## Основные алгоритмы синхронного шифрования

### Цель работы

Познакомиться с механизмами реализации самых простых алгоритмов шифрования и дешифрования текста.

### Задания для выполнения

1. Написать функцию шифрования и дешифрования текста обобщенным шифром Цезаря.
2. Написать функцию, принимающую шифротекст, зашифрованный шифром из предыдущего задания и восстанавливающий текст, без знания ключа.
3. Реализовать в виде функций шифр Вернама.

### Методические указания

#### Шифр Цезаря

Классический шифр Цезаря предполагает смещение каждой буквы текста на следующую за ней через три. Последние буквы смещаются в начало алфавита по кольцу. Мы будем использовать модифицированный алгоритм, в котором ключом является целое число - величина смещения.

Работать с алфавитом программно не очень удобно, да и нет никакой необходимости. Мы воспользуемся кодировкой текста - соответствием между символами и целыми числами. В питоне встроены очень полезные функции: ord(c) - возвращает число, соответствующее символу с в кодировке unicode; chr(i) - наоборот, возвращает символ по его целочисленному коду.

Для реализации шифра Цезаря нам нужно написать две функции - encrypt(k, m) и decrypt(k, c). Первая принимает в качестве аргументов число, служащее ключом и строку текста. Она должна превратить строку в массив чисел используя функцию ord(), затем прибавить к каждому числу ключ и преобразовать получившийся массив чисел в строку.

Обратите внимание, что чтобы исключить выход за границы кодировки операцию сложения необходимо производить по модулю 65536 - это максимальное число, поддерживающееся функцией chr().

Функция дешифрования работает аналогично, но вместо сложения, вычитает ключ из каждого символа (также по модулю).

#### Взлом шифра Цезаря

Шифр Цезаря даже в такой обобщенной форме не является стойким и может быть взломан простым частотным анализом. Для этого достаточно подсчитать частоты вхождения различных символов в состав шифротекста. Так как любой символ исходного сообщения имеет однозначное соответствие с символом шифротекста, частоты символов будут совпадать с частотами символов текста.

Текст на любом языке имеет довольно устойчивую частотную характеристику. Различные буквы в языке используются с разной частотой. В любом более-менее длинном тексте самый часто встречаемый символ - это пробел. Зная это мы можем предположить, что символ, наиболее часто встречающийся в шифротексте - это символ, соответствующий пробелу. Зная это, мы можем легко вычислить смещение (то есть ключ) и восстановить текст, даже не зная ключа.

Обратите внимание, что частотный анализ будет прекрасно работать на шифротекстах, стостоящих из примерно десятка слов и более. Чем короче шифротекст, тем больше вероятность, что такой подход даст сбой.

#### Шифр Вижинера

Фатальный недостаток шифра Цезаря, как мы говорили - то, что каждый символ текста преобразуется в один и тот же символ шифротекста. Это можно исправить используя ключ длиной не в одно число, а в несколько. Тогда первый символ смещается на первое число в ключе, второй - на второе и так далее. Дополнительно, такой ключ тоже можно воспринимать как строку - выполнив преобразование из массива чисел в массив символов.

Такой шифр имеет абсолютную криптографическую стойкость, если ключ равен по длине тексту и используется только один раз. На практике это очень неудобно и приходится использовать ключ гораздо короче, чем сам текст. В таком случае, ключ просто “размножается” до нужной длины.

Вам нужно написать также две функции - encrypt(k, m) и decrypt(k, c), только теперь вместо числового ключа они будут принимать строку произвольной длины, а для проведения математических операций ее также нужно преобразовать в массив чисел. Как виртуально “размножить” ключ до нужной длины придумайте самостоятельно.

#### Использование XOR

На практике в шифрах, наподобие Вижинера используется функция XOR вместо сложения. Она обладает большей равномерностью, не требует взятия модуля и обратна самой себе. Попробуйте в предыдущем задании заменить операцию сложения двух чисел на XOR между ними.

#### Цепочка блоков

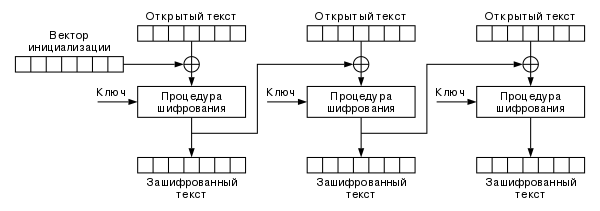
Шифр, описанный в предыдущем пункте уже гораздо сложнее взломать, чем примитивного Цезаря, однако это тоже возможно. Здесь можно использовать тот факт, что ключ повторяется много раз, а значит, частотный анализ все еще можно применить, хотя и не так “в лоб”. Почитайте, например, [здесь](https://habr.com/ru/post/221485/).

Для того, чтобы использовать каждый раз разные строки, с которыми мы складываем исходный текст, можно использовать много приемов, Например, создать псевдослучайный генератор. А можно обратить внимание на различные режимы шифрования. Один из самых простых примеров - использование случайного IV и цепочки блоков.

Давайте представим наше исходное сообщение в виде цепочки блоков такой же длины, что и ключ. В стандартных шифрах длина блока регламентирована, но мы не будем усложнять. Тогда мы можем сказать, что каждый блок в предыдущем задании просто складывается с одним и тем же ключом и получаются соответствующие блоки шифротекста.

Для того, чтобы “исказить” ключ необходимо сгенерировать случайный набор символов, так называемый IV или вектор инициализации. Первый блок текста перед шифрованием складывается с ним. Второй - с первым блоком шифротекста и так далее. В таком случае, мы избежим повторения одного и того же ключа много раз.

Диаграмма показывает этот процесс более наглядно:



Обратите внимание, что теперь для расшифровки текста необходимо знать и ключ и вектор инициализации. Однако, зная только вектор, все равно расшифровать сообщение невозможно, так что для упрощения обмена IV склеивается с шифротекстом и передается вместе с ним в открытом виде.

### Контрольные вопросы

1. В чем заключается алгоритм частотного анализа?
2. Какие есть другие распространенные атаки на криптографические алгоритмы?
3. Какие симметричные шифры используются в настоящее время и считаются надежными?

### Дополнительные задания

1. Реализуйте алгоритм шифрования OTP (one time pad).
2. Реализуйте алгоритм цепочки блоков (Cipher block chaining)
3. (\*) Реализуйте сеть Фейстеля