся букве алфавита (для английского – буква «Е»). Далее осуществляется поиск сдвига относительно алфавита и вычисляется буква ключа (взлом Шифра Цезаря).

# Глава 5. Листинг программы поиска ключа.

*Деление шифротекста на блоки*

List<String> blocks = new ArrayList<>();  
for (int startIndex = 0; startIndex < encrypted.length(); startIndex += keyLength) {  
 int endIndex = Math.min(startIndex + keyLength, encrypted.length());  
 String block = encrypted.substring(startIndex, endIndex);  
 blocks.add(block);  
}  
  
System.out.println("Individual blocks are:");  
blocks.forEach(System.out::println);

*Частотный алализ*

List<Map<Character, Integer>> digitToCounts = Stream.generate(HashMap<Character, Integer>::new)  
 .limit(keyLength)  
 .collect(Collectors.toList());  
  
for (String block : blocks) {  
 for (int i = 0; i < block.length(); i++) {  
 char c = block.charAt(i);  
 Map<Character, Integer> counts = digitToCounts.get(i);  
 counts.compute(c, (character, count) -> count == null ? 1 : count + 1);  
 }  
}  
  
List<List<CharacterFrequency>> digitToFrequencies = new ArrayList<>();  
for (Map<Character, Integer> counts : digitToCounts) {  
 int totalCharacterCount = counts.values()  
 .stream()  
 .mapToInt(Integer::intValue)  
 .sum();  
 List<CharacterFrequency> frequencies = new ArrayList<>();  
 for (Map.Entry<Character, Integer> entry : counts.entrySet()) {  
 double frequency = entry.getValue() / (double) totalCharacterCount;  
 frequencies.add(new CharacterFrequency(entry.getKey(), frequency));  
 }  
 Collections.sort(frequencies);  
 digitToFrequencies.add(frequencies);  
}  
  
System.out.println("Frequency distribution for each digit is:");  
digitToFrequencies.forEach(System.out::println);

*Подбор ключа*

StringBuilder keyBuilder = new StringBuilder();  
for (List<CharacterFrequency> frequencies : digitToFrequencies) {  
 char mostFrequentChar = frequencies.get(0)  
 .getCharacter();  
 int keyInt = mostFrequentChar - mostCommonCharOverall;  
 keyInt = keyInt >= 0 ? keyInt : keyInt + ALPHABET\_SIZE;  
 char key = (char) (FIRST\_CHAR\_IN\_ALPHABET + keyInt);  
 keyBuilder.append(key);  
}  
  
String key = keyBuilder.toString();  
System.out.println("The guessed key is: " + key);  
System.out.print("Is key correct? (yes/no)");  
Scanner scanner = new Scanner(System.in);  
String response = scanner.nextLine();  
if ("no".equals(response)) {  
 System.out.print("Enter correct guessing key: ");  
 key = scanner.nextLine();  
}

# Глава 6. Расшифровка текста.

**Ключ шифрования:** «LEMON»

**Расшифрованный текст:** «ILIVEINAHOUSENEARTHEMOUNTAINSIHAVETWOBROTHERSANDONESISTERANDIWASBORNLASTMYFATHERTEACHESMATHEMATICSANDMYMOTHERISANURSEATABIGHOSPITALMYBROTHERSAREVERYSMARTANDWORKHARDINSCHOOLMYSISTERISANERVOUSGIRLBUTSHEISVERYKINDMYGRANDMOTHERALSOLIVESWITHUSSHECAMEFROMITALYWHENIWASTWOYEARSOLDSHEHASGROWNOLDBUTSHEISSTILLVERYSTRONGSHECOOKSTHEBESTFOODAMICHIGANPROSECUTORHASFILEDCHARGESAGAINSTTHEPARENTSOFAYEAROLDBOYACCUSEDOFFATALLYSHOOTINGHISCLASSMATESATANOAKLANDCOUNTYHIGHSCHOOLEARLIERTHISWEEKARAREMOVESHESAIDWASJUSTIFIEDBYTHEEGREGIOUSFACTSOFTHECASEJAMESANDJENNIFERCRUMBLEYPARENTSOFSHOOTINGSUSPECTETHANCRUMBLEYFACEFOURCOUNTSEACHOFINVOLUNTARYMANSLAUGHTEROAKLANDCOUNTYPROSECUTORKARENMCDONALDANNOUNCEDFRIDAYAUTHORITIESSAYTHETEENAGERKILLEDFOURSTUDENTSANDWOUNDEDSEVENPEOPLEATOXFORDHIGHSCHOOLONNOVWHILEUSINGASEMIAUTOMATICHANDGUNPURCHASEDDAYSEARLIERBYHISFATHERINTHEDEADLIESTSCHOOLSHOOTINGINMORETHANTHREEYEARSWHILETHESHOOTERWASTHEONEWHOENTEREDTHEHIGHSCHOOLANDPULLEDTHETRIGGERTHEREWEREOTHERINDIVIDUALSWHOCONTRIBUTEDTOTHEEVENTSMCDONALDSAIDMCDONALDDECLINEDTOSAYWHETHERTHECRUMBLEYSWEREINCUSTODYTHEDECISIONCOMESAFTERAUTHORITIESSAIDTHATJAMESCRUMBLEYBOUGHTTHEMMSIGSAUERSPPISTOLONNOVANEMPLOYEEOFACMESHOOTINGGOODSINOXFORDMICHCONFIRMEDETHANWASPRESENTWHENHISFATHERBOUGHTTHEGUNMCDONALDSAIDSHECITEDSOCIALMEDIAPOSTSFROMTHEPARENTSTHATCONFIRMTHEGUNWASFORTHEIRSONMCDONALDREADTOREPORTERSSEVERALPIECESOFEVIDENCEFRIDAYINCLUDINGASOCIALMEDIAPOSTBYJENNIFERCRUMBLEYSHORTLYAFTERTHEGUNWASPURCHASEDTHATSHOWEDHERANDETHANATAGUNRANGETOGETHERTHECAPTIONREADMOMANDSONDAYTESTINGOUTHISNEWCHRISTMASPRESENTONNOVDAYSBEFORETHESHOOTINGATEACHERATOXFORDHIGHSCHOOLNOTICEDETHANUSINGHISCELLPHONETOSEARCHFORINFORMATIONONFIREARMAMMUNITIONWHENJENNIFERCRUMBLEYWASCONTACTEDBYTHESCHOOLVIAVOICEMAILABOUTHERSONSTROUBLINGINTERNETSEARCHSHENEVERRESPONDEDMCDONALDSAIDINSTEADSHEEXCHANGEDATEXTMESSAGESWITHHERSONTHATREADLOLIMNOTMADATYOUYOUHAVETOLEARNNOTTOGETCAUGHTDAVIDCHIPMANAVETERANBUREAUOFALCOHOLTOBACCOFIREARMSANDEXPLOSIVESAGENTONCENOMINATEDTOLEADTHEAGENCYSAIDIFPROSECUTORSFINDEVIDENCETHESHOOTERSPARENTSACTEDUNLAWFULLYTHECRIMINALJUSTICESYSTEMHASADUTYTOACTASGUNOWNERSWEHAVEARESPONSIBILITYTHATWHENWEACQUIREFIREARMSWEDONTPUTOURNEIGHBORSANDINPARTICULARTHEIRCHILDRENATRISKCHIPMANSAIDIFAPARENTPROVIDEDTHEIRCARTOANUNLICENSEDTEENFORANIGHTOFDRUNKENJOYRIDINGNOONEWOULDBESHOCKEDIFAPROSECUTORSOUGHTTOHOLDTHEADULTSACCOUNTABLEOAKLANDCOUNTYSHERIFFMICHAELBOUCHARDHASPREVIOUSLYSAIDCRUMBLEYSPARENTSWERENOTCOOPERATINGWITHINVESTIGATORS»

# Итоги лабораторной работы

Мы научились находить ключ шифра методом Касиски и расшифровывать текст, зашифрованный методом Виженера.

# Список литературы

1. [[Wikipedia.org](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F)] – Шифр Виженера;
2. [[Wikipedia.org](https://ru.wikipedia.org/wiki/%D0%A8%D0%B8%D1%84%D1%80_%D0%A6%D0%B5%D0%B7%D0%B0%D1%80%D1%8F)] – Метод Касиски;
3. [[lingua.com](https://lingua.com/english/reading/wonderful-family/)] – Исходный английский текст;