Лабораторная работа №8

Дисциплина: Основы информационной безопасности

Феоктистов Владислав Сергеевич

Содержание

# 1 Цель работы

Целью данной работы является освоение на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом.

# 2 Задание

Два текста кодируются одним ключом (однократное гаммирование). Требуется не зная ключа и не стремясь его определить, прочитать оба текста. Необходимо разработать приложение, позволяющее шифровать и дешифровать тексты и в режиме однократного гаммирования. Приложение должно определить вид шифротекстов и обоих текстов и при известном ключе ; Необходимо определить и выразить аналитически способ, при котором злоумышленник может прочитать оба текста, не зная ключа и не стремясь его определить.

# 3 Теоретическое введение

## 3.1 Однакратное гаммирование

Предложенная Г. С. Вернамом так называемая «схема однократного использования (гаммирования)» (рис. 1) является простой, но надёжной схемой шифрования данных.

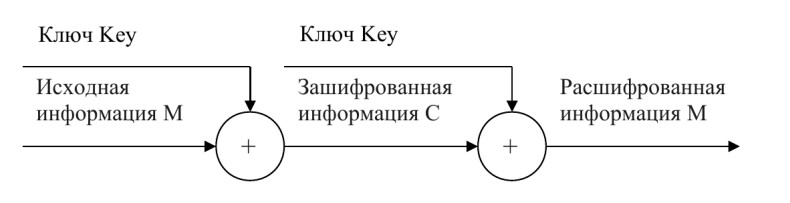


Рис. 1: Схема однократного использования Вернама

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.

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 (XOR) (обозначаемая знаком ) между элементами гаммы и элементами подлежащего сокрытию текста. Напомним, как рабо- тает операция XOR над битами: .

Такой метод шифрования является симметричным, так как двойное прибавление одной и той же величины по модулю 2 восстанавливает исходное значение, а шифрование и расшифрование выполняется одной и той же про- граммой.

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.

Если известны шифротекст и открытый текст, то задача нахождения ключа решается также в соответствии с , а именно, обе части равенства необходимо сложить по модулю 2 с :

Открытый текст имеет символьный вид, а ключ — шестнадцатеричное представление. Ключ также можно представить в символьном виде, воспользовавшись таблицей ASCII-кодов.

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 все различные ключевые последовательности K возможны и равновероятны, а значит, возможны и любые сообщения .

Необходимые и достаточные условия абсолютной стойкости шифра:

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

## 3.2 Таблицы

Таблица 1: Описание некоторых каталогов файловой системы GNU Linux

| Имя каталога | Описание каталога |
| --- | --- |
| / | Корневая директория, содержащая всю файловую |
| /bin | Основные системные утилиты, необходимые как в однопользовательском режиме, так и при обычной работе всем пользователям |
| /etc | Общесистемные конфигурационные файлы и файлы конфигурации установленных программ |
| /home | Содержит домашние директории пользователей, которые, в свою очередь, содержат персональные настройки и данные пользователя |
| /media | Точки монтирования для сменных носителей |
| /root | Домашняя директория пользователя root |
| /tmp | Временные файлы |
| /usr | Вторичная иерархия для данных пользователя |

Таблица 2: Описание некоторых используемых в работе команд

| Команда | Описание команды |
| --- | --- |
| touch | Создает текстовый файл по указанному пути и с указанным именем внутри пути. |
| nano | Запуск в терминале текстовый редактор |
| python3 | Компилятор языка программирования python. В работе используется для запуска .py программ. |

Более подробно об Unix см. в [1–6].

# 4 Выполнение лабораторной работы

## 4.1 Написание вспомогательных функций

Для начала импортируем необходимые для работы библиотеки и функции (рис. 2):

Рис. 2: Импорт функции randint

Рис. 2: Импорт функции randint

Напишем функцию добавления пробелов в конце строки до нужного размера строки (рис. 4). Это необходимо, посокльку длины открытых сообщений должны совпадать, иначе их нельзя будет зашифровать одним и тем же ключом гаммирования.

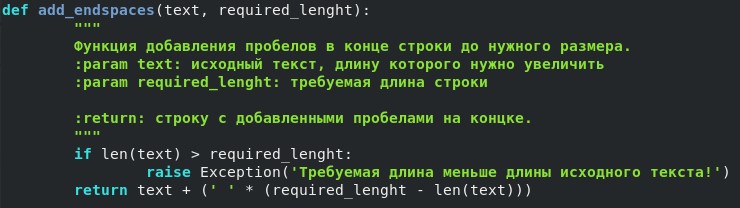


Рис. 3: Функция добавления пробелов

Также напишем функцию, представляющую исходную строку в виде строки последовательности символов в шестнадцатиричном виде, разделенных двоеточиями (рис. 3). Это необходимо, поскольку ключ имеет шестнадцатиричное представление. Также это необходимо будет еще и потому, что в процессе поэлементного применения XOR мы получаем новый код символа, который зачастую невозможно отобразить в нормальном виде.

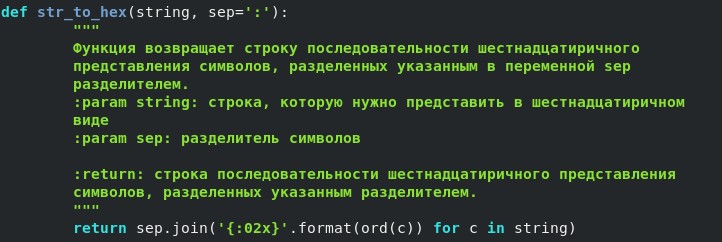


Рис. 4: Функция представления строки в шестнадцатиричном виде

Для удобства дальнейшей работы, напишем функцию генерации случайной строки заданной длины *lenght* с использованием заданных символов *letters*, причем по умолчанию - это латинские символы верхнего и нижнего регистров (рис. 5).

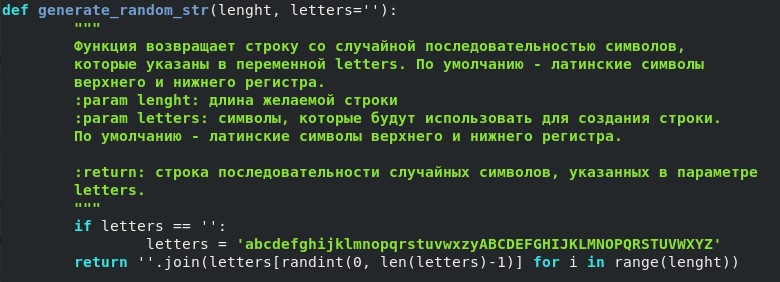


Рис. 5: Функция генерации случайной строки

Напишем функцию, позволяющую шифровать и дешифровать текстовые сообщения в режиме однократного гаммирования (рис. 6).

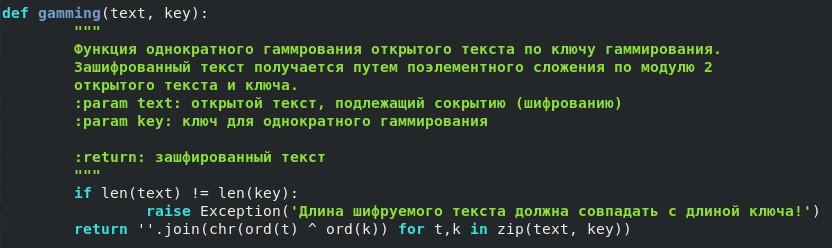


Рис. 6: Функция однократного гаммирования

## 4.2 Эмуляция сценария расшифроваки злоумышленником двух сообщений

Пусть у центра есть две телеграммы и , а также ключ . Тогда с помощью этого ключа можно зашифровать эти две телеграммы методом однократного гаммирования:

Открытый текст можно найти в соответствии с формулами выше, зная шифротекст двух телеграмм, зашифрованных одним ключом. Для это оба равенства складываются по модулю 2. Тогда с учётом свойства операции XOR

получаем:

Предположим, что одна из телеграмм является шаблоном — т.е. имеет текст фиксированный формат, в который вписываются значения полей. Допустим, что злоумышленнику этот формат известен. Тогда он получает достаточно много пар (известен вид обеих шифровок). Тогда зная и учитывая , имеем:

Таким образом, злоумышленник получает возможность определить те символы сообщения , которые находятся на позициях известного шаблона сообщения . В соответствии с логикой сообщения P2, злоумышленник имеет реальный шанс узнать ещё некоторое количество символов сообщения . Затем вновь используется та же формула с подстановкой вместо полученных на предыдущем шаге новых символов сообщения . И так далее. Действуя подобным образом, злоумышленник даже если не прочитает оба сообщения, то значительно уменьшит пространство их поиска.

Сымитируем похожую ситуацию, используя реализованные выше функции (рис. 7).

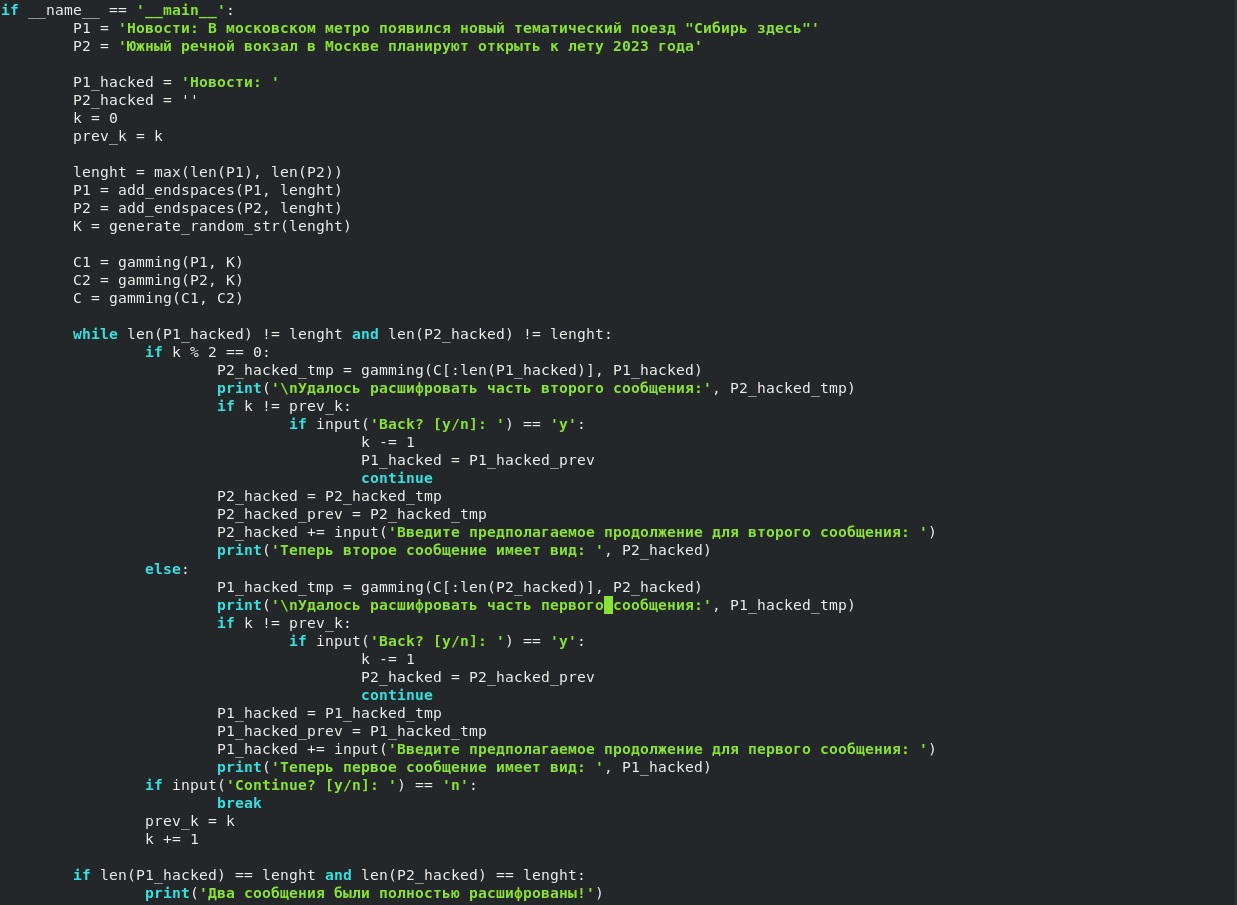


Рис. 7: Код эмуляции ситуации взлома

Теперь можно посмотреть, как работает программа [**cmd:** python3 Lab8.py] (рис. 8 и 9):

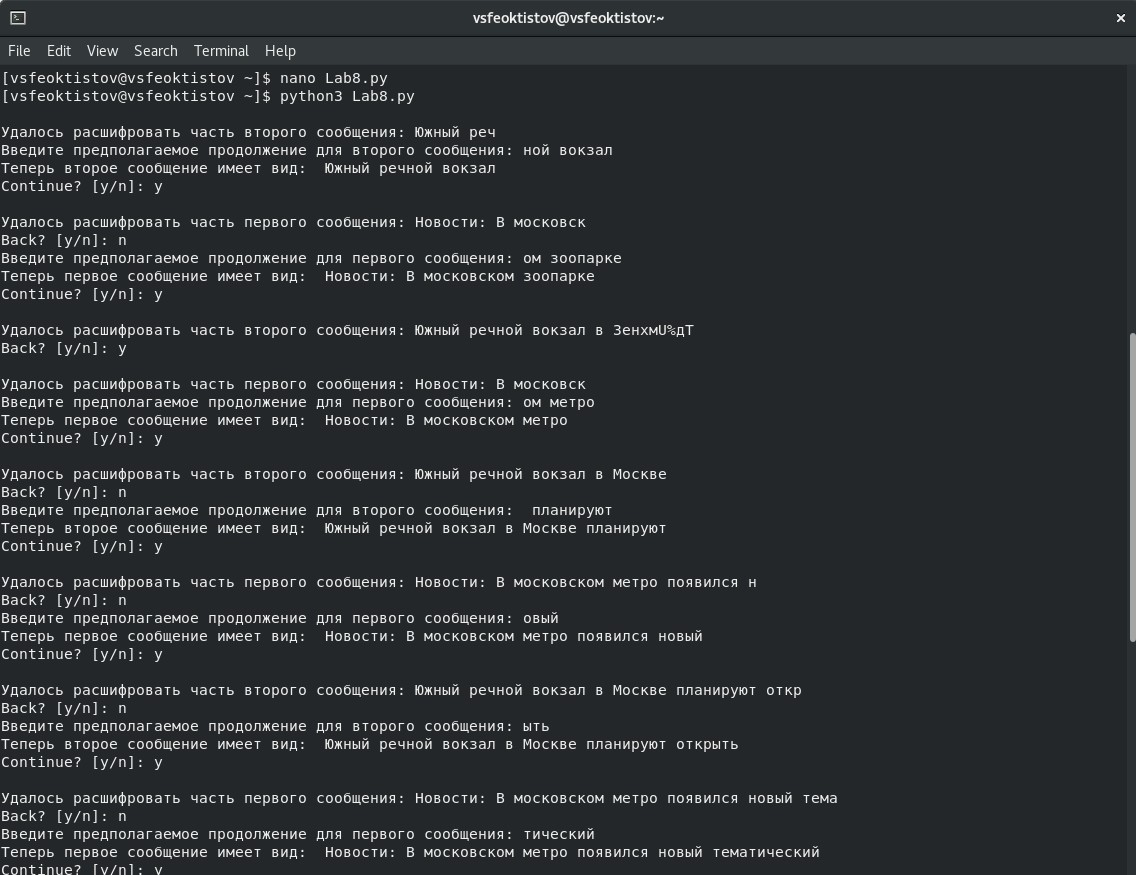


Рис. 8: Работа программы расшифровки

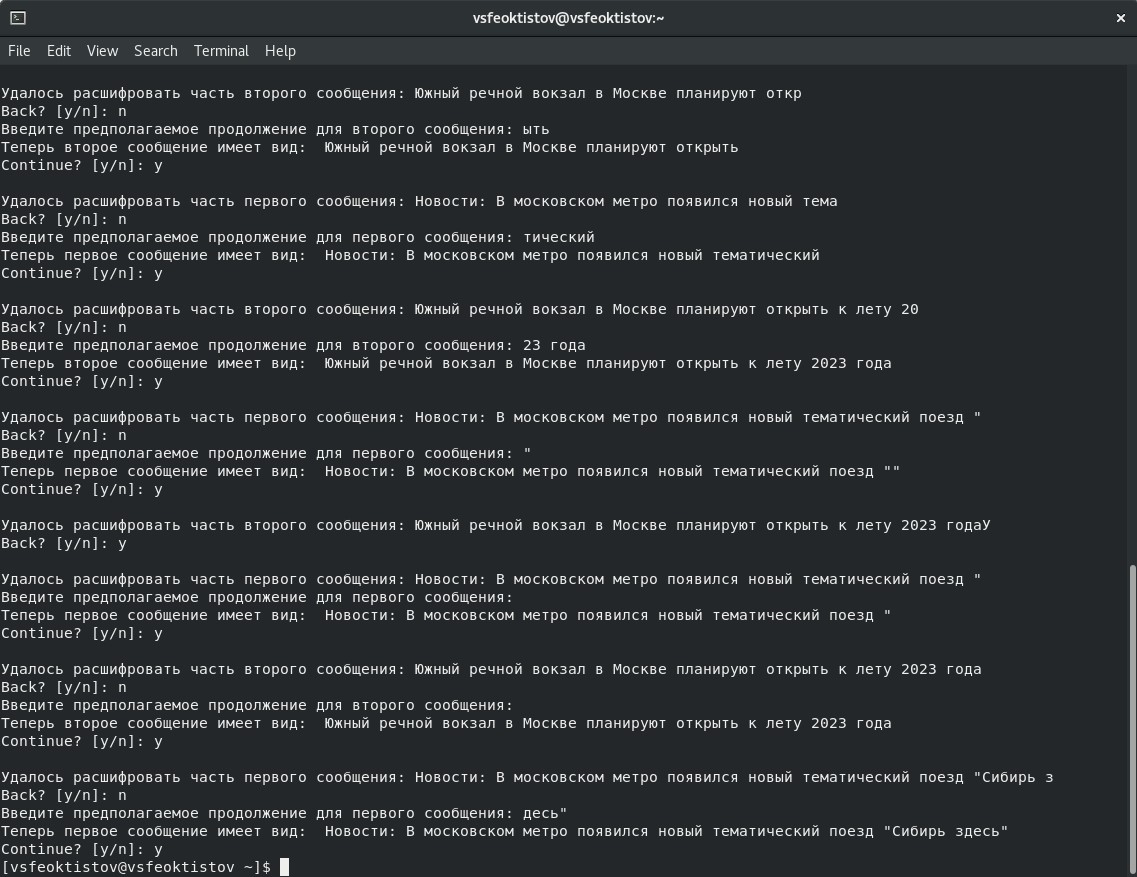


Рис. 9: Работа программы расшифровки

Как можно видеть по рисункам выше, зная лишь только часть одного из сообщений, можно постепенно расшифровать оба сообщения. Таким образом, приходим к выводу, что использование одного и того же ключа для шифрования нескольких сообщений методов однократного гаммирования - это плохая идея.

# 5 Выводы

В процессе выполнения лабораторной работы освоил на практике применение режима однократного гаммирования на примере кодирования различных исходных текстов одним ключом; написал программу, позволяющую расшифровать два сообщения, зная только один из них.

# 6 Контрольные вопросы

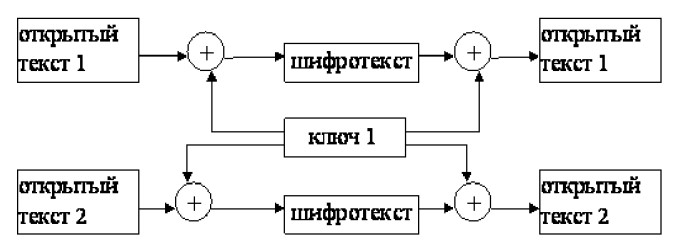
1. **Как, зная один из текстов ( или ), определить другой, не зная при этом ключа?**

* Допустим, что злоумышленнику известен полностью текст . Тогда учитывая , имеем:
* Таким образом, злоумышленник может полностью получить второй текст (). При этом для этого даже не нужно находить ключ гаммирования. Таким же способом можно найти весь текст , если известен полностью текст . Если же известна только часть только одного текста, то можно угадывать символы постепенно, расшифровывая один за счет другого и добавляя веротяную последовательность символов в соотвествии со спецификой информации и языка.

1. **Что будет при повторном использовании ключа при шифровании текста?**

* При повторном использовании ключа при шифровании текста, в соотвествии с формулами:
* получаем исходный текст (текст до шифровки).

1. **Как реализуется режим шифрования однократного гаммирования одним ключом двух открытых текстов?**

* Режим шифрования однократного гаммирования одним ключом двух видов открытого текста реализуется в соответсвии с формулами и со схемой ниже:
* 
* Рис. 10: Общая схема шифрования двух различных текстов одним ключом
* Пусть у центра есть две телеграммы и , а также ключ . Тогда с помощью этого ключа можно зашифровать эти две телеграммы методом однократного гаммирования:

1. **Перечислите недостатки шифрования одним ключом двух открытых текстов.**

* Недостатки шифрования одним ключом двух открытых текстов:
  + зная полностью одно сообщение, можно получить и второе;
  + зная лишь часть одного сообщения, возможно постепенно расшифровать оба сообщения;
  + необходимо, чтобы оба сообщения были одинаковых размеров (либо создать одие ключ длиной равной максимум длин сообщений и для коротких обрезать его, тем самым можно немного увеличить безопасность).

1. **Перечислите преимущества шифрования одним ключом двух открытых текстов.**

* Преимущества шифрования одним ключом двух открытых текстов:
  + из преимуществ только немного сэкономленная память и сокращенное число итераций. Однако это польностью невелируется отсутствием достаточной безопасности.

# Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: <https://www.gnu.org/software/bash/manual/>.

2. Newham C. [Learning the bash Shell: Unix Shell Programming](http://www.amazon.com/Learning-bash-Shell-Programming-Nutshell/dp/0596009658). O’Reilly Media, 2005. 354 с.

3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.

4. Robbins A. [Bash Pocket Reference](https://www.ncbi.nlm.nih.gov/pubmed/25246403). O’Reilly Media, 2016. 156 с.

5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.

6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.