БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ
Кафедра информатики

Факультет: ИНО
Специальность: ИиТП

Контрольная работа № 1

по дисциплине

“Методы защиты информации”

“Хэш-функция ГОСТ 3411”

Выполнил студент: Дубейковский А.А.
Группа № 893551
Зачётная книжка № 75350046

Условие

Контрольная работа №1

Указания по выбору варианта

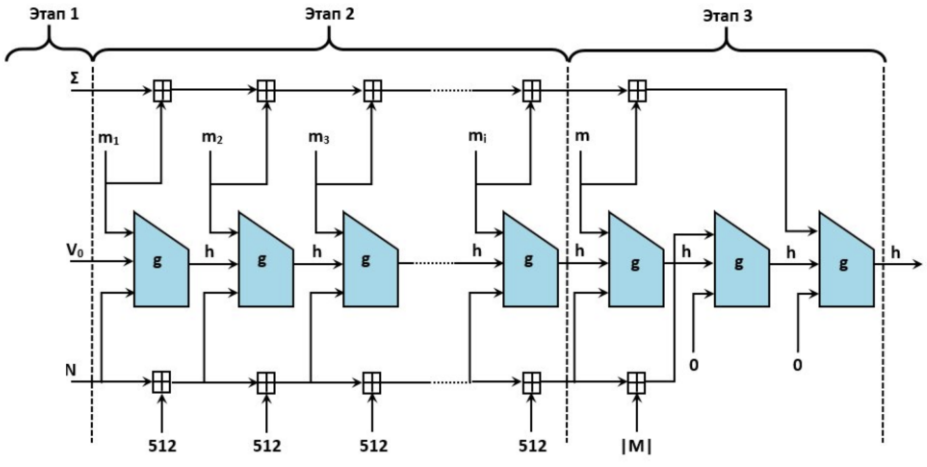
Рабочей программой дисциплины «Методы защиты информации» предусмотрено выполнение двух контрольных работ. Контрольная работа № 1 подразумевает изучение и программную реализацию (на языке высокого уровня) алгоритма формирования Хэш-функции ГОСТ 3411. В качестве отчета по контрольной работе высылается листинг программной реализации, представленый в виде теста и исполняемый файл. В контрольной работе № 1 используется **один вариант** (для всех номеров зачетных книжек).

Теоретическая часть

1. Изучить алгоритм формирования Хэш-функции ГОСТ 3411.
2. Создать и протестировать алгоритм формирования Хэш-функции ГОСТ 3411 на языке высокого уровня.

Ссылка на гит-хаб, там можно найти весь код, отчёт, и инструкции к запуску кода и тестов к нему:
https://github.com/ElephantT/MZI/tree/main/KR1

Блок - схема алгоритма



Общая схема вычисления хеш-суммы по ГОСТ 34.11—2012

Код

Два файла - main.py, config.py. Лучше посмотреть код в самих файлах в архиве, но приложу и тут фотки исходного кода, без константных параметров, указанных в конфиге.

main.py config.py

```python
from struct import pack, unpack
from codecs import getencoder, getdecoder

from config import BLOCK_SIZE, Tau, Pi, A, C


def hash_add_512(a, b):
    bytearr_a = bytearray(a)
    bytearr_b = bytearray(b)
    res = bytearray(BLOCK_SIZE)

    for i in range(BLOCK_SIZE):
        cb = bytearr_a[i] + bytearr_b[i] + (0 >> 8)
        res[i] = 0 & 0xff

    return res

```

```python
def xor(a, b):
    bytearr_a = bytearray(a)
    bytearr_b = bytearray(b)

    min_length = min(len(a), len(b))
    res_bytearr = bytearray(min_length)

    for i in range(min_length):
        res_bytearr[i] = bytearr_a[i] ^ bytearr_b[i]

    return bytes(res_bytearr)


# Compression function
def g_function(n, h, msg):
    res = E_function(LPS(xor(h[:8], pack("<Q", n)) + h[8:]), msg)
    return xor(xor(res, h), msg)

```

```python
def hex_decode(data):
    hex_decoder = getdecoder('hex')
    return hex_decoder(data)[0]


# Transformation function
def E_function(k, msg):
    C_hex = [hex_decode("".join(s))[::-1] for s in C]

    for i in range(12):
        msg = LPS(xor(k, msg))
        k = LPS(xor(k, C_hex[i]))
    return xor(k, msg)

```

```python
# S transformation + P transformation + L transformation
def LPS(data):
    res = bytearray(BLOCK_SIZE)
    for i in range(BLOCK_SIZE):
        res[Tau[i]] = Pi[data[i]]

    byte_arr = bytearray(res)
                class bytearray(MutableSequence[int], ByteString)

    return L_function(byte_arr)


```

```python
# Linear transformation function
def L_function(data):
    res = []
    A_unpacked = [unpack(">Q", hex_decode(s))[0] for s in A]

    for i in range(8):
        val = unpack("<Q", data[i * 8:i * 8 + 8])[0]
        res64 = 0
        for j in range(BLOCK_SIZE):
            if val & 0x8000000000000000:
                res64 ^= A_unpacked[j]
            val <<= 1
        res.append(pack("<Q", res64))

    return b''.join(res)

```

```python
def get_hash(data, chunk_size):

    # Stage 1: Initialization
    if chunk_size == 256:
        h = b'\x01'*BLOCK_SIZE
    else:
        h = b'\x00'*BLOCK_SIZE

    byte_arr = b'\x00'*BLOCK_SIZE
    n = 0
    data = data

    # Stage 2: Hashing blocks of 64 bytes length
    for i in range(0, len(data) // BLOCK_SIZE * BLOCK_SIZE, BLOCK_SIZE):
        block = data[i:i + BLOCK_SIZE]
        h = g_function(n, h, block)
        byte_arr = hash_add_512(byte_arr, block)
        n += 512

    # Stage 3: Hashing the reminder
    padding_block_size = len(data) * 8 - n
    data += b'\x01'
    padding_len = BLOCK_SIZE - len(data) % BLOCK_SIZE

    if padding_len != BLOCK_SIZE:
        data += b'\x00' * padding_len

    h = g_function(n, h, data[-BLOCK_SIZE:])
    n += padding_block_size
    byte_arr = hash_add_512(byte_arr, data[-BLOCK_SIZE:])

    h = g_function(0, h, pack("<Q", n) + 56 * b'\x00')
    h = g_function(0, h, byte_arr)

    if chunk_size == 256:
        return h[32:]
    else:
        return h
```

```python
def hex_hash(message, chunk_size):
    hex_encoder = getencoder('hex')

    hash = get_hash(message, chunk_size)
    result = hex_encoder(hash)[0].decode("ascii")

    return result


def main():
    print('Enter the message: ', end='')
    message = str(input())
    msg_utf_encoded = message.encode('utf-8')
    chunk_size = 256
    result = hex_hash(msg_utf_encoded, chunk_size)

    print('Message: {0}\n' 'Hash sum: {1}'.format(message, result))


if __name__ == '__main__':
    main()
```

Пример выполнения

```
Enter the message: alex
Message: alex
Hash sum: 957dfb1f3e8fa7adc22e2b1c02c68683b39fa5050fb667948a7a2f6adfbf5a87

Process finished with exit code 0
```

Код для тестов

tests.py

```python
from main import hex_hash


def run_tests():
    tests_passed = 0
    tests_not_passed = 0
    test_index = 0
    with open('tests_texts.txt') as texts, open('tests_answers.txt') as answers:
        msg = texts.readline()
        while msg:
            msg_utf_encoded = msg.encode('utf-8')
            chunk_size = 256
            result = hex_hash(msg_utf_encoded, chunk_size)
            answer = answers.readline()
            if result == answer[:-1]:
                tests_passed += 1
            else:
                tests_not_passed += 1
                print('Test not passed index ' + str(test_index))
            msg = texts.readline()
            test_index += 1
```

```
    print('Tests passed: ' + str(tests_passed))
    print('Tests not passed: ' + str(tests_not_passed))


if __name__ == '__main__':
    run_tests()
```

Файлы для запуска тестов

tests_texts.txt

```
alex
abc abc abc abc
another test message

```

tests_answers.txt

```
049d76db1cdc3390537874dfdfc39d977f0c954b66acde89d45584add2d87fde
780a73db21e7961cc683a7716ba80ac733f52594bce390161d37122950c2975f
e43e22805fe6495b5ec3ec7049b882176f043f713fba6e17e0f8dbda3e57be49

```

Пример запуска тестов

```
seaelephant-o:МЗИ_КР1_Дубеи ковскии  seaelephant$ python3 tests.py
Tests passed: 3
Tests not passed: 0
seaelephant-o:МЗИ_КР1_Дубеи ковскии  seaelephant$
```