

**Московский авиационный институт  
(национальный исследовательский университет)**

**Факультет информационных технологий и прикладной  
математики**

**Кафедра вычислительной математики и программирования**

**Лабораторная работа №3 по курсу Криптография**

Студент: В. А. Петросян  
Преподаватель: А. В. Борисов  
Группа: М8О-308Б  
Дата:  
Оценка:  
Подпись:

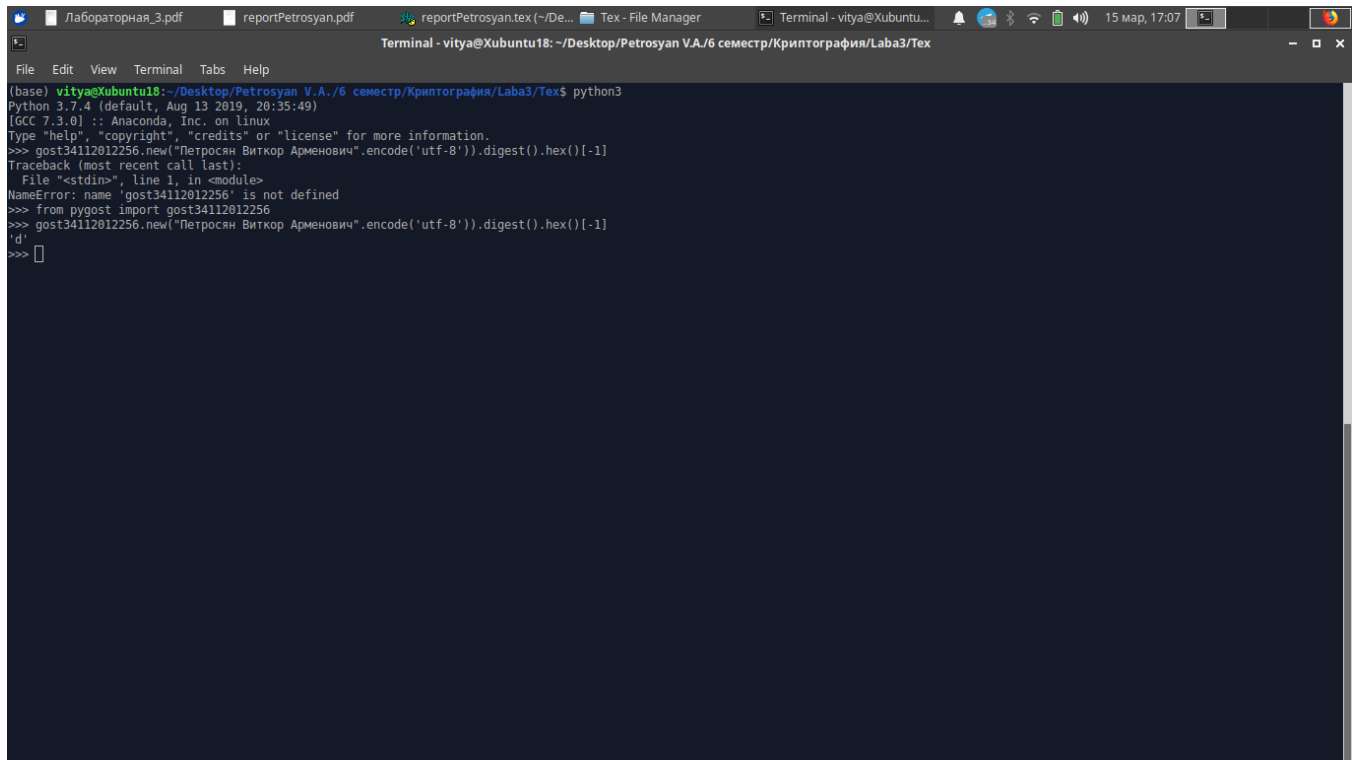
**Москва, 2020**

## Условие

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 число, которое в дальнейшем будет номером варианта. Процесс выбора варианта требуется отразить в отчёте.
2. Программно реализовать один из алгоритмов функции хеширования в соответствии с номером варианта. Алгоритм содержит в себе несколько раундов. Допустимо использовать сторонние реализации, при условии, что они проходят тесты из стандарта и пригодны для дальнейшей модификации
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. Если в алгоритме описывается семейство с разными размерами блоков, то можно выбрать любой из них.
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.

## Метод решения

скрин запуска Госта с b"Петросян Виткор Арменович"



```
Terminal - vitya@Xubuntu18: ~/Desktop/Petrosyan V.A./6 семестр/Криптография/Laba3/Tex
File Edit View Terminal Tabs Help
(base) vitya@Xubuntu18:~/Desktop/Petrosyan V.A./6 семестр/Криптография/Laba3/Tex$ python3
Python 3.7.4 (default, Aug 13 2019, 20:35:49)
[GCC 7.3.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license()" for more information.
>>> gost34112012256.new("Петросян Виткор Арменович".encode('utf-8')).digest().hex()[1]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'gost34112012256' is not defined
>>> from pygost import gost34112012256
>>> gost34112012256.new("Петросян Виткор Арменович".encode('utf-8')).digest().hex()[1]
'd'
```

D == MD5

Сначала я нашёл в интернете реализацию MD5 на Python. Проверил, что она нормально работает, сравнив результат запуска на компе и на каком-то онлайн сайте одной и той же строки. Поменял немного программу ручками, чтобы можно было задавать количество раундов вручную. Я добавил переменную `stopAtRound`. Снова проверил при 4-х раундах результат работы программы.

После написал две программы на C++:

### 1. generate.cpp

Генерирует случайное 128 битное число и пишет в файл `TestData.txt` после также пишет в этот файл 128 чисел, которые отличаются от первого только в одном конкретном бите. В итоге получается  $129 * n$  строк 128-битных чисел.

### 2. Analyze.cpp

Читает из файла `Result.txt` блоки строк чисел размера 129. Сначала считывает первое число. После в цикле остальные 128 и сравнивает сколько битов отличается от первого в блоке. Результат своей работы пишет в файл `DataToPlot.txt`.

Чтобы не мучаться с запуском я написал `work.sh` который запускает это всё веселье.

## Листинг программного кода алгоритма MD5 хеширования

Program MD5.py

---

```
import math

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]

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

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

functions = 16*[lambda b, c, d: (b & c) | (~b & d)] + \
              16*[lambda b, c, d: (d & b) | (~d & c)] + \
              16*[lambda b, c, d: b ^ c ^ d] + \
              16*[lambda b, c, d: c ^ (b | ~d)]

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

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

def md5(message , stopAtRound = 4):

    message = bytearray(message) #copy our input into a mutable buffer
    orig_len_in_bits = (8 * len(message)) & 0xffffffff
    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 i in range( stopAtRound * 16 ):
    f = functions[i](b, c, d)
    g = index_functions[i](i)
    to_rotate = a + f + constants[i] + \
    int.from_bytes(chunk[4*g:4*g+4], byteorder='little')
    new_b = (b + left_rotate(to_rotate, rotate_amounts[i])) & 0xFFFFF
    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))

def md5_to_hex(digest):
    raw = digest.to_bytes(16, byteorder='little')
    return '{:032x}'.format(int.from_bytes(raw, byteorder='big'))

if __name__ == '__main__':

    Data = open("TestData.txt", "r")
    if Data.mode == 'r':
        contents = Data.read()
    tmp = [ bytearray(x, 'ascii') for x in contents.split()]

    '''
demo = [b"Victor", b"Petrosyan", b"Petrosyan", b"petrosyan"]
'''

stopAtRound = int( input("When_stop?\nAt_round_#_") )
size = len(tmp)

Output = open("Result.txt", "w")
if Output.mode == 'w':
    Output.write(str(size) + "_" + str(stopAtRound) + "\n")
    for i in range(size):
        Output.write(md5_to_hex(md5(tmp[i], stopAtRound)))
        Output.write("\n")
Output.close()

```

Script work.sh

---

```
#!/bin/bash
g++ generate.cpp -o genData
./genData
python3 ./MD5.py
g++ Analyze.cpp -o countData
./countData
```

## Выводы

Я изучил один из методов дифференциального криптоанализа, а также узнал чуть больше про алгоритм хэширования MD5, пока искал в интернете нужную информацию о нём. MD5(message digest 5) это 128-битный алгоритм хеширования, изобретённый в 1991 году в MIT. Предназначен для создания «отпечатков» или дайджестов сообщения произвольной длины и последующей проверки их подлинности. Широко применялся для проверки целостности информации и хранения хешей паролей. У алгоритма всего 4 раунда. Результаты моего крипто исследования представлены на гистограмме.

