МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«**Вятский государственный университет**»
Факультет автоматики и вычислительной техники
Кафедра электронных вычислительных машин

АЛГОРИТМЫ ШИФРОВАНИЯ

Отчёт по лабораторной работе №3 дисциплины
«Защита информации»

Выполнил студент группы ИВТ-42_____/Рзаев А.Э./
Проверил старший преподаватель _____/Караваева О.В./

Киров 2020

Задача №1. Алгоритмы замены.

Используя шифр простой замены одной буквы на соответствующую другую, ключ <оыблеякэщмчжьфхйвёстюършпизаднугц>, расшифруйте:

жмвм, июфм, стёяфмсн фцтяэхяч,
люем, сяейч бйейбйёйт,
еоч бяёмтн, птй ц тйэя вёяэхмч
стйг хое ёосвёяч вёяэхмр бйе!

Листинг программы:

```
def decrypt(ciphertext):
    KEY = {a: b for a, b in zip('оыблеякэщмчжьфхйвёстюършпизаднугц',
'абвгдеёжзийклмнопрстуфхцчшщъыьэюя')}

    plaintext = []
    for c in ciphertext:
        if c in KEY.keys():
            plaintext.append(KEY[c])
        else:
            plaintext.append(c)

    return ''.join(plaintext)

ciphertext = '''жмвм, июфм, стёяфмсн фцтяэхяч,
люем, сяейч бйейбйёйт,
еоч бяёмтн, птй ц тйэя вёяэхмч
стйг хое ёосвёяч вёяэхмр бйе!
'''

print(decrypt(ciphertext))
```

Задача 2. Алгоритм шифрования ГОСТ 28147-89.

Выполните два цикла алгоритма шифрования ГОСТ 28147 89 в режиме простой замены. Для получения 64 бит исходного текста используйте 6 букв, начиная с 4 буквы, из своих данных: Фамилии Имени Отчества. Для получения ключа (256 бит) используют текст, состоящий из 50 символов. Первый подключ содержит 8 символов, начиная с десятого.

Листинг программы:

```
def f(a, x):
    sbox = (
        (5, 15, 0, 12, 8, 4, 14, 2, 13, 11, 3, 6, 10, 1, 9, 7),
        (12, 13, 8, 14, 7, 5, 1, 6, 2, 3, 11, 0, 15, 9, 10, 4),
        (1, 10, 13, 9, 12, 0, 6, 3, 8, 15, 11, 2, 14, 4, 7, 5),
        (6, 9, 4, 2, 3, 13, 8, 14, 1, 11, 15, 7, 0, 10, 5, 12),
        (11, 0, 9, 10, 1, 2, 8, 12, 15, 3, 6, 4, 7, 5, 14, 13),
        (5, 15, 2, 1, 7, 14, 8, 6, 0, 10, 3, 9, 4, 13, 12, 11),
```

```
        (1, 0, 13, 4, 3, 2, 6, 11, 8, 5, 7, 14, 9, 10, 15, 12),
        (4, 5, 9, 11, 10, 1, 6, 13, 7, 2, 3, 0, 12, 14, 15, 8)
    )

    c = (a + x) & 0xFFFFFFFF
    seq = [(c >> (i * 4)) & 0xF for i in range(8)]
    seq = [sbox[i][seq[i]] for i in range(8)]

    out = reduce(lambda x, y: x | y, [seq[i] << (i * 4) for i in range(8)])
    return ((out >> 11) | (out << (32 - 11))) & 0xFFFFFFFF


def split(block):
    number = int.from_bytes(block, 'little')
    return number & 0xFFFFFFFF, (number >> 32) & 0xFFFFFFFF


def join(a, b):
    return ((b << 32) | a).to_bytes(8, 'little')


def flipbytes(block):
    a, b = split(block)
    return join(b, a)


def subkeys(key):
    K = [int.from_bytes(key[i * 4:i * 4 + 4], 'little') for i in range(8)]
    X = [K[i % 8] for i in range(24)] + [k for k in reversed(K)]

    return X


def encround(block, subkey):
    a, b = split(block)
    a, b = b ^ f(a, subkey), a

    return join(a, b)


def encrypt(block, key, rounds=32):
    for subkey in subkeys(key)[:rounds]:
        block = encround(block, subkey)
    return flipbytes(block)


def decrypt(block, key, rounds=32):
    for subkey in reversed(subkeys(key)[:rounds]):
        block = encround(block, subkey)
    return flipbytes(block)
```

```
plaintext = 'Рзаев Али Эльдар оглы'.encode('utf-8')[:8]
key = 'Результат разбивается на восемь'.encode('utf-8')[:32]

print('2 rounds')
print('plaintext:\t{}'.format(plaintext.hex()))
ciphertext = encrypt(plaintext, key, 2)
print('ciphertext:\t{}'.format(ciphertext.hex()))
plaintext = decrypt(ciphertext, key, 2)
print('plaintext:\t{}'.format(plaintext.hex()))
```

Задача 3. Алгоритм шифрования RSA.

Сгенерируйте открытый и закрытый ключи в алгоритме шифрования RSA, используя простые числа p=277 и q=521. Зашифруйте сообщение, состоящее из вашей фамилии и инициалов.

Листинг программы:

```
import math
import random

def isprime(num):
      if num <= 1:
            return False

      for i in range(2, math.ceil(math.sqrt(num))):
            if num % i == 0:
                  return False

      return True


def newkeys(p, q):
    n = p * q
    f = (p - 1) * (q - 1)

    E = [n for n in range(f) if isprime(n) and f % n != 0]
    e = random.choice(E)

    d = 1
    while (d * e) % f != 1:
        d += 1

    return (e, n), (d, n)


def encrypt(message, pubkey):
    e, n = pubkey
    return list(pow(ord(c), e, n) for c in message)
```

```
def decrypt(message, privkey):
    d, n = privkey
    return ''.join(chr(pow(c, d, n)) for c in message)


random.seed(0)
pubkey, privkey = newkeys(277, 521)
plaintext = 'Рзаев Али Эльдар оглы'
print('plaintext:\t{}'.format(plaintext))
ciphertext = encrypt(plaintext, pubkey)
print('ciphertext:\t{}'.format(ciphertext))
```

Задача 4. Функция хеширования.

Найти хеш–образ своей фамилии, используя хеш–функцию (выбрать свою) , где n = pq.

Листинг программы:

```
def hash_(message):
    p = 277
    q = 521

    hashvalue = 0
    for i, c in enumerate(message):
        hashvalue = (hashvalue + pow(p * q, i, 1 << 128) * ord(c) + 1) & ~(1
<< 128)
    return hashvalue.to_bytes(16, 'little')


plaintext = 'Рзаев'
print('plaintext:\t{}'.format(plaintext))
hashvalue = hash_(plaintext)
print('hash:\t{}'.format(hashvalue.hex()))
```

Задача 5. Электронная цифровая подпись.

Используя хеш-образ своей Фамилии, вычислите электронную цифровую подпись по схеме RSA.

Листинг программы:

```
import math
import random

def isprime(num):
      if num <= 1:
            return False

      for i in range(2, math.ceil(math.sqrt(num))):
```

```
            if num % i == 0:
                  return False

      return True


def newkeys(p, q):
    n = p * q
    f = (p - 1) * (q - 1)

    E = [n for n in range(f) if isprime(n) and f % n != 0]
    e = random.choice(E)

    d = 1
    while (d * e) % f != 1:
        d += 1

    return (e, n), (d, n)


def encrypt(message, pubkey):
    e, n = pubkey
    return list(pow(ord(c), e, n) for c in message)


def decrypt(message, privkey):
    d, n = privkey
    return ''.join(chr(pow(c, d, n)) for c in message)


def hash_(message):
    p = 277
    q = 521

    hashvalue = 0
    for i, c in enumerate(message):
        hashvalue = (hashvalue + pow(p * q, i, 1 << 128) * ord(c) + 1) & ((1
<< 128) - 1)
    return hashvalue.to_bytes(16, 'little')


random.seed(0)
pubkey, privkey = newkeys(277, 521)
plaintext = 'Рзаев Али Эльдар оглы'
print('plaintext:\t{}'.format(plaintext))
hashvalue = hash_(plaintext)
print('hash:\t{}'.format(hashvalue.hex()))
signature = encrypt(hashvalue.hex(), privkey)
print('signature:\t{}'.format(signature))
```

Выводы

В результате лабораторной работы были изучены и реализованы некоторые алгоритмы и способы шифрования данных. Реализован алгоритм RSA и алгоритм цифровой подписи. Для хеширования использовался полиноминальный хеш. Наиболее трудным в реализации оказался шифр блочный ГОСТ.