Отчёт по курсовому проекту по курсу «Криптография»

Выполнил Попов Николай, М8О-308Б-21

Задание

1. Строку в которой записано своё ФИО подать на вход в хеш-функцию ГОСТ Р 34.11-2012 (Стрибог). Младшие 4 бита выхода интерпретировать как 16-тиричное число, которое в дальнейшем будет номером варианта.

2. Программно реализовать один из алгоритмов функции хеширования в соответствии с номером варианта. Алгоритм содержит в себе несколько раундов.

3. Модифицировать оригинальный алгоритм таким образом, чтобы количество раундов было настраиваемым параметром программы. в этом случае новый алгоритм не будет являться стандартом, но будет интересен для исследования.

4. Применить подходы дифференциального криптоанализа к полученным алгоритмам с разным числом раундов.

5. Построить график зависимости количества раундов и возможности различения отдельных бит при количестве раундов 1,2,3,4,5,... .

6. Сделать выводы.

Ход работы

1. Определение варианта:

```
from pygost import gost34112012256

fio = "Попов Николай Александрович"
fio_bytes = fio.encode("utf-8")
hash_result = gost34112012256.new(fio_bytes).digest()

print(hash_result)
variant_number = hash_result[-1] & 0x0F
print(variant_number)
```

```
~/Desktop/crypto-cp ❯ python variant.py
b'z~\xf6\x9f\xef\\\xd0\x9c\x86]E#\x9a\x0e\x02\xeb\x08e\x1b\xb6\x8cIc\xc6\x90~d\xb8\x19v\xae]'
13
~/Desktop/crypto-cp ❯
```

13 вариант соответствует алгоритму MD5

2. Программная реализация:

Алгоритм был взят из бибилиотеки hashlib

```
def md5_built_in(s):
    return hashlib.md5(s).hexdigest()
```

3. Программная реализация модифицированного алгоритма:

```
rotate_amounts = [7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22, 7, 12, 17, 22,
                  5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20, 5, 9, 14, 20,
                  4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23, 4, 11, 16, 23,
                  6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21, 6, 10, 15, 21]

tsin = [int(abs(math.sin(i + 1)) * 2 ** 32) & 0xFFFFFFFF for i in range(64)]

init_values = [0x67452301, 0xefcdab89, 0x98badcfe, 0x10325476]

functions = [lambda b, c, d: (b & c) | (~b & d),
             lambda b, c, d: (d & b) | (~d & c),
             lambda b, c, d: b ^ c ^ d ,
             lambda b, c, d: c ^ (b | ~d)]

index_functions = [lambda i: i,
                   lambda i: (5 * i + 1) % 16,
                   lambda i: (3 * i + 5) % 16,
                   lambda i: (7 * i) % 16]


def left_rotate(x, amount):
    x &= 0xFFFFFFFF
    return ((x << amount) | (x >> (32 - amount))) & 0xFFFFFFFF
```

```
def md5(message_, rounds=4):
    message = bytearray(message_)
    orig_len_in_bits = (8 * len(message)) & 0xffffffffffffffff
    message.append(0x80)
    while len(message) % 64 != 56:
        message.append(0)
    message += orig_len_in_bits.to_bytes(8, byteorder='little')

    hash_pieces = init_values[:]

    for chunk_ofst in range(0, len(message), 64):
        a, b, c, d = hash_pieces
        chunk = message[chunk_ofst:chunk_ofst + 64]
        for rr in range(rounds):
            r = rr % 4
            for kk in range(16):
                i = r*16 + kk
                k = index_functions[r](i)
                x_k = int.from_bytes(chunk[4*k : 4*k+4], byteorder='little')
                f = functions[r](b, c, d)
                to_rotate = a + f + tsin[i] + x_k
                new_b = (b + left_rotate(to_rotate, rotate_amounts[i])) & 0xFFFFFFFF
                a, b, c, d = d, new_b, b, c
        for i, val in enumerate([a, b, c, d]):
            hash_pieces[i] += val
            hash_pieces[i] &= 0xFFFFFFFF

    return sum(x << (32 * i) for i, x in enumerate(hash_pieces))
```

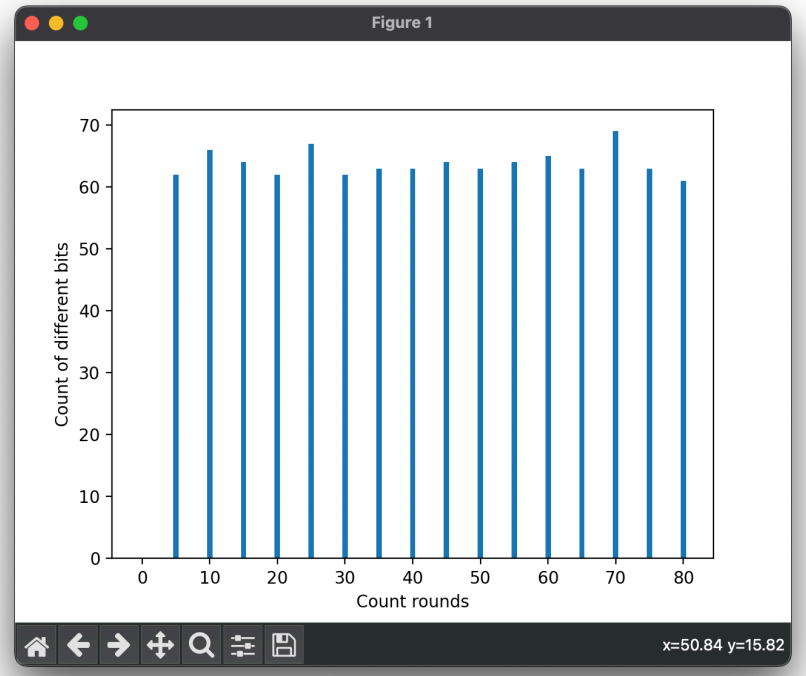
4. Дифференциальный криптоанализ

Я генерирую строку и создаю её копию с инвертированным последним битом. Затем вычисляю хэши обеих строк для различных значений количества раундов, начиная с 0 и до 80 с шагом 5. После этого считаю количество отличающихся битов в полученных хэшах. Чтобы провести более объективный анализ, я повторяю эту процедуру 10 раз и вычисляю среднее количество изменённых битов для каждого количества раундов.

Количество раундов	**Количество различающихся бит**
0	0
5	64
10	64
15	64
20	63
25	62
30	64
35	62
40	64
45	64
50	66
55	66
60	65

65	68
70	64
75	62
80	65

5. График зависимости между кол-вом раундов и кол-вом различающихся бит



6. Вывод

В целом, хеш-функция демонстрирует хорошую способность к перемешиванию битов, начиная с ранних этапов (5 раундов) и поддерживая это на протяжении всех 80 раундов. Среднее количество изменённых битов колеблется вокруг 64, что соответствует ожидаемому значению для хорошей хеш-функции, где приблизительно половина битов должна изменяться при небольшом изменении входного сообщения.
Это подчеркивают эффективность функции хеширования MD5 в обеспечении значительного уровня перемешивания битов даже при небольшом количестве раундов и устойчивость этого уровня перемешивания при увеличении количества раундов.