

Лабораторная работа 3

Шифрование гаммированием

Пологов Владислав Александрович

Содержание

1	Цель работы	4
2	Описание реализации	5
3	Реализация	6
3.1	Шифрование гаммированием	6
3.2	Условия достижения максимальной длины периода m	7
3.3	Код реализации шифрования гаммированием	7
3.4	Код реализации шифрования гаммированием	8
3.5	Код дешифратора	8
3.6	Код дешифратора	9
4	Вывод	10

List of Figures

2.1	Функции шифратора и дешифратора	5
3.1	Гаммирование	6
3.2	Код шифрования гаммированием	8
3.3	Код дешифратора	9

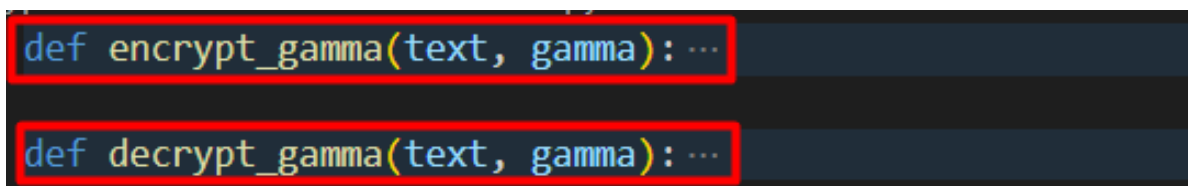
1 Цель работы

Реализовать алгоритм шифрования гаммированием конечной гаммой.

2 Описание реализации

Для реализации алгоритмов использовались средства языка Python.

Были реализованы как шифратор, так и дешифратор реализуемого алгоритма.
(рис. -fig. 2.1)



```
def encrypt_gamma(text, gamma): ...  
  
def decrypt_gamma(text, gamma): ...
```

The image shows a dark-themed code editor with two lines of Python code. The first line is `def encrypt_gamma(text, gamma): ...` and the second line is `def decrypt_gamma(text, gamma): ...`. Both lines are highlighted with a red rectangular border.

Figure 2.1: Функции шифратора и дешифратора

3 Реализация

3.1 Шифрование гаммированием

Гаммирование - процедура наложения при помощи некоторой функции F на исходный текст гаммы шифра, то.е. псевдослучайной последовательности (ПСП) с выходов генератора G . Псевдослучайная последовательность является детерминированной, т.е. известен алгоритм её формирования. (рис. -fig. 3.1)

$$\gamma_i = \alpha * \gamma_{i-1} + b * \text{mod}(m), i = 1, m$$

где γ_i — i -й член последовательности псевдослучайных чисел, α, γ_0, b — ключевые параметры.

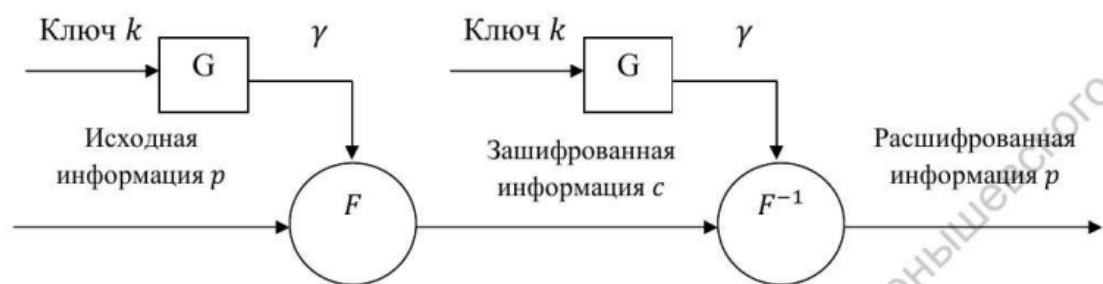


Figure 3.1: Гаммирование

3.2 Условия достижения максимальной длины периода m

ПСП является периодической. Знание периода гаммы существенно облегчает криптоанализ. Максимальная длина периода равна m . Для её достижения необходимо удовлетворить следующим условиям:

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

3.3 Код реализации шифрования гаммированием

Для реализации были использованы функции получения алфавита и продления ключа до длины исходной строки из предыдущих лабораторных. (рис. -fig. 3.2)

3.4 Код реализации шифрования гаммированием

```
def encrypt_gamma(text, gamma):
    alphabet = [chr(c) for c in range(ord('a'), ord('я') + 1)]
    key = (gamma*(len(text)//len(gamma)+1))[:len(text)]
    ind_text = []
    ind_key = []
    res_index = []
    res = ''
    for c in text:
        for l in alphabet:
            if c == l:
                ind_text.append(alphabet.index(l))
    for k in key:
        for z in alphabet:
            if k == z:
                ind_key.append(alphabet.index(z))
    for i in range(len(text)):
        res_index.append((ind_text[i] + ind_key[i]) % 33 + 1)
    for v in res_index:
        res += alphabet[v].upper()
    return res
```

Figure 3.2: Код шифрования гаммированием

3.5 Код дешифратора

Дешифрация отличается лишь формулой получения индекса элемента. (рис. - fig. 3.3)

3.6 Код дешифратора

```
def decrypt_gamma(text, gamma):
    alphabet = [chr(c) for c in range(ord('a'), ord('я') + 1)]
    key = (gamma*(len(text)//len(gamma)+1))[:len(text)]
    ind_text = []
    ind_key = []
    res_index = []
    res = ''
    for c in text:
        for l in alphabet:
            if c == l:
                ind_text.append(alphabet.index(l))
    for k in key:
        for z in alphabet:
            if k == z:
                ind_key.append(alphabet.index(z))
    for i in range(len(text)):
        res_index.append(ind_text[i] - (ind_key[i] % 33) - 1)
    for v in res_index:
        res += alphabet[v].upper()
    return res
```

Figure 3.3: Код дешифратора

4 Вывод

- Реализовали алгоритм шифрования гаммированием конечной гаммой.
- Узнали алгоритм формирования псевдослучайной последовательности.