Информационная безопасность

Л.7. Элементы криптографии. Однократное гаммирование

Греков Максим Сергеевич

2021

Содержание

# Цель работы

Освоить на практике применение режима однократного гаммирования

# Задание

Нужно подобрать ключ, чтобы получить сообщение «С Новым Годом, друзья!». Требуется разработать приложение, позволяющее шифровать и дешифровать данные в режиме однократного гаммирования. Приложение должно:

1. Определить вид шифротекста при известном ключе и известном открытом тексте.
2. Определить ключ, с помощью которого шифротекст может быть преобразован в некоторый фрагмент текста, представляющий собой один из возможных вариантов прочтения открытого текста.

# Теория

## Гаммирование

Гаммирование представляет собой наложение (снятие) на открытые (зашифрованные) данные последовательности элементов других данных, полученной с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Иными словами, наложение гаммы — это сложение её элементов с элементами открытого (закрытого) текста по некоторому фиксированному модулю, значение которого представляет собой известную часть алгоритма шифрования.

## Как применять

В соответствии с теорией криптоанализа, если в методе шифрования используется однократная вероятностная гамма (однократное гаммирование) той же длины, что и подлежащий сокрытию текст, то текст нельзя раскрыть. Даже при раскрытии части последовательности гаммы нельзя получить информацию о всём скрываемом тексте. Наложение гаммы по сути представляет собой выполнение операции сложения по модулю 2 между элементами гаммы и элементами подлежащего сокрытию текста.

## Нахождение шифротекста

Если известны ключ и открытый текст, то задача нахождения шифротекста заключается в применении к каждому символу открытого текста следующего правила:

где — i-й символ получившегося зашифрованного послания, — i-й символ открытого текста, — i-й символ ключа. Размерности открытого текста и ключа должны совпадать, и полученный шифротекст будет такой же длины.

## Нахождение ключа

Если известны шифротекст и открытый текст, то, чтобы найти ключ, обе части равенства необходимо сложить по модулю 2 с :

## Стойкость шифра

К. Шеннон доказал абсолютную стойкость шифра в случае, когда однократно используемый ключ, длиной, равной длине исходного сообщения, является фрагментом истинно случайной двоичной последовательности с равномерным законом распределения. Криптоалгоритм не даёт никакой информации об открытом тексте: при известном зашифрованном сообщении C все различные ключевые последовательности K возможны и равновероятны, а значит, возможны и любые сообщения P.

## Условия стойкости шифра

Необходимые и достаточные условия абсолютной стойкости шифра:

* полная случайность ключа;
* равенство длин ключа и открытого текста;
* однократное использование ключа.

# Ход работы

## Класс Gumming

Для разработки приложения был описан класс *Gumming* (рис. 1), описывающий интересующие нас поля (открытый текст, закрытый текст, ключ шифрования), а также методы для работы с ними:

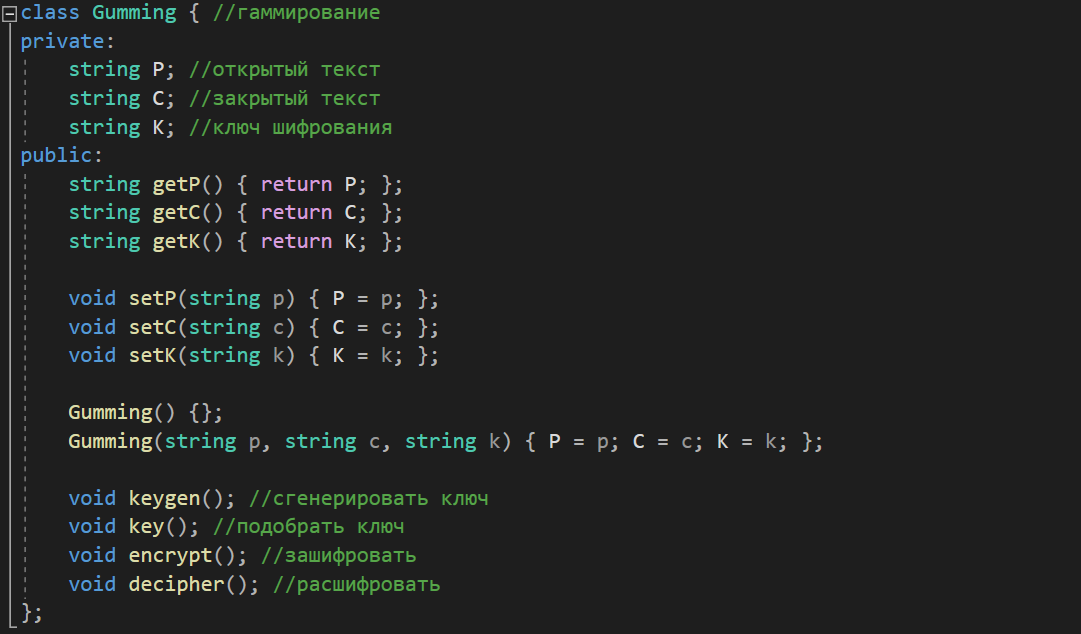


Figure 1: Класс Gumming

## Метод keygen()

Метод *keygen()* (рис. 2) позволяет сгенерировать псевдослучайный ключ такой же длины, как и открытый текст:

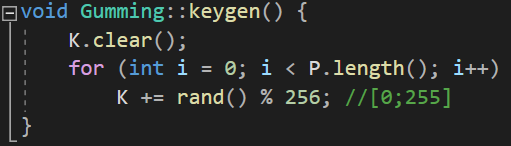


Figure 2: Метод keygen()

## Метод key()

Метод *key()* (рис. 3) позволяет получать ключ, зная открытый и закрытый текст:

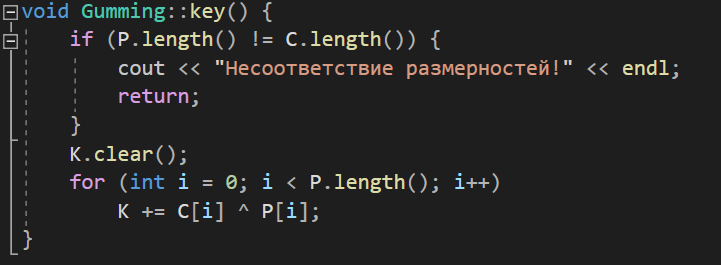


Figure 3: Метод key()

## Метод encrypt()

Метод *encrypt()* (рис. 4) позволяет зашифровывать текст (получать закрытый текст), зная открытый текст и ключ:

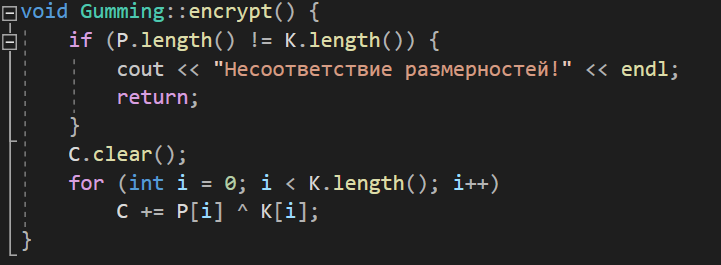


Figure 4: Метод encrypt()

## Метод decipher()

Метод *decipher()* (рис. 5) позволяет расшифровывать текст (получать открытый текст), зная закрытый текст и ключ:

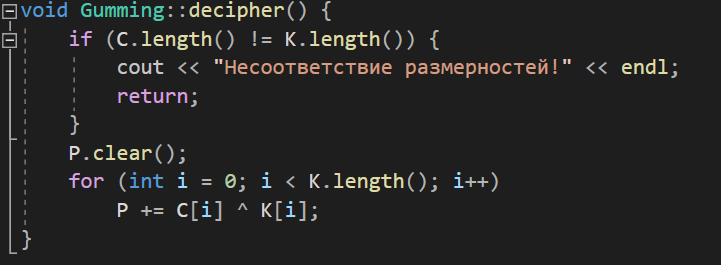


Figure 5: Метод decipher()

## Вывод информации

Также были реализованы методы (рис. 6), служащие для вывода информации на экран в различных представлениях (текстовое, десятичное, шестнадцатеричное):

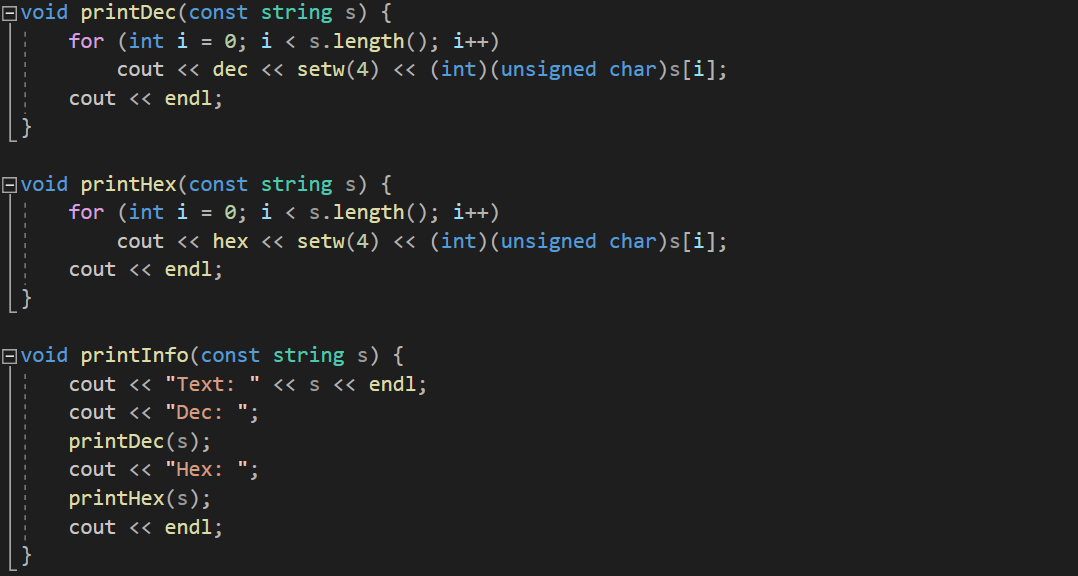


Figure 6: Вывод информации

## Главная программа

В главной программе подключили кириллические символы в консоль (рис. 7), сгенерировали псевдослучайную последовательность, создали объект класса, установили открытый текст *“С Новым Годом, друзья!”*:

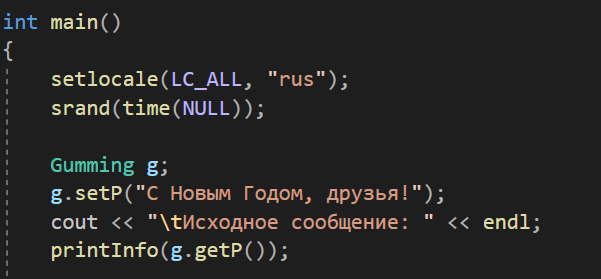


Figure 7: Главная программа (1)

Затем вызвали методы для генерации псевдослучайного ключа и зашифровки текста (рис. 8). Всю информацию поэтапно выводим на экран:

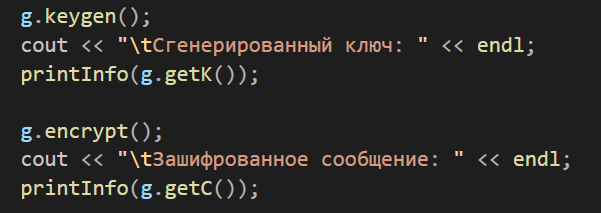


Figure 8: Главная программа (2)

После этого проверили работу методов для расшифровки и нахождения ключа (рис. 9), убедились в корректности работы программы по информации, получанной в консоли (рис. 10):

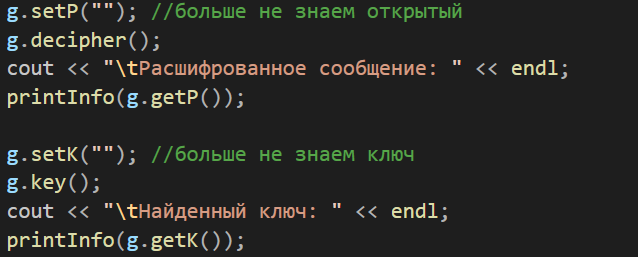


Figure 9: Главная программа (3)

## Вывод программы

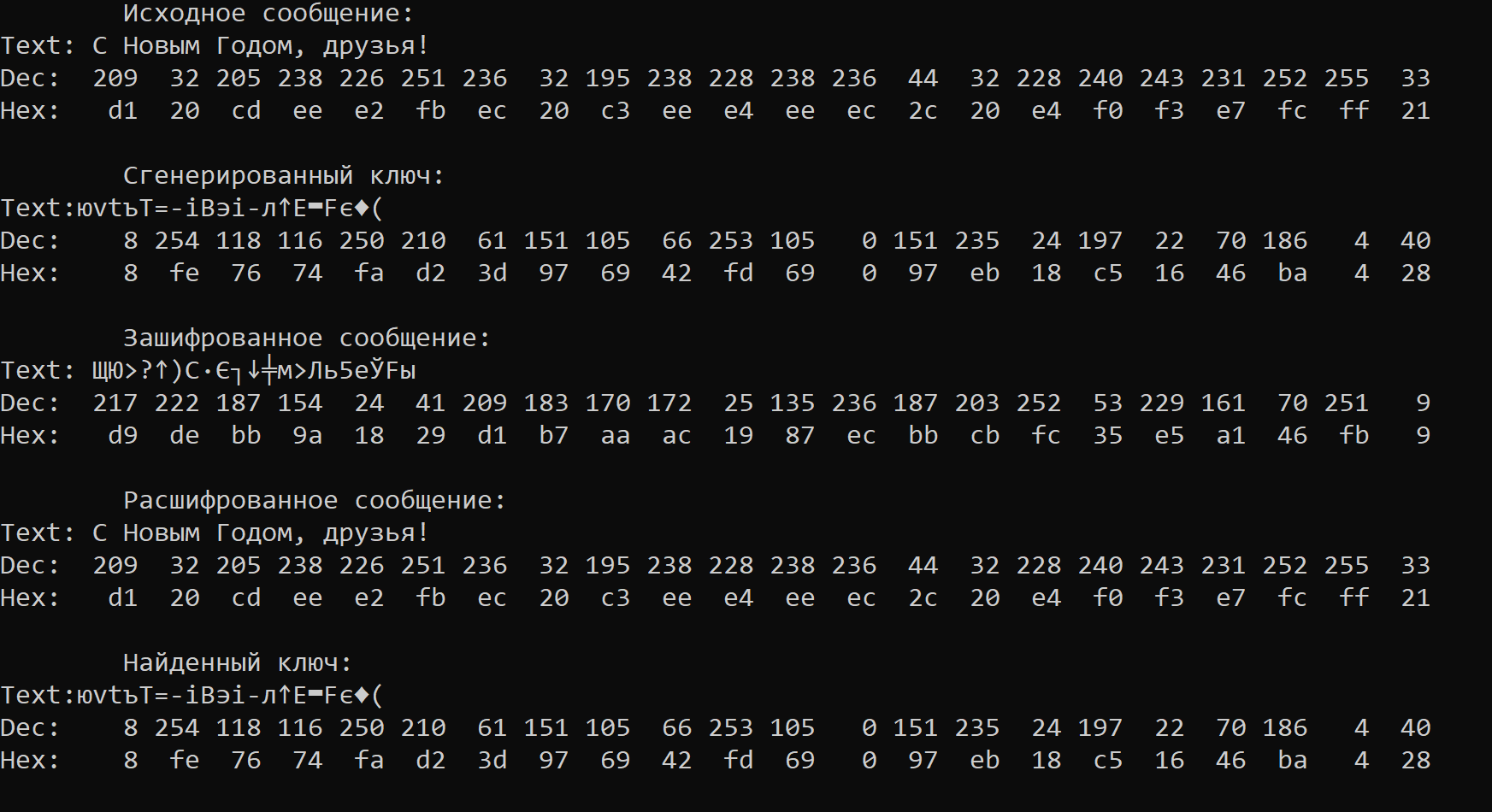


Figure 10: Вывод программы

# Вывод

Освоили на практике применение режима однократного гаммирования