Научно-исследовательская статья: Реализация книжного шифра с использованием стихотворения

Введение

**Книжный шифр** — вид шифра, в котором каждый элемент открытого текста (каждая буква или слово) заменяется на указатель (например, номер страницы, строки и столбца) аналогичного элемента в дополнительном тексте-ключе.

Для дешифрования необходимо иметь как закрытый текст, так и дополнительный текст-ключ. В качестве дополнительного текста часто использовали распространённые книги, либо книги, которые с большой долей вероятности были и у отправителя, и у адресата.

В данной статье представлена реализация книжного шифра на языке программирования Java, где вместо книги используется стихотворение.

Описание алгоритма

Книжный шифр использует конкретный текст для генерации уникальных координат для каждого символа алфавита. В данной реализации, текстом является стихотворение, разбитое на строфы и строки. Каждая буква в тексте получает уникальные координаты, основанные на её местоположении.

Суть метода книжного шифра — это выбор любого текста из книги, где номера слов начинающихся на определённую букву или координаты (строка, номер в строке) самих букв выступают в качестве шифра исходного сообщения. При этом одной исходной букве может соответствовать несколько символов

Класс *BookCipher*

Этот класс отвечает за загрузку ключа из текстового файла, шифрование и дешифрование сообщений.

Поля класса:

- charToCoords: отображение символов в координаты.

- *coordsToChar*: отображение координат в символы.

Методы класса:

1. Конструктор *BookCipher(String keyFilePath)*

- Инициализирует объект и вызывает метод загрузки ключа.

2. Метод *loadKey(String keyFilePath)*

- Загружает текст из файла и разбивает его на строфы и строки.

- Для каждой буквы генерирует уникальные координаты в формате строфа-линия-позиция.

3. Метод *encrypt(String text)*

- Шифрует входной текст, заменяя каждую букву на её координаты.

4. Метод *decrypt(String encryptedText)*

- Дешифрует входной текст, заменяя координаты обратно на буквы.

Класс Main

Этот класс содержит метод *main*, который служит точкой входа для программы. Он выполняет следующие действия:

1. Создаёт объект класса *BookCipher*, используя файл ключа.

2. Запрашивает у пользователя ввод текста для шифрования.

3. Отображает зашифрованный текст.

4. Дешифрует зашифрованный текст и выводит его на экран.

Пример кода

import java.io.BufferedReader;  
import java.io.FileReader;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Random;  
import java.util.Scanner;  
class BookCipher {  
 private final Map<Character, String> charToCoords = new HashMap<>();  
 private final Map<String, Character> coordsToChar = new HashMap<>();  
 static final long *seed* = (long) (Math.*random*() \* 500);  
 private final Random random = new Random(*seed*);  
  
 public BookCipher(String keyFilePath) throws Exception { loadKey(keyFilePath); }  
  
 private void loadKey(String keyFilePath) throws Exception {  
 try (BufferedReader rd = new BufferedReader(new FileReader(keyFilePath))) {  
 int stanza = 1;  
 for (String stanzaText : rd.lines().toArray(String[]::new)) {  
 int line = 1;  
 for (String lineText : stanzaText.split(**";"**)) {  
 for (int i = 0; i < lineText.length(); i++) {  
 char ch = lineText.charAt(i);  
 if (Character.*isLetter*(ch) || ch == **' '** || ch == **','**) {  
 String coords = String.*format*(**"%d-%d-%d"**, stanza, line, random.nextInt(50));  
 charToCoords.put(Character.*toLowerCase*(ch), coords);  
 coordsToChar.put(coords, Character.*toLowerCase*(ch));  
 }  
 line++;  
 }  
 stanza++;  
 }  
 }  
 } catch (Exception ex)  
 System.*err*.println(**"Не удалось загрузить ключ: "** + ex.getMessage());  
 }  
  
 public String encrypt(String text) {  
 StringBuilder encryptedText = new StringBuilder();  
 for (char ch : text.toCharArray()) {  
 if (charToCoords.containsKey(Character.*toLowerCase*(ch)))  
 encryptedText.append(charToCoords.get(Character.*toLowerCase*(ch))).append(**" "**);  
 else encryptedText.append(ch);  
 }  
 return encryptedText.toString().trim();  
 }  
  
 public String decrypt(String encryptedText) {  
 StringBuilder decryptedText = new StringBuilder();  
 for (String coords : encryptedText.split(**" "**)) {  
 if (coordsToChar.containsKey(coords))  
 decryptedText.append(coordsToChar.get(coords));  
 else  
 decryptedText.append(coords);  
 }  
 return decryptedText.toString();  
 }  
}  
  
public class Main {  
 public static void main(String[] args) throws Exception {  
 Scanner sc = new Scanner(System.*in*);  
 String keyFilePath = **"key.txt"**;  
 BookCipher key = new BookCipher(keyFilePath);  
 System.*out*.print(**"Введите текст для шифрования: "**);  
 String text = **"съешь ещё этих мягких французских булок, да выпей чаю"**;//sc.nextLine();  
 System.*out*.println(text);  
 System.*out*.printf(**"Семечко: %d**\n**"**, BookCipher.*seed*);  
 String encryptedText = key.encrypt(text);  
 System.*out*.println(**"Зашифрованный текст: "** + encryptedText);  
 String decryptedText = key.decrypt(encryptedText);  
 System.*out*.println(**"Расшифрованный текст: "** + decryptedText);  
 }  
}

Анализ работы

Загрузка ключа

Метод *loadKey* читает файл ключа и анализирует его содержимое, разделяя текст на строфы и строки. Каждая буква получает координаты в формате `строфа-линия-позиция`, что позволяет уникально идентифицировать её.

Шифрование и дешифрование

Метод *encrypt* заменяет каждую букву исходного текста на её координаты, генерируя зашифрованное сообщение. Метод *decrypt* выполняет обратное преобразование, восстанавливая исходный текст из зашифрованного сообщения.

Преимущества и недостатки

**Преимущества**:

- Простота реализации: Использование стихотворения или любого другого текста в качестве ключа делает алгоритм простым в реализации.

- Уникальность ключа: Каждый текст генерирует уникальные координаты для символов, что усложняет задачу злоумышленнику.

- Маскировка: Зашифрованный текст выглядит как последовательность координат, что затрудняет его распознавание.

**Недостатки**:

- Необходимость наличия ключа: Для расшифровки необходимо иметь доступ к точному ключу (стихотворению), что может быть неудобно.

- Ограничение по длине ключа: Длина сообщения ограничена длиной текста ключа.

Возможные атаки и защита

Атаки

*1. Атака по известному тексту (Known-plaintext attack)*

- **Описание**: Злоумышленник имеет доступ к части зашифрованного текста и его расшифровке.

- **Механизм атаки**: Используя совпадения между зашифрованными и расшифрованными сегментами текста, злоумышленник может определить соответствие координат и символов.

- **Защита**: Использовать различные ключи для разных сообщений, регулярно менять текст ключа.

2. *Атака по избранному шифротексту (Chosen-ciphertext attack)*

- **Описание**: Злоумышленник может выбрать фрагменты шифротекста для расшифровки.

- **Механизм атаки**: Злоумышленник выбирает определённые координаты и анализирует их расшифровку для выявления закономерностей.

- **Защита**: Обеспечить ограниченный доступ к системе дешифрования, использовать дополнительные уровни защиты, такие как цифровые подписи и проверку целостности данных.

Заключение

Представленная реализация книжного шифра демонстрирует, как можно использовать стихотворение для создания ключа шифрования. Использование координат для каждой буквы обеспечивает уникальность и сложность расшифровки без наличия ключа. Данный подход может быть адаптирован для различных текстов и приложений, что делает его универсальным инструментом в области криптографии.