Правительство Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»
(НИУ ВШЭ)

Московский институт электроники и математики им. А.Н. Тихонова

ОТЧЕТ
О ПРАКТИЧЕСКОЙ РАБОТЕ № 2
по дисциплине «Криптографические методы защиты информации»
ТЕМА РАБОТЫ

# *Современные симметричные шифры*

Студент гр. МКБ231
Е.В. Шараев
«24» марта 2024 г.

Руководитель
Заведующий кафедрой информационной безопасности киберфизических систем
канд. техн. наук, доцент
_____О.О. Евсютин
«___» _____ 2024 г.

Москва 2024

**СОДЕРЖАНИЕ**

**1 Задание на практическую работу**

Целью данной работы является приобретение навыков программной реализации современных алгоритмов симметричного шифрования.

В рамках практической работы необходимо выполнить следующее:

1 написать программную реализацию одного из следующих симметричных шифров (по выбору студента):

- Магма;
- **Кузнечик; (выбранный вариант)**
- AES;

2 подготовить отчет о выполнении работы.

Программа должна обладать следующей функциональностью:

- принимать на вход файл, содержащий открытый текст, подлежащий зашифрованию, или шифртекст, подлежащий расшифрованию;
- принимать на вход секретный ключ;
- [***дополнительная опция, не являющаяся обязательной***] давать пользователю возможность выбирать режим работы блочного шифра;
- осуществлять зашифрование или расшифрование выбранного файла по выбору пользователя и сохранять результат в новом файле.

Отчет должен содержать следующие составные части:

- ✓ раздел с заданием;
- ✓ раздел с краткой теоретической частью;
- ✓ раздел с описанием программной реализации с учетом особенностей выбранной среды разработки и языка программирования;
- ✓ раздел с результатами работы программы;
- ✓ раздел с выводами о проделанной работе.

## 2 Краткая теоретическая часть

«Кузнечик» (англ. Kuznyechik[1] или англ. Kuznechik[2][3]) — симметричный алгоритм блочного шифрования с размером блока 128 бит и длиной ключа 256 бит, использующий для генерации раундовых ключей SP-сеть.

**Описание алгоритма**. Для шифрования, расшифрования и генерации ключа используются следующие функции:

1. XOR с раундовым ключом
2. Нелинейное биективное преобразование **S** (подстановка по таблице соотвестствия)
3. Линейное преобразование **L**, где происходит сдвиг элементов блока на 1 блок, а утраченный после сдвига блок восполняется путем свертки всех блоков в один в результате линейного преобразования

При зашифровке операции XSL производятся 9 раз (раундов), а 10-й раунд включает только операцию наложения раундового ключа. Расшифрование представляет собой последовательное применение обратных процедур. Подборнее алогитм описывается в Части 3

**Режим работы.**

Режимы работы гаммирования (реализованный мною помимо простой подстановки блоков) подразумевает не прямое шифрование блоков открытого текста , а сгенерированного из синхропосылки счетчика (получение *гаммы*), затем на каждом блоке открытого текста применяется операция XOR с поученной *гаммой*, результат чего и попадает в ширф-текст.

Режим

## 3. Описание программной реализации

Все файлы данной практической работы я опубликовал в своем гитхаб репозитории https://github.com/Djoongaar/cricket

Программная реализация целиком находится в файле https://github.com/Djoongaar/cricket/blob/master/cricket.py

Максимально подробные комментарии к коду я постарался оставить в самих блоках кода, а здесь лишь описание основных методов класса.

Демонстрационный юпитер ноутбук (пдф файл этого ноутбука приложен к данному отчету) https://github.com/Djoongaar/cricket/blob/master/demo.ipynb

### 3.1 Описание блочного шифра Кузнечик

Программа реализована в классе *Cricket* в файле *cricket.py* в данном репозитории.

* Статические параметры *pi* и pi_*inv*: Целочисленные массивы, где индекс каждого элемента соответствует значению исходного при биективном нелинейном отображении (X) и обратной опреации ($X^{-1}$).

* Метод *__init__(self)* Запускается при инициализации класса. Принимает на вход главный ключ (256 bit), запускает генерацию раундовых ключей и сохраняет их в виде упорядоченного массива в параметре объекта *self.round_keys*

* Статический метод *__generate_round_keys(key)* Принимает на вход главный 256 битный ключ и генерирует из него 10 раундовых 128 битных ключей.

а) На первом метод разбивает глвный ключ на две равные части, и сохраняет их в массив *self.round_keys* в качестве первых двух ключей.

б) Остальные 8 (4 пары) ключей вырабатываются в цикле for i in range(4):, где на каждом шаге цикла к предыдущей паре ключей 8-кратно применяются преобразования сети Фейстеля.

* Метод *encrypt(self, x)*: Принимая на вход блок длинной 128 бит, затем в цикле for rnd in range(9): выполняет 9 раундов зашифрования.

а) Запускает сначала операцию **X** (XOR) с райндовым ключом *x ^ self.round_keys[rnd]*

б) Затем нелинейное биективное преобразование **S** *Cricket.__s_transformation(x ^ self.round_keys[rnd])*

в) И наконец линейное преобразование **L**

```
x = Cricket.__l_transformation(Cricket.__s_transformation(x ^ self.round_keys[rnd]))
```

где каждый байт 16 раз помножается на соответствующий ему элемент поля Галуа. В программе этот шаг реализован в методе *__linear_function(x)*

г) Последний, 10 раунд зишифрования является не полным и состоит только из наложения последнего раундового ключа

```
        return x ^ self.round_keys[-1]
```

* Расшифрование *def decrypt(self, x)* устроено противоположным образом. Метод принимает на вход зашифрованный блок длинной 128 бит.

а) Сначала строится развернутый массив ключей

```
        keys = self.round_keys[::-1]
```

б) Затем 9 раз применяются обратные функции $X^{-1}$ $S^{-1}L^{-1}$

```
        x = Cricket.__s_inv_transformation(Cricket.__l_inv_transformation(x ^ keys[rnd]))
```

в) Последний раунд также является не полным и заключается лишь в применении последнего раундового ключа

```
        return x ^ self.round_keys[-1]
```

### 3.2 Описание режимов работы

Режимы реализованы в классе *EncryptionMode*.

#### 3.2.1 Режим простой замены

В режиме *ecb_mode* блоки открытого текста напрямую шифруются шифром Кузнечик.

Размер блока всегда 16 байт (128 бит) *block_size = 16*

Статический метод *__padding_bytes* сначал добавляет 1 (b'\x01'), а затем нулями добивает количество байт до кратного 16-ти

Шифрования блоков происходит последовательно и независимо друг от друга, поэтому в шифртекст переносятся статистические характеристики исходного текста, а значит этот шифр является не надежным

### 3.2.2 Режим простой замены с зацеплением

Реализован в методе *cbc_mode*

Сначала генерируется синхропосылка

```
# Генерируем синхропосылку
init_val = EncryptionMode.__get_initializing_value(init_val_size)
```

Размер синхропосылки в данном режиме всегда равен 16 байтам.

Далее при зашифровании синхропосылка добавляется в массив зашифрованных данных

```
# Добавляем синхропосылку в результат работ
result_bytes.extend(init_val)
```

Добавляем паддинги методом добавления 1-цы, а затем заполнения 0-ми до кратной длины блока

При расшифровании определяем массив данных и синхропосылку просто отделяя их из массива

```
if operator == "decrypt":
        # Или получаем синхропосылку и зашифрованный текст из зашифрованных данных
        init_val, byte_text = byte_text[:init_val_size], byte_text[init_val_size:]
```

Затем объявляем объект шифровального класса

```
# Инициализируем объект класса Cricket
cricket = Cricket(key)
```

И наконец в цикле зашифровываем или расшифровываем каждый блок после чего переопределяем сдвиговый регистр как конкатенацию предыдущего массива за вычетом использованных первых n байтов и шифр-текста

```
# Переопределяем сдвиговый регистр
init_val = init_val[block_size:] + encrypted_block
```

На каждом шаге полученные зашифрованные / расшифрованные данные я записываю в заранее объявленный массив

```
result_bytes.extend(encrypted_block)
```

### 3.2.3 Режим гаммирования

В режиме Гаммирования шифрование происходит не напрямую. Вместо открытого текста алгоритм блочного шифра шифрует счетчик, состоящий из синхропосылки и нулей (на первом этапе), а затем усекается на заданное количество байт. В моей реализации я по умолчанию задаю размер блока 13 байт *block_size: int = 13*, но оно может быть изменено при вызове функции.

Гамма считается как результат шифрования счетчика, а затьем усечается на заданное количество байт

```
gamma = cricket.encrypt(counter) >> right_shift
```

Затем я увеличиваю счетчик. Инкремент счетчика был вынесен мною в отдельный метод.

```
counter = EncryptionMode.__increment_counter(counter)
```

И накладываю полученную гамму на блоки открытого текста

```
encrypted_block = gamma ^ block_int
```

В конце конкатенирую полученные данные к коллекции *result_bytes.extend(encrypted_block)*

Метод *__get_counter* генерирует счетчик (размером 128 бит) из синхнопосылки и нулей

Метод *__increment_counter* очевидно служит для увеличения значения счетчика по мере шифрования.

Расшифрование происходит точно также, за исключением: того что на первом шаге нужно отделить синхропосылку от массива зашифрованных данных, а также удалить нулевые байты в конце до единичного байта (включительно)

### 3.2.4 Режим гаммирования с обратной связью по выходу

Отличие данного режима от обычного гаммирования что каждый последующий блок XOR-ится не со счетчиком а с сдвиговым регистром, который на каждом шаге избавляется от первых n-байт и конкатенируется с gamma.

Я оставил название переменной `counter` как из предыдущего режима, хотя на самом дела счетчиком эта переменная не является. Отделяем первые n байт для дальнейшего зашифрования

```
counter = init_val >> (m_value - 16) * 8
```

Вторую часть в байтовом виде сохранит отдельно для составления сдвигового регистра

```
counter_rest = int.to_bytes(init_val, m_value, byteorder="big", signed=False)[16:]
```

Затем переопределяем переменную `init_val` как сумму конкатенирую массивы байтов `counter_rest` и `gamma`. Прочие шаги — аналогичны методу гаммирования.

### 3.2.5 Режим гаммирования с обратной связью по шифртексту

Реализация аналогична предыдущему методу, но в конце каждого блока сдивговый регистр переопределяется как конкатенация байт из остатка предыдущего регистра + полученного блока шифртекста

```
init_val = int.from_bytes(
            counter_rest +
            encrypted_block,
            byteorder='big',
            signed=False
)
```

Также при расшифровании в в сдвиговый регистр добавляется не расшифрованный блок гаммы а блок полученный на вход

```
if operator == "decrypt":
            # Если расшифровываем, то добавлять в регистр блок шифртекста
            encrypted_block = block
```

## 4. Демонстрация работы программы

Подробная демонстрация работы программы приведена в Приложении А.

## 5. Выводы

Глубоко изучил алгорит симметричного шифрования «Кузнечик» и получил навыки его реализации и реализации его режимов работы.

# Приложение А

## Демонстрация

## Часть 4.1 Демонстрация работы блочного шифра Кузнечик

1. Чтобы запустить код выполните следующие действия:

`evgeny@hp:~/cricket$ python3 cricket.py <command> <mode> <path/to/file> <key>` , где:

`<command>` - *--encrypt* - чтобы зашифровать файл, *--decrypt* - чтобы расшифровать

`<mode>` - режим шифрования / расшифрования (--dummy / --counter)

`<path/to/file>` - путь к файлу. При зашифровании имя зашифрованного файла будет содержать дополнительное расширение `.enc` . При расшифровании - данное расширение, если оно имеется, будет удалено

`<key>` - 256-битный ключ в виде строки

In [1]:
```
# Демонстрация работы блочного шифра Кузнечик
# импорт классов
from cricket import Cricket

# Создам объект класса Cricket и передам ему 256-битный ключ
cricket = Cricket("8899aabbccddeeff0011223344556677fedcba98765432100
# Объект класса готов к работе
cricket
```

Out[1]:
```
<cricket.Cricket at 0x7f94cb90b8d0>
```

In [2]:
```
# Блочный шифр Кузнечик выполняет 10 раундов шифрования и для каждог
# Раундовые ключи генерируются при инициализации объекта и их можно

cricket.round_keys
```

Out[2]:
```
[181572891734806641530322838679085999735,
 338770000845734292516042252062085074415,
 291356820539020174378226036445198912580,
 814428768517603488548074600570961257 00,
 116164101860579397447240808000140210604,
 251263443283993162038968266093410015259,
 108863003319109490105301954974994962609,
 120259538107168560546004230169145309572,
 248923301836046559943424202620957811991,
 152746288297545385236998257316467458115]
```

In [3]:
```
# Случайная строка для демонстрации работы класса
string = "qB!5pZ@7#tC2*dXe".encode()

# Длинная строки 16 байт - 128 бит
print(len(string))

# Метод encrypt принимает данные в виде целочисленных значений
string_int = int.from_bytes(string, byteorder="big")
encrypted = cricket.encrypt(string_int)
encrypted_bytes = int.to_bytes(encrypted, 16, byteorder="big")

# Зашифрованные текст
print(encrypted_bytes)

# Расшифровываем обратно
decrypted = cricket.decrypt(encrypted)
decrypted_bytes = int.to_bytes(decrypted, 16, byteorder="big")
print(decrypted_bytes)

# Проверка на правильность
assert decrypted_bytes == string

print("Assert: [OK]")
```

```
16
b'"\x1f\xf2F3G\x92vS\x89\xe7Ir\xef\xa32'
b'qB!5pZ@7#tC2*dXe'
Assert: [OK]
```

## Часть 4.2 Демонстрация работы режимов шифрования

### 4.2.1 Режим простой замены

In [4]:
```
# Выведем на консоль все файлы текущей директории
# Вы видите 5 тестовых файлов с разными отрвыками знаменитой поэмы
!ls -l
```

```
total 540
-rwxr-xr-x 1 evgeny evgeny  25651 Apr  6 19:21 cricket.py
-rw-r--r-- 1 evgeny evgeny  37686 Apr  6 20:04 demo.ipynb
drwxr-xr-x 2 evgeny evgeny   4096 Apr  6 19:39 __pycache__
-rw-r--r-- 1 evgeny evgeny 410450 Apr  6 19:21 report.pdf
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 19:21 test_part1.txt
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 19:21 test_part2.txt
-rw-r--r-- 1 evgeny evgeny    629 Apr  6 19:21 test_part3.txt
-rw-r--r-- 1 evgeny evgeny    655 Apr  6 19:21 test_part4.txt
-rw-r--r-- 1 evgeny evgeny    683 Apr  6 19:21 test_part5.txt
-rw-r--r-- 1 evgeny evgeny  41484 Apr  6 19:21 отчет.docx
```

In [5]:
```
# Запускаем зашифрование файла test_part1.txt в режиме простой замен
!./cricket.py --encrypt --dummy "test_part1.txt" "8899aabbccddeeff00
```

In [6]:
```
# Видим что появился файл test_part1.txt.cricket - результат работы
!ls -l
```

```
total 544
-rwxr-xr-x 1 evgeny evgeny  25651 Apr  6 19:21 cricket.py
-rw-r--r-- 1 evgeny evgeny  37686 Apr  6 20:04 demo.ipynb
drwxr-xr-x 2 evgeny evgeny   4096 Apr  6 19:39 __pycache__
-rw-r--r-- 1 evgeny evgeny 410450 Apr  6 19:21 report.pdf
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 19:21 test_part1.txt
-rw-r--r-- 1 evgeny evgeny    672 Apr  6 20:04 test_part1.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 19:21 test_part2.txt
-rw-r--r-- 1 evgeny evgeny    629 Apr  6 19:21 test_part3.txt
-rw-r--r-- 1 evgeny evgeny    655 Apr  6 19:21 test_part4.txt
-rw-r--r-- 1 evgeny evgeny    683 Apr  6 19:21 test_part5.txt
-rw-r--r-- 1 evgeny evgeny  41484 Apr  6 19:21 отчет.docx
```

In [7]:
```
# Попробуем прочитать зашифрованный файл
!cat test_part1.txt.cricket
```

```
<U►G►4′►►vHW[L►Ķ►►└►►►►ꟳC►►bx►►/_%≥◆-►►k►►M►►E
►►Br^~≤►►►~o►>►{c≠l3:0►►-r►►G►"p►►K6├{f
h►[►►►3►T►►►►<a►►0ꟳ►►`'►BYㄱ►►├&►►Y►►►►►►πU
►t…i┤►nQ◆>p,E\►/►i►F►►►H►z≥►►►►►_̣►►►►ᵏ3►̵▒►h◆ɬ—►
►d►►!◆├►1►°►ძ9·►►►
J,►W►q►►►<T못]►7lR(►►►┤eo|►Km►►z爪<►p►►┼►·►►►►S►
j9►►►►ぐ}$►►_►►►►>►x►►►►뎣►≠[►7:►(uu►►e┘H;►
►►└►►nemU►▓►►mᵏᵣ►►►h_►►ꟳ►►►►n¯►Q*►►Y►T¯►├►m►►
x►►!►r¯w►►►0)G̃►K►_►F►&►ꟳ·lᴸꟳ▓K?►4►K►·►►►►U
►►►□D~q├►►►P►►►Bʁ►≥►┘►►►►►ᶜ┻ᴴᵧsE]►►…►►|Y►►M7►
►►►►►gS►ə1►►¯ꟳ►►□►□8o►K►24'è>╎►L►►J@►5►—►►)►c
```

In [8]:
```
# Расшифруем файл в режиме простой замены
!./cricket.py --decrypt --dummy "test_part1.txt.cricket" "8899aabbcc
```

In [9]:
```
# Видим что появился файл test_part1.txt.decrypted - результат работ
!ls -l
```

```
total 548
-rwxr-xr-x 1 evgeny evgeny  25651 Apr  6 19:21 cricket.py
-rw-r--r-- 1 evgeny evgeny  37686 Apr  6 20:04 demo.ipynb
drwxr-xr-x 2 evgeny evgeny   4096 Apr  6 19:39 __pycache__
-rw-r--r-- 1 evgeny evgeny 410450 Apr  6 19:21 report.pdf
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 19:21 test_part1.txt
-rw-r--r-- 1 evgeny evgeny    672 Apr  6 20:04 test_part1.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 20:04 test_part1.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 19:21 test_part2.txt
-rw-r--r-- 1 evgeny evgeny    629 Apr  6 19:21 test_part3.txt
-rw-r--r-- 1 evgeny evgeny    655 Apr  6 19:21 test_part4.txt
-rw-r--r-- 1 evgeny evgeny    683 Apr  6 19:21 test_part5.txt
-rw-r--r-- 1 evgeny evgeny  41484 Apr  6 19:21 отчет.docx
```

In [10]:
```
# Попробуем прочитать расшифрованный файл
!cat test_part1.txt.decrypted
```

```
I
«Мой дядя самых честных правил,
Когда не в шутку занемог,
Он уважать себя заставил
И лучше выдумать не мог.
Его пример другим наука;
Но, боже мой, какая скука
С больным сидеть и день и ночь,
Не отходя ни шагу прочь!
Какое низкое коварство
Полуживого забавлять,
Ему подушки поправлять,
Печально подносить лекарство,
Вздыхать и думать про себя:
Когда же черт возьмет тебя!»
```

In [11]:
```
# Проверим правильность расшифрования
with open("test_part1.txt", "rb") as file:
    open_text = bytearray(file.read())

with open("test_part1.txt.decrypted", "rb") as file:
    decrypted_text = bytearray(file.read())

bool(open_text == decrypted_text)
```

Out[11]: True

### 4.2.2 Режим простой замены с зацеплением

In [12]:
```
!./cricket.py --encrypt --cbc_mode "test_part2.txt" "8899aabbccddeef
```

In [13]:
```
# Видим что появился файл test_part2.txt.cricket - результат работы
!ls -l
```

```
total 552
-rwxr-xr-x 1 evgeny evgeny  25651 Apr  6 19:21 cricket.py
-rw-r--r-- 1 evgeny evgeny  37686 Apr  6 20:04 demo.ipynb
drwxr-xr-x 2 evgeny evgeny   4096 Apr  6 19:39 __pycache__
-rw-r--r-- 1 evgeny evgeny 410450 Apr  6 19:21 report.pdf
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 19:21 test_part1.txt
-rw-r--r-- 1 evgeny evgeny    672 Apr  6 20:04 test_part1.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 20:04 test_part1.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 19:21 test_part2.txt
-rw-r--r-- 1 evgeny evgeny    704 Apr  6 20:04 test_part2.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    629 Apr  6 19:21 test_part3.txt
-rw-r--r-- 1 evgeny evgeny    655 Apr  6 19:21 test_part4.txt
-rw-r--r-- 1 evgeny evgeny    683 Apr  6 19:21 test_part5.txt
-rw-r--r-- 1 evgeny evgeny  41484 Apr  6 19:21 отчет.docx
```

In [14]:
```
# Попробуем прочитать зашифрованный файл
!cat test_part2.txt.cricket
```

```
<binary_data_omitted>
```

In [15]:
```
!./cricket.py --decrypt --cbc_mode "test_part2.txt.cricket" "8899aabl
```

In [16]:
```
# Попробуем прочитать расшифрованный файл
!cat test_part2.txt.decrypted
```

```
II
Так думал молодой повеса,
Летя в пыли на почтовых,
Всевышней волею Зевеса
Наследник всех своих родных.
Друзья Людмилы и Руслана!
С героем моего романа
Без предисловий, сей же час
Позвольте познакомить вас:
Онегин, добрый мой приятель,
Родился на брегах Невы,
Где, может быть, родились вы
Или блистали, мой читатель;
Там некогда гулял и я:
Но вреден север для меня 1.
```

In [17]:
```
# Проверим правильность расшифрования
with open("test_part2.txt", "rb") as file:
    open_text = bytearray(file.read())

with open("test_part2.txt.decrypted", "rb") as file:
    decrypted_text = bytearray(file.read())

bool(open_text == decrypted_text)
```

Out[17]: True

### 4.2.3 Режим гаммирования

In [18]:
```
!./cricket.py --encrypt --ctr_mode "test_part3.txt" "8899aabbccddeef
```

In [19]:
```
# Видим что появился файл test_part3.txt.cricket - результат работы
!ls -l
```

```
total 560
-rwxr-xr-x 1 evgeny evgeny  25651 Apr  6 19:21 cricket.py
-rw-r--r-- 1 evgeny evgeny  37686 Apr  6 20:04 demo.ipynb
drwxr-xr-x 2 evgeny evgeny   4096 Apr  6 19:39 __pycache__
-rw-r--r-- 1 evgeny evgeny 410450 Apr  6 19:21 report.pdf
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 19:21 test_part1.txt
-rw-r--r-- 1 evgeny evgeny    672 Apr  6 20:04 test_part1.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 20:04 test_part1.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 19:21 test_part2.txt
-rw-r--r-- 1 evgeny evgeny    704 Apr  6 20:04 test_part2.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 20:04 test_part2.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    629 Apr  6 19:21 test_part3.txt
-rw-r--r-- 1 evgeny evgeny    645 Apr  6 20:04 test_part3.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    655 Apr  6 19:21 test_part4.txt
-rw-r--r-- 1 evgeny evgeny    683 Apr  6 19:21 test_part5.txt
-rw-r--r-- 1 evgeny evgeny  41484 Apr  6 19:21 отчет.docx
```

In [20]:
```
# Попробуем прочитать зашифрованный файл
!cat test_part3.txt.cricket
```

```
x ~|lKo.Z9FR0j     !-yUi
la=HL]5`T|M(@B i
5m2lZpCsAt3xcSF
aebWX+J X o NMC ;!cX1.c(f
W #2 "b m0o+P
K%e] V w л \G t vhv f +{s 3m
&4r4Y c@A I b 7o m pS8h o o
o4yK qk r$U# WU Ple $ >y 9',Vf
2) a e E Z Xg6 : : n @" ? ` | &q n
| : I ^X' 0 [1'(%T^2
d V
```

In [21]:
```
!./cricket.py --decrypt --ctr_mode "test_part3.txt.cricket" "8899aabl
```

In [22]:
```
# Попробуем прочитать зашифрованный файл
!cat test_part3.txt.decrypted
```

```
III
Служив отлично благородно,
Долгами жил его отец,
Давал три бала ежегодно
И промотался наконец.
Судьба Евгения хранила:
Сперва Madame за ним ходила,
Потом Monsieur ее сменил.
Ребенок был резов, но мил.
Monsieur l'Abbé, француз убогой,
Чтоб не измучилось дитя,
Учил его всему шутя,
Не докучал моралью строгой,
Слегка за шалости бранил
И в Летний сад гулять водил.
```

In [23]:
```
# Проверим правильность расшифрования
with open("test_part3.txt", "rb") as file:
    open_text = bytearray(file.read())

with open("test_part3.txt.decrypted", "rb") as file:
    decrypted_text = bytearray(file.read())

bool(open_text == decrypted_text)
```

Out[23]:
```
True
```

### 4.2.4 Режим гаммирования с обратной связью по выходу

In [24]:
```
!./cricket.py --encrypt --ofb_mode "test_part4.txt" "8899aabbccddeef
```

In [25]:
```
# Видим что появился файл test_part4.txt.cricket - результат работы
!ls -l
```

```
total 568
-rwxr-xr-x 1 evgeny evgeny  25651 Apr  6 19:21 cricket.py
-rw-r--r-- 1 evgeny evgeny  37686 Apr  6 20:04 demo.ipynb
drwxr-xr-x 2 evgeny evgeny   4096 Apr  6 19:39 __pycache__
-rw-r--r-- 1 evgeny evgeny 410450 Apr  6 19:21 report.pdf
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 19:21 test_part1.txt
-rw-r--r-- 1 evgeny evgeny    672 Apr  6 20:04 test_part1.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 20:04 test_part1.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 19:21 test_part2.txt
-rw-r--r-- 1 evgeny evgeny    704 Apr  6 20:04 test_part2.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 20:04 test_part2.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    629 Apr  6 19:21 test_part3.txt
-rw-r--r-- 1 evgeny evgeny    645 Apr  6 20:04 test_part3.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    629 Apr  6 20:04 test_part3.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    655 Apr  6 19:21 test_part4.txt
-rw-r--r-- 1 evgeny evgeny    695 Apr  6 20:04 test_part4.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    683 Apr  6 19:21 test_part5.txt
-rw-r--r-- 1 evgeny evgeny  41484 Apr  6 19:21 отчет.docx
```

In [26]:
```
# Попробуем прочитать зашифрованный файл
!cat test_part4.txt.cricket
```

```
?W□□_·□       □oA□@□□¯\b├┴(□□□TS□7□□□□□□□-□□□┘▽
□_$□□□□□□2樑X□`
□{K□□□Pg□ -7K□□]□□□¬┤□□WY□Ĩ¨□Ⓢe괢≥;{□jg□□C□□┗C
□┤i K□□B□□□□□□ᶜᵣ□□>#H □
ᶜᵣ□□ᴴᵀ┤"x       ¨□□±□□}□□□┗ᴸᴸ▒□`Pπ勦□O  □≤□□ᴴᵀ□□□_8D=ᶠᵣF
F□7w□□□¯$-□□-□□□□□4□□□□□□□¯□□□□z□KYuz□□^▒n
□◆W□s!┬o□b┻#□鬱□oU□M□CgiY□}l□□\≠▒□□□□□_□┼□_@L#□◆z:
hG□□:□□□'>cU:ɬ'□□□o□□_&□³□<□□□V≤c8▓┐□□□5□□>49□
□a□□l□?,□Gb□□□┬┗E□&·&┬┻□□□bɬ'□f6d□'*□q□▓P□□□?≠K
Ӈ_□c□ǵ:ᵒE□□□
```

In [27]:
```
!./cricket.py --decrypt --ofb_mode "test_part4.txt.cricket" "8899aabl
```

In [28]:
```
# Попробуем прочитать зашифрованный файл
!cat test_part4.txt.decrypted
```

```
IV
Когда же юности мятежной
Пришла Евгению пора,
Пора надежд и грусти нежной,
Monsieur прогнали со двора.
Вот мой Онегин на свободе;
Острижен по последней моде,
Как dandy 2 лондонский одет —
И наконец увидел свет.
Он по-французски совершенно
Мог изъясняться и писал;
Легко мазурку танцевал
И кланялся непринужденно;
Чего ж вам больше? Свет решил,
Что он умен и очень мил.
```

In [29]:
```
# Проверим правильность расшифрования
with open("test_part4.txt", "rb") as file:
    open_text = bytearray(file.read())

with open("test_part4.txt.decrypted", "rb") as file:
    decrypted_text = bytearray(file.read())

bool(open_text == decrypted_text)
```

Out[29]:
```
True
```

### 4.2.5 Режим гаммирования с обратной связью по шифртексту

In [30]:
```
!./cricket.py --encrypt --cfb_mode "test_part5.txt" "8899aabbccddeef
```

In [31]:
```
# Видим что появился файл test_part5.txt.cricket - результат работы
!ls -l
```

```
total 576
-rwxr-xr-x 1 evgeny evgeny  25651 Apr  6 19:21 cricket.py
-rw-r--r-- 1 evgeny evgeny  37686 Apr  6 20:04 demo.ipynb
drwxr-xr-x 2 evgeny evgeny   4096 Apr  6 19:39 __pycache__
-rw-r--r-- 1 evgeny evgeny 410450 Apr  6 19:21 report.pdf
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 19:21 test_part1.txt
-rw-r--r-- 1 evgeny evgeny    672 Apr  6 20:04 test_part1.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    671 Apr  6 20:04 test_part1.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 19:21 test_part2.txt
-rw-r--r-- 1 evgeny evgeny    704 Apr  6 20:04 test_part2.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    660 Apr  6 20:04 test_part2.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    629 Apr  6 19:21 test_part3.txt
-rw-r--r-- 1 evgeny evgeny    645 Apr  6 20:04 test_part3.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    629 Apr  6 20:04 test_part3.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    655 Apr  6 19:21 test_part4.txt
-rw-r--r-- 1 evgeny evgeny    695 Apr  6 20:04 test_part4.txt.crick
et
-rw-r--r-- 1 evgeny evgeny    655 Apr  6 20:04 test_part4.txt.decry
pted
-rw-r--r-- 1 evgeny evgeny    683 Apr  6 19:21 test_part5.txt
-rw-r--r-- 1 evgeny evgeny    720 Apr  6 20:04 test_part5.txt.crick
et
-rw-r--r-- 1 evgeny evgeny  41484 Apr  6 19:21 отчет.docx
```

In [32]:
```
!./cricket.py --decrypt --cfb_mode "test_part5.txt.cricket" "8899aabl
```

In [33]:
```
# Попробуем прочитать зашифрованный файл
!cat test_part5.txt.decrypted
```

```
V
Мы все учились понемногу
Чему-нибудь и как-нибудь,
Так воспитаньем, слава богу,
У нас немудрено блеснуть.
Онегин был по мненью многих
(Судей решительных и строгих)
Ученый малый, но педант:
Имел он счастливый талант
Без принужденья в разговоре
Коснуться до всего слегка,
С ученым видом знатока
Хранить молчанье в важном споре
И возбуждать улыбку дам
Огнем нежданных эпиграмм.
```

In [34]:
```
# Проверим правильность расшифрования

with open("test_part5.txt", "rb") as file:
    open_text = bytearray(file.read())

with open("test_part5.txt.decrypted", "rb") as file:
    decrypted_text = bytearray(file.read())

bool(open_text == decrypted_text)
```

Out[34]: True

## Спасибо за внимание и да пребудет с вами сила!

*С уважением, Шараев Евгений!*

## ПРИЛОЖЕНИЕ Б.

### Список использованных источников

1. ГОСТ 34-12-2015 Информационная технология. Криптографическая защита информации. Блочные шифры. – М: Стандартинформ, 2015 . – 25 с.

2. ГОСТ 34-13-2015 Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров – М: Стандартинформ, 2015 . – 42 с.

3. Исходный код использованный программы — URL: https://github.com/Djoongaar/cricket/blob/master/cricket.py

4. Е.А. Ищукова, Р.А. Кошуцкий, Л.К. Бабенко. Разработка и реализация высокоскоростного шифрования данных с использованием алгоритма Кузнечик. - Ростов-на-Дону: Южный федеральный Университет, 2015. - 25 с.

5. В. Рудской. Российские криптографические стандарты: функции хэширования, блочные шифры и режимы их работы [Электронный курс]. - URL: https://events.yandex.ru/lib/talks