Математические основы защиты информации и информационной безопасности

Отче по лабораторной работе № 3

Кейела Патачона НПМмд-02-21

Содержание

# Цель работы

Изучить алгоритм шифрования гаммированием конечной гаммой.

# Теоретические сведения

## Гаммированием конечной гаммой

Гаммирование — процедура наложения при помощи некоторой функции F на исходный текст гаммы шифра, т.е. псевдослучайной последовательности (ПСП) с выходов генератора G. Псевдослучайная поСледовательность по своим статистическим свойствам неотличима от случайной послеловательности, но является детерминированной, те. известен алгоритм ее формирования. Чаще Обычно в качестве функции F берется операция поразрядного сложения по модулю два или по модулю N (N число букв алфавита открытого текста).

Простейший генератор псевдослучайной последовательности можно представить рекуррентным соотношением:

уi = aу(i-1) + b mod(m), i = (1;m),

где уi — i-й член последовательности псевдослучайных чисел, а, y0, b — ключевые параметры. Такая последовательность состоит из целых чисел от 0 до m — 1. Если элементы уi и уj совпадут, то совпадут и последующие участки. Таким образом, ПСП является периодической. Знание периода гаммы бущественно облегчает криптоанализ. Максимальная длина периода равна m. Для ее достижения необходимо удовлетворить следующим условиям:

1. b и m — взаимно простые числа;
2. а- 1 делится на любой простой делитель числа m;
3. а- 1 кратно 4, если т кратно 4.

Стойкость шифров, основанных на процедуре гаммирования, зависит от характеристик гаммы — длины и равномерности распределения вероятностей появления знаков гаммы.

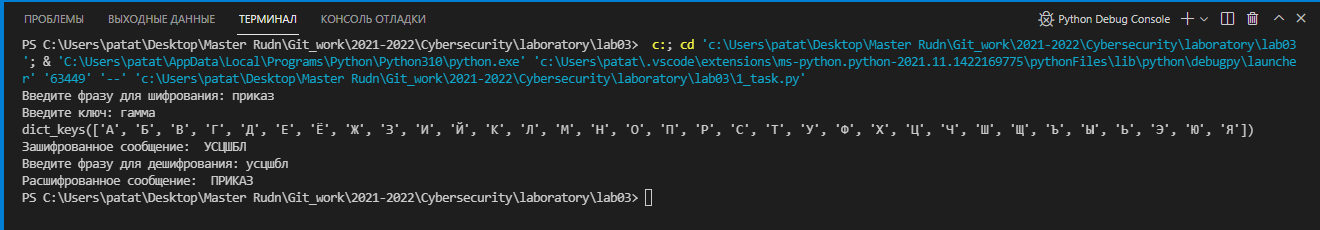
При использовании генератора ПСП получаем бесконечную гамму. Однако, возможен режим шифрования конечной гаммы. В роли конечной гаммы может выступать фраза. Как и ранее, используется алфавитный порядок букв, т. буква «а» имеет порядковый номер 1, «6» —2 итд.

# Выполнение работы

## Реализация шифра на языке Python

# Вводим алфавит и ключ  
word\_to\_encode = input("Введите фразу для шифрования: ").upper()  
key\_word = input("Введите ключ: ").upper()  
# Растягиваем ключ на длину слова   
if len(key\_word) < len(word\_to\_encode):  
 k = (len(word\_to\_encode) % len(key\_word))  
 key\_word = '' + key\_word \* (len(word\_to\_encode) // len(key\_word)) + key\_word[:k]  
# Формируем алфвавит и порядковый словарь  
alphabet = 'АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ'  
alp\_dict = {letter: idx + 1 for idx, letter in enumerate(alphabet)}  
# процесс кодировки  
encoded\_word = ''  
for word\_letter, key\_letter in zip(word\_to\_encode, key\_word):  
 encoded\_word += list(alp\_dict.keys())[(alp\_dict[word\_letter] + alp\_dict[key\_letter] % len(alphabet))-1]  
print("Зашифрованное сообщение: ", encoded\_word)  
# процесс декодировки  
word\_to\_decode = input("Введите фразу для дешифрования: ").upper()  
decoded\_word = ''  
for word\_letter, key\_letter in zip(word\_to\_decode, key\_word):  
 decoded\_word += list(alp\_dict.keys())[(alp\_dict[word\_letter] - alp\_dict[key\_letter] % len(alphabet))-1]  
print("Расшифрованное сообщение: ", decoded\_word)

## Контрольный пример



Работа алгоритма гаммированием конечной гаммой

# Выводы

Изучили алгоритм шифрования гаммированием конечной гаммой.

# Список литературы

1. [ШИФРЫ ГАММИРОВАНИЯ](https://www.sites.google.com/site/anisimovkhv/learning/kripto/lecture/tema6)
2. [Гаммирование](https://ru.wikipedia.org/wiki/Гаммирование)