Московский авиационный институт

(государственный технический университет)

**Факультет прикладной математики и физики**

Кафедра вычислительной математики и программирования

**Лабораторная работа №4**

по спецкурсу «Криптография»:  
Криптографические алгоритмы

|  |  |
| --- | --- |
| Выполнил: | Баскаков О.А. |
| Группа: | 08-306 |
| № по списку: | 2 |
| Преподаватель: | Рисенберг Д.В. |
| Оценка: |  |
| Дата: |  |

Москва

2011 г.

# Задание

Необходимо написать программу на языке C++, C# или Python, реализующую алгоритм.

*Вариант №3.*

Симметричная схема шифрования IDEA.

### Описание

IDEA (англ. International Data Encryption Algorithm, международный алгоритм шифрования данных) — симметричный блочный алгоритм шифрования данных, запатентованный швейцарской фирмой Ascom. Известен тем, что применялся в пакете программ шифрования PGP.

Так как IDEA использует 128-битный ключ и 64-битный размер блока, открытый текст разбивается на блоки по 64 бит. Если такое разбиение невозможно, используются различные режимы шифрования. Каждый исходный незашифрованный 64-битный блок делится на четыре подблока по 16 бит каждый, так как все алгебраические операции, использующиеся в процессе шифрования, совершаются над 16-битными числами. Для шифрования и расшифрования IDEA использует один и тот же алгоритм.

Фундаментальным нововведением в алгоритме является использование операций из разных алгебраических групп, а именно:

* сложение по модулю
* умножение по модулю
* побитовое исключающее ИЛИ (XOR).

Эти три операции несовместимы в том смысле, что:

* никакие две из них не удовлетворяют дистрибутивному закону;
* никакие две из них не удовлетворяют ассоциативному закону;

Применение этих трех операций затрудняет криптоанализ IDEA по сравнению с DES, который основан исключительно на операции исключающее ИЛИ, а также позволяет отказаться от использования S-блоков и таблиц замены. IDEA является модификацией сети Фейстеля.

### Исходный код на языке Python:

#! /usr/bin/python3.1

# -\*- coding: utf-8 -\*-

SHIFT = 25

def get\_inv(a, b = (2\*\*16) +1):

if (a == 0): return 0

for i in range(0, b):

if ( (a\*i)%b == 1): return i

return None

def get\_sub(a, b = 2\*\*16):

return (b - a) % b

def mul(a,b):

return mod( a \* b, 2\*\*16 + 1)

def add(a,b):

return mod( a + b, 2\*\*16)

def low128(x):

return ((x) & 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF)

#~ процедура генерации 52 ключей выглядит очень запутанно

def ExpandKey(userkey):

EK = []

# mixing keys

for j in range(8):

EK.append( low16(userkey>>(16\*j)) )

#step 2

userkey = low128(userkey << SHIFT) | low128(userkey >> (128-SHIFT))

for j in range(2,8):

EK.append( low16(userkey>>(16\*j)) )

for j in range(2):

EK.append( low16(userkey>>(16\*j)) )

#step 3

userkey = low128(userkey << SHIFT) | low128(userkey >> (128-SHIFT))

for j in range(6,8):

EK.append( low16(userkey>>(16\*j)) )

for j in range(6):

EK.append( low16(userkey>>(16\*j)) )

#step 4

userkey = low128(userkey << SHIFT) | low128(userkey >> (128-SHIFT))

for j in range(0,8):

EK.append( low16(userkey>>(16\*j)) )

#step 5

userkey = low128(userkey << SHIFT) | low128(userkey >> (128-SHIFT))

for j in range(4,8):

EK.append( low16(userkey>>(16\*j)) )

for j in range(4):

EK.append( low16(userkey>>(16\*j)) )

#step 6

userkey = low128(userkey << SHIFT) | low128(userkey >> (128-SHIFT))

for j in range(6,8):

EK.append( low16(userkey>>(16\*j)) )

for j in range(6):

EK.append( low16(userkey>>(16\*j)) )

#step 7

userkey = low128(userkey << SHIFT) | low128(userkey >> (128-SHIFT))

for j in range(2,8):

EK.append( low16(userkey>>(16\*j)) )

ZK = [ (EK[6\*i:6\*(i+1)]) for i in range(8)]

ZK.append(EK[48:52])

return ZK

def InverseKey(userkey):

ZK = ExpandKey(userkey)

IK = []

for i in range(8, 0, -1):

IK.append( ZK[i][0:4] + ZK[i-1][4:6] )

IK.append( ZK[0][0:4] )

for i in range(9):

IK[i][0] = get\_inv( IK[i][0] )

IK[i][1] = get\_sub( IK[i][1] )

IK[i][2] = get\_sub( IK[i][2] )

IK[i][3] = get\_inv( IK[i][3] )

#swap

if(0<i<8): (IK[i][1], IK[i][2]) = (IK[i][2], IK[i][1])

return IK

#~ Само шифрование наоборот просто и умещается в 18 строк кода

def IDEA(D, K):

D = copy(D) #does not matter

for i in range(8):

a = mul(D[0], K[i][0])

b = add(D[1], K[i][1])

c = add(D[2], K[i][2])

d = mul(D[3], K[i][3])

e = a ^ c

f = b ^ d

g = mul( add( f, mul(e, K[i][4])) , K[i][5])

h = add( mul(e, K[i][4]), g)

D[0] = a ^ g

D[1] = c ^ g

D[2] = b ^ h

D[3] = d ^ h

a = mul(D[0], K[8][0])

b = add(D[2], K[8][1])

c = add(D[1], K[8][2])

d = mul(D[3], K[8][3])

return [a,b,c,d]

### Тестирование:

oleg@debian:~/crypto$ ./4.py -help

International Data Encryption Algorithm

-help for this text

-encode key128 text64

-decode key128 text64

text64 - test with userr text

none - standart test

oleg@debian:~/crypto$ ./4.py

key 52:

0001 0002 0003 0004 0005 0006

0007 0008 0400 0600 0800 0a00

0c00 0e00 1000 0200 0010 0014

0018 001c 0020 0004 0008 000c

2800 3000 3800 4000 0800 1000

1800 2000 0070 0080 0010 0020

0030 0040 0050 0060 0000 2001

4000 6000 8000 a000 c000 e000

0080 00c0 0100 0140

key inverse:

fe01 ff40 ff00 659a c000 e000

fffd 8000 a000 cccc 0000 2001

a556 ffb0 ffc0 52ab 0010 0020

554b ff90 e000 fe01 0800 1000

332d c800 d000 fffd 0008 000c

4aab ffe0 ffe4 c001 0010 0014

aa96 f000 f200 ff81 0800 0a00

4925 fc00 fff8 552b 0005 0006

0001 fffe fffd c001

testing...

IDEA:

000a 000b 000c 000d

RES:

6623 7a88 1879 e3a5

RES-1:

000a 000b 000c 000d

oleg@debian:~/crypto$ ./4.py 0xFFFFFFFFFFFFFFFF

word= 0xFFFFFFFFFFFFFFFF

testing...

IDEA:

ffff ffff ffff ffff

RES:

2dde 0cd2 9778 92ef

RES-1:

ffff ffff ffff ffff

#~ Требуется чтобы ключ содержал более 48 бит.

oleg@debian:~/crypto$ ./4.py 1234567

word= 1234567

uncorrect word

oleg@debian:~/crypto$ ./4.py 0x1234567901234

word= 0x1234567901234

testing...

IDEA:

0001 2345 6790 1234

RES:

71ea c457 00c8 00df

RES-1:

0001 2345 6790 1234

#~ Шифрование и дешифрование данным с пользовательским ключем.

oleg@debian:~/crypto$ ./4.py encode 0x12345678901234567890123456789012 0x1234567890123456

encoding

key = 12345678901234567890123456789012

word= 1234567890123456

IDEA:

5865 90c0 2f02 fdda

oleg@debian:~/crypto$ ./4.py decode 0x12345678901234567890123456789012 0x586590c02f02fdda

decoding

key = 12345678901234567890123456789012

word= 586590c02f02fdda

IDEA^-1:

1234 5678 9012 3456

# Результат

Алгоритм IDEA является торговой маркой и запатентован во многих странах. Действие патента истекает в 2010—2011 годах. В Мае 2005 года MediaCrypt официально представила новый шифр IDEA NXT, призванный заменить IDEA.

Типичные области применения IDEA:

* шифрование аудио и видео данных для кабельного телевидения, видеоконференций, дистанционного обучения, VoIP;
* защита коммерческой и финансовой информации, отражающей конъюнктурные колебания;
* смарт-карты;
* линии связи через модем, роутер или ATM линию, GSM технологию;
* общедоступный пакет конфиденциальной версии электронной почты PGP v2.0.