Отчёт по лабораторной работе № 3

Нирдоши Всеволод Раджендер

Шифрование методом гаммирования

## Цель работы

Изучить принцип работы шифрования методом гаммирования, включая генерацию псевдослучайных последовательностей для гаммы и реализацию операции модульного сложения (mod N) для шифрования и дешифрования данных. Понять механизм защиты информации с использованием конечных и бесконечных гамм.

## Задание

1. Реализовать алгоритм шифрования исходного сообщения с использованием метода гаммирования.
2. Использовать псевдослучайную последовательность в качестве гаммы для шифрования данных.
3. Провести дешифрование сообщения и убедиться, что восстановленный текст совпадает с исходным.
4. Проанализировать стойкость шифра и сделать выводы о зависимости стойкости от характеристик гаммы.

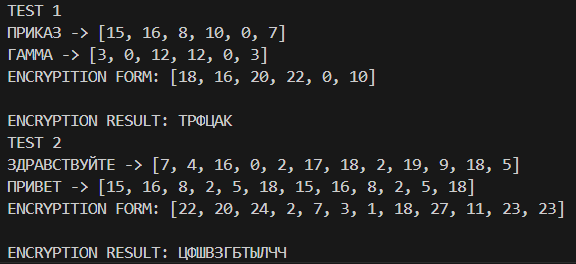
## Выполнение работы

1. Для шифрования была использована гамма, сгенерированная на основе рекуррентного соотношения γ\_i = a⋅γ\_(i+1) + b mod m, где параметры a, b и m заданы согласно условиям задачи.

def encrypt(letters\_pair: tuple):  
 idx = (letters\_pair[0] + letters\_pair[1]) % m  
 return idx

1. Операция модульного сложения была применена к каждому символу сообщения и соответствующему символу гаммы для получения зашифрованного текста. Формула шифрования: c\_i = (p\_i + k\_i) mod N , где p\_i – это i-й символ исходного сообщения, k\_i – i-й символ гаммы, N – количество символов в алфавите.
   * message\_cleared = list(filter(lambda s: s.lower() in alphabet, message)) gamma\_cleared = list(filter(lambda s: s.lower() in alphabet, gamma))
   * message\_ind = list(map(lambda s: alphabet.index(s.lower()), message\_cleared)) gamma\_ind = list(map(lambda s: alphabet.index(s.lower()), gamma\_cleared))
   * for i in range(len(message\_ind) - len (gamma\_ind)): gamma\_ind.append(gamma\_ind[i % len(gamma\_ind)])
   * print(f’{message.upper()} -> {message\_ind}‘) print(f’{gamma.upper()} -> {gamma\_ind}’)
   * encrypted\_ind = list(map(lambda s: encrypt(s), zip(message\_ind, gamma\_ind)))
   * print(f’ENCRYPITION FORM: {encrypted\_ind}’)
   * return ’’.join(list(map(lambda s: alphabet[s],encrypted\_ind))).upper()
2. В ходе работы были проведены эксперименты с различными значениями параметров гаммы для проверки влияния длины и равномерности гаммы на стойкость шифра.

def test\_encryption(message: str, gamma: str):  
 print(f'ENCRYPTION RESULT: {gamma\_encryption(message, gamma)}')  
  
 def main():  
 message = "приказ"  
 gamma = "гамма"  
 print("TEST 1")  
 test\_encryption(message, gamma)  
  
 message = "здравствуйте"  
 gamma = "привет"  
 print("TEST 2")  
 test\_encryption(message, gamma)  
  
 if \_\_name\_\_ == "\_\_main\_\_":  
 main()



## Выводы

В ходе лабораторной работы были изучены принципы шифрования методом гаммирования. Полученные результаты показали, что стойкость шифра напрямую зависит от длины гаммы и её равномерности. При использовании псевдослучайной последовательности генератора гаммы шифр становится устойчивым к криптоанализу, но в случае периодичности гаммы возможно упрощение расшифровки. Таким образом, для достижения максимальной стойкости шифрования необходимо выбирать параметры генерации гаммы так, чтобы она была как можно более случайной и непредсказуемой.