Шифр гаммирования

Савченко Елизавета, НБИ-01-20

21 октября, 2023. Россия, Москва

Российский Университет Дружбы Народов

Цели и задачи

Цель лабораторной работы

Изучение алгоритма шифрования гаммированием

Выполнение лабораторной работы

Гаммирование

Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных.

Гаммирование

Наложение (или снятие) гаммы на блок сообщения в рассматриваемом нами стандарте реализуется с помощью операции побитного сложения по модулю 2 (XOR). То есть при шифровании сообщений каждый блок открытого сообщения ксорится с блоком криптографической гаммы, длина которого должна соответствовать длине блоков открытого сообщения. При этом, если размер блока исходного текста меньше, чем размер блока гаммы, блок гаммы обрезается до размера блока исходного текста (выполняется процедура усечения гаммы).

Алгоритм

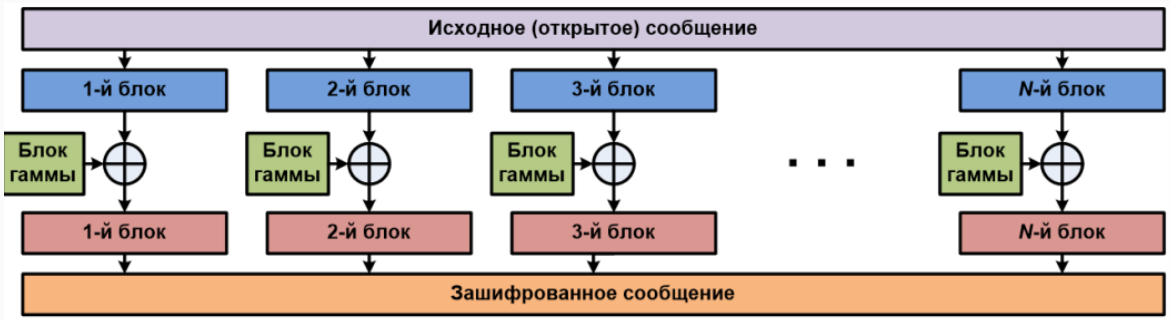
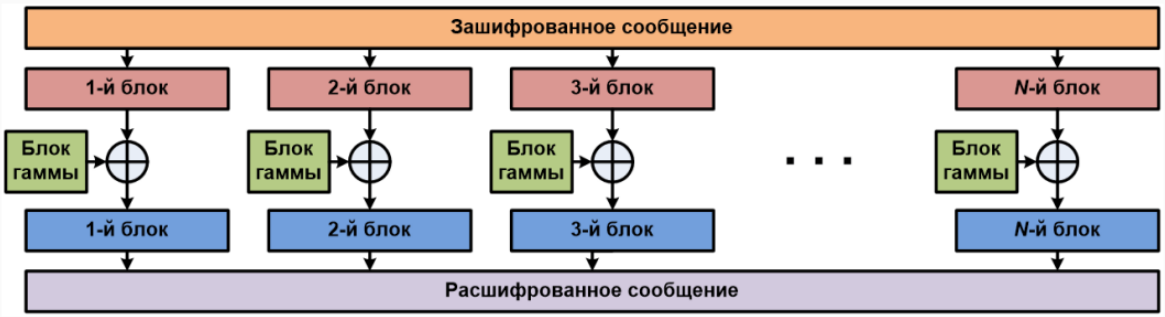


Figure 1: шифрование

Дешифровка


Зашифрованное сообщение
1-й блок
2-й блок
3-й блок
N-й блок
Блок гаммы
Блок гаммы
Блок гаммы
Блок гаммы
1-й блок
2-й блок
3-й блок
N-й блок
Расшифрованное сообщение

!

Работа алгоритма гаммирования

T	К	А	Ф	Е	Д	Р	А		С	И	С	Т	Е	М		И	Н	Ф	О	Р	М	А	Т	И	К	И
G	С	И	М	В	О	Л	С	И	М	В	О	Л	С	И	М	В	О	Л	С	И	М	В	О	Л	С	И
T	12	1	22	6	5	18	1	34	19	10	19	20	6	14	34	10	15	22	16	18	14	1	20	10	12	10
G	19	10	14	3	16	13	19	10	14	3	16	13	19	10	14	3	16	13	19	10	14	3	16	13	19	10
T+G	31	11	36	9	21	31	20	44	33	13	35	33	25	24	48	13	31	35	35	28	28	4	36	23	31	20
mod N	31	11	36	9	21	31	20	0	33	13	35	33	25	24	4	13	31	35	35	28	28	4	36	23	31	20
0 →N	31	11	36	9	21	31	20	44	33	13	35	33	25	24	4	13	31	35	35	28	28	4	36	23	31	20
C	Э	Й	1	З	У	Э	Т	9	Я	Л	0	Я	Ч	Ц	Г	Л	Э	0	0	Ъ	Ъ	Г	1	Х	Э	Т

Figure 2: работа алгоритма гаммирования

Подбор ключа

```
[16]: import string
      import random

[20]:
def function1(text):
    return ' '.join(hex(ord(i))[2:] for i in text)

def function2(size):
    return ''.join(random.choice(string.ascii_letters+string.digits) for _ in range(size))

def function3(text,key):
    return ''.join(chr(a^b) for a,b in zip (text,key))

def function4(text,encrypt):
    return ''.join(chr(a^b) for a,b in zip (text,encrypt))
```

Figure 3: подбор ключа

Определение вида шифротекста

```
message = 'С Новым годом, друзья!'
key  = function2(len(message))
hex_key = function1(key)
print("Используем ключ: ", key)
print("Ключ в шестнадцатиричном виде: ", hex_key)
encrypt = function3([ord(i) for i in message], [ord(i) for i in key])
hex_encrypt = function1(encrypt)
print("Зашифрованное: ", hex_encrypt)
decrypt = function3([ord(i) for i in encrypt], [ord(i) for i in key])
print("Расшифрованное: ", decrypt)
```

```
Используем ключ:  oASxvwD2hJADSEssddFqeX
Ключ в шестнадцатиричном виде:  6f 41 53 78 76 77 44 32 68 4a 41 44 53 45 73 73 64 64 46 71 65 58
Зашифрованное:  44e 61 44e 446 444 43c 478 12 45b 474 475 47a 46f 69 53 447 424 427 471 43d 42a 79
Расшифрованное:  С Новым годом, друзья!
```

Figure 4: определение вида шифротекста

Определение ключа

```
compute_key = function4([ord(i) for i in message], [ord(i) for i in encrypt])
decrypt_compute_key = function3([ord(i) for i in encrypt], [ord(i) for i in key])
print("Исходный ключ: ", key)
print("Вариант прочтения открытого текста: ", decrypt_compute_key)
```

```
Исходный ключ:  oASxvwD2hJADSEssddFqeX
Вариант прочтения открытого текста:  С Новым годом, друзья!
```

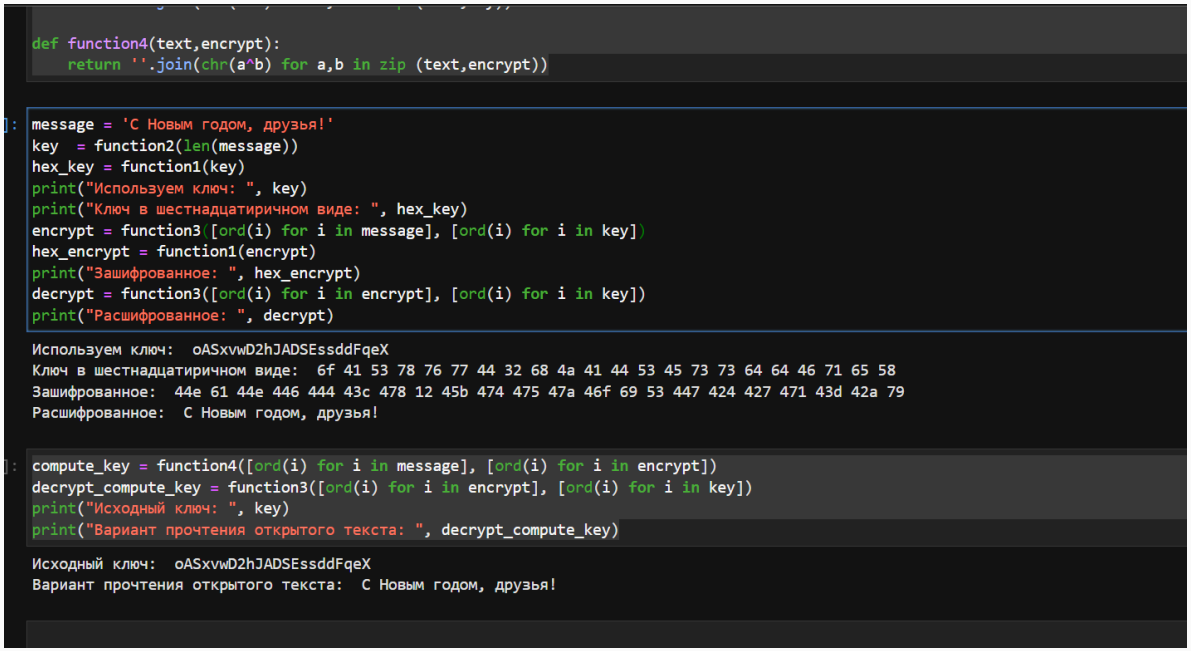
Figure 5: определение ключа

Формула

В аддитивных шифрах символы исходного сообщения заменяются числами, которые складываются по модулю с числами гаммы. Ключом шифра является гамма, символы которой последовательно повторяются. Перед шифрованием символы сообщения и гаммы заменяются их номерами в алфавите и само кодирование выполняется по формуле

$$Ci = (Ti + Gi) mod N$$

Пример работы алгоритма



```
def function4(text,encrypt):
    return ''.join(chr(a^b) for a,b in zip (text,encrypt))
```

```
message = 'С Новым годом, друзья!'
key  = function2(len(message))
hex_key = function1(key)
print("Используем ключ: ", key)
print("Ключ в шестнадцатиричном виде: ", hex_key)
encrypt = function3([ord(i) for i in message], [ord(i) for i in key])
hex_encrypt = function1(encrypt)
print("Зашифрованное: ", hex_encrypt)
decrypt = function3([ord(i) for i in encrypt], [ord(i) for i in key])
print("Расшифрованное: ", decrypt)
```

```
Используем ключ:  oASxvwD2hJADSEssddFqeX
Ключ в шестнадцатиричном виде:  6f 41 53 78 76 77 44 32 68 4a 41 44 53 45 73 73 64 64 46 71 65 58
Зашифрованное:  44e 61 44e 446 444 43c 478 12 45b 474 475 47a 46f 69 53 447 424 427 471 43d 42a 79
Расшифрованное:  С Новым годом, друзья!
```

```
compute_key = function4([ord(i) for i in message], [ord(i) for i in encrypt])
decrypt_compute_key = function3([ord(i) for i in encrypt], [ord(i) for i in key])
print("Исходный ключ: ", key)
print("Вариант прочтения открытого текста: ", decrypt_compute_key)
```

```
Исходный ключ:  oASxvwD2hJADSEssddFqeX
Вариант прочтения открытого текста:  С Новым годом, друзья!
```

Figure 6: работа алгоритма гаммирования на практике

Выводы

Результаты выполнения лабораторной работы

Изучили алгоритм шифрования с помощью гаммирования