Учреждение образования

«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ»

Кафедра информатики

Отчет по лабораторной работе №1

Выполнил:

студент гр. 953504

Кондрашов И.Д.

Проверил:

Протько М.И.

Минск 2022

**Оглавление**

[**Оглавление** 2](#_Toc116323918)

[**1. Постановка задачи** 3](#_Toc116323919)

[**2. Теоретическая сведения** 3](#_Toc116323920)

[**Алгоритм шифрования DES.** 3](#_Toc116323921)

[**Начальная и конечная перестановки.** 4](#_Toc116323922)

[**Создание подключей.** 7](#_Toc116323923)

[**Двойной DES.** 8](#_Toc116323924)

[**Тройной DES.** 8](#_Toc116323925)

[**3. Блок-схема алгоритма** 10](#_Toc116323926)

[**Стандарт шифрования ГОСТ 28147-89** 11](#_Toc116323927)

[**Схема алгоритма ГОСТ 28147-89** 12](#_Toc116323928)

[**Описание работы ГОСТ 28147-89** 12](#_Toc116323929)

[**Описание ГОСТ 28147-89** 14](#_Toc116323930)

[**ГОСТ 28147-89 Режимы работы** 15](#_Toc116323931)

[**4. Пример работы программы** 16](#_Toc116323932)

[**5. Код программы** 17](#_Toc116323933)

[**6. Вывод** 18](#_Toc116323934)

# **1. Постановка задачи**

Необходимо реализовать программные средства шифрования и дешифрования текстовых файлов при помощи алгоритмов DES и *ГОСТ 28147-89.*

**ЗАДАНИЕ:**

1. Изучить теоретические сведения.
2. Создать программы, читающие данные из файла и шифрующие (дешифрующие) их с помощью с помощью DES (двойной и тройной) и *ГОСТ 28147-89*.

# **2. Теоретическая сведения**

## **Алгоритм шифрования DES.**

**DES** (Data Encryption Standard) — алгоритм для симметричного шифрования, в котором данные шифруются 64-битными блоками и специальным ключом с длиной 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки E, IP, IP-1) преобразований. Алгоритм преобразует за несколько раундов 64-битный вход в 64-битный выход.

Процесс шифрования состоит из четырех этапов:

1. Выполняется начальная перестановка (IP) 64-битного исходного текста (забеливание), во время которой биты переупорядочиваются в соответствии со стандартной таблицей.
2. Этап состоит из 16 раундов одной и той же функции, которая использует операции сдвига и подстановки.
3. Левая и правая половины выхода последней (16-й) итерации меняются местами.
4. Выполняется перестановка IP-1 результата, полученного на третьем этапе. Перестановка IP-1 инверсна начальной перестановке.

Прямым развитием DES в настоящее время является алгоритм Double DES (2DES) и Triple DES (3DES). В них шифрование/расшифровка выполняются путём двукратного и трехкратного выполнения алгоритма DES соответственно.

Схема шифрования с помощью алгоритма DES:



Рисунок 1 – Схема шифрования с помощью алгоритма DES.

## **Начальная и конечная перестановки.**

Начальная перестановка и ее инверсия определяются стандартной таблицей. Если М – это произвольные 64 бита, то X = IP (M) – это переставленные 64 бита. Если применить обратную функцию перестановки Y = IP-1 (X) = IP-1 (IP(M)), то получится первоначальная последовательность бит.

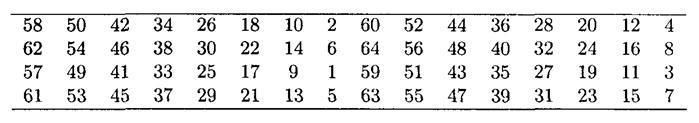


Рисунок 2 - DES. Начальная перестановка

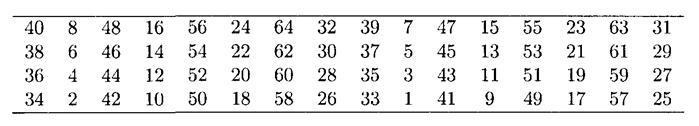


Рисунок 3 - DES. Заключительная перестановка

**Последовательность преобразований отдельного раунда.**

Теперь рассмотрим последовательность преобразований, используемую в каждом *раунде*.

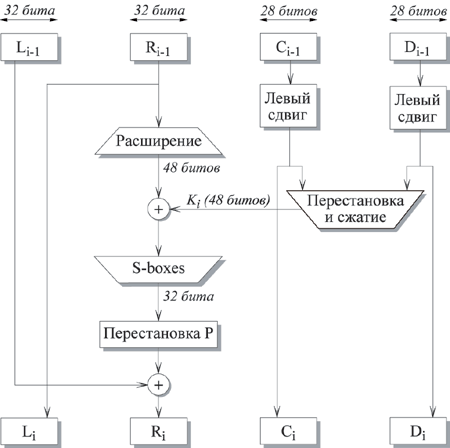


Рисунок 4 - i-ый раунд DES

64-битный входной блок проходит через 16 *раундов*, при этом на каждой итерации получается промежуточное 64-битное значение. Левая и правая части каждого промежуточного значения трактуются как отдельные 32-битные значения, обозначенные *L* и *R*.

Каждую итерацию можно описать следующим образом:

Li = Ri-1

Ri = Li-1 F(Ri-1, Ki) где, – это операцию XOR.

Таким образом, выход левой половины *Li* равен входу правой половины *Ri-1*. Выход правой половины *Ri*является результатом применения операции XOR к *Li-1* и функции *F*, зависящей от *Ri-1* и *Ki*.

Рассмотрим функцию *F* более подробно:

*Ri*, которое подается на вход функции *F*, имеет длину 32 бита. Вначале Ri расширяется до 48 бит, используя таблицу, которая определяет перестановку плюс расширение на 16 бит. Расширение происходит следующим образом. 32 бита разбиваются на группы по 4 бита и затем расширяются до 6 бит, присоединяя крайние биты из двух соседних групп. Например, если часть входного сообщения:

**. . . efgh ijkl mnop . . .**

то в результате расширения получается сообщение:

**. . . defghi hijklm lmnopq . . .**

После этого для полученного 48-битного значения выполняется операция XOR с 48-битным *подключом Ki*. Затем полученное 48-битное значение подается на вход функции подстановки, результатом которой является 32-битное значение.

Подстановка состоит из восьми *S-boxes,* каждый из которых на входе получает 6 бит, а на выходе создает 4 бита. Эти преобразования определяются специальными таблицами. Первый и последний биты входного значения *S-box* определяют номер строки в таблице, средние 4 бита определяют номер столбца. Пересечение строки и столбца определяет 4-битный выход. Например, если входом является 011011, то номер строки равен 01 (строка 1) и номер столбца равен 1101 (столбец 13). Значение в строке 1 и столбце 13 равно 5, т.е. выходом является 0101.

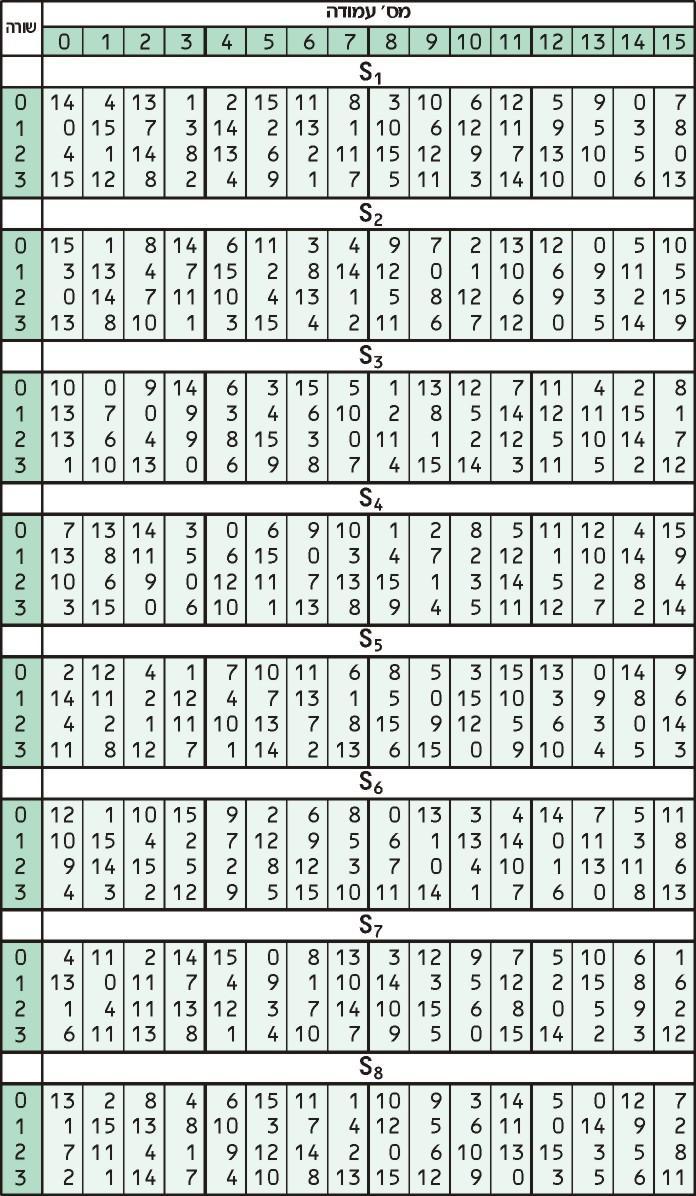


Рисунок 5 - S-boxes

Далее полученное 32-битное значение обрабатывается с помощью перестановки *Р*, целью которой является максимальное переупорядочивание бит, чтобы в следующем *раунде* шифрования с большой вероятностью каждый бит обрабатывался другим *S-box*.

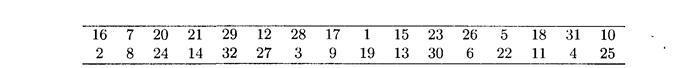


Рисунок 6 - Перестановка с помощью Р-блоков

## **Создание подключей.**

Ключ для отдельного *раунда Ki*состоит из 48 бит. Ключи *Ki* получаются по следующему алгоритму. Для 56-битного ключа, используемого на входе алгоритма (если используется 64-битный ключ, то, как видно из рис. 6 убираются биты 64, 56, 48, 40, 32, 16, 8), вначале выполняется перестановка в соответствии с таблицей Permuted Choice 1 (РС-1).

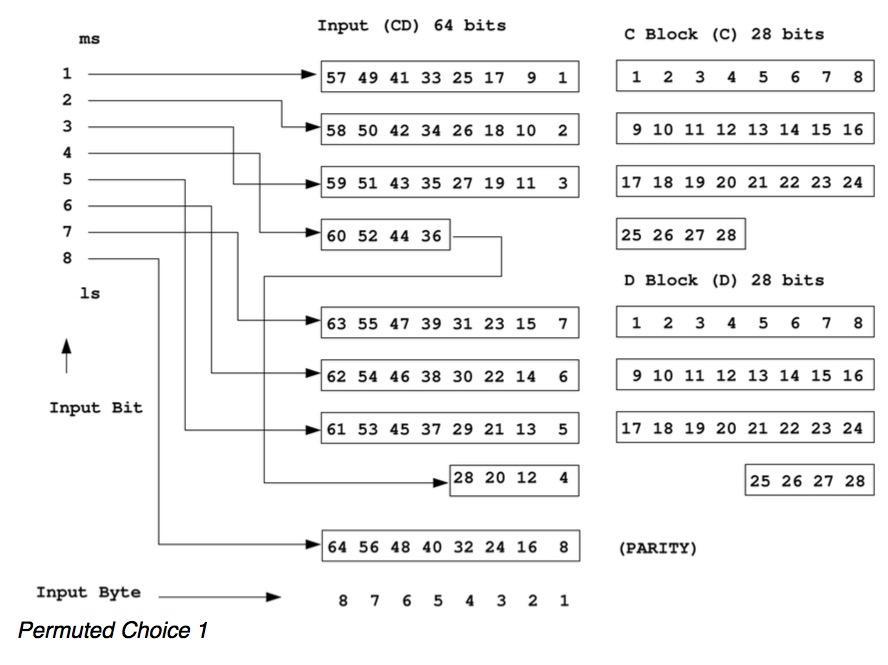


Рисунок 7 - Схема Permuted Choice

Полученный 56-битный ключ разделяется на две 28-битные части, обозначаемые как C0 и D0 соответственно. На каждом *раунде Ci* и *Di*независимо циклически сдвигаются влево на 1 или 2 бита, в зависимости от номера *цикла*.

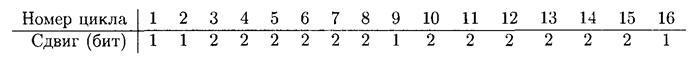


Рисунок 8 - Сдвиг ключа в зависимости от номера цикла

Полученные значения являются входом следующего *раунда*. Они также представляют собой вход в Permuted Choice 2 (РС-2), который создает 48-битное выходное значение, являющееся входом функции *F*(*Ri-1*, *Ki*).

Процесс ***дешифрования*** аналогичен процессу шифрования. На входе алгоритма используется зашифрованный текст, но ключи *Ki* используются в обратной последовательности. *K16* используется на первом *раунде*, *K1* используется на последнем *раунде*.

## **Двойной DES.**

Наиболее логичным способом противодействия полному перебору ключа DES выглядит многократное шифрование данных алгоритмом DES с различными ключами. Следующий алгоритм получил название Double DES (двойной DES):

где: — половины двойного ключа алгоритма Double DES, каждая из которых представляет собой обычный 56-битный ключ DES; Е — функция шифрования блока данных обычным алгоритмом DES. Если бы при двойном шифровании DES выполнялось следующее свойство: для любых значений, то двойное шифрование не приводило бы к усилению против полного перебора ключа — всегда нашелся бы такой ключ *k*, *однократное* шифрование которым было бы эквивалентно двукратному шифрованию на ключах, а для нахождения ключа к достаточно было бы перебрать 255 ключей. Double DES действительно удваивает эффективный размер ключа— до 112 битов, а при современном развитии вычислительной техники полный перебор 112-битного ключа невозможен.

## **Тройной DES.**

Triple DES (3DES) — симметричный [блочный шифр](https://ru.wikipedia.org/wiki/%D0%91%D0%BB%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%88%D0%B8%D1%84%D1%80), созданный [Уитфилдом Диффи](https://ru.wikipedia.org/wiki/%D0%94%D0%B8%D1%84%D1%84%D0%B8,_%D0%A3%D0%B8%D1%82%D1%84%D0%B8%D0%BB%D0%B4), [Мартином Хеллманом](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D0%BB%D0%BB%D0%BC%D0%B0%D0%BD,_%D0%9C%D0%B0%D1%80%D1%82%D0%B8%D0%BD) и Уолтом Тачманном в [1978 году](https://ru.wikipedia.org/wiki/1978_%D0%B3%D0%BE%D0%B4) на основе алгоритма [DES](https://ru.wikipedia.org/wiki/DES) с целью устранения главного недостатка последнего — малой длины ключа (56 бит), который может быть взломан методом [полного перебора](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D0%B5%D0%B1%D0%BE%D1%80) ключа. Скорость работы 3DES в 3 раза ниже, чем у DES, но криптостойкость намного выше — время, требуемое для криптоанализа 3DES, может быть в миллиард раз больше, чем время, нужное для вскрытия DES. 3DES используется чаще, чем DES, который легко взламывается при помощи сегодняшних технологий (в [1998 году](https://ru.wikipedia.org/wiki/1998_%D0%B3%D0%BE%D0%B4) организация [Electronic Frontier Foundation](https://ru.wikipedia.org/wiki/Electronic_Frontier_Foundation), используя специальный компьютер *DES Cracker*, вскрыла DES за 3 дня). 3DES является простым способом устранения недостатков DES. Алгоритм 3DES построен на основе DES, поэтому для его реализации возможно использовать программы, созданные для DES.

Официальное название алгоритма, используемое в стандартах - TDEA или Triple DEA (англ. Triple Data Encryption Algorithm). Однако, термин "3DES" используется более широко поставщиками, пользователями и разработчиками криптосистем.

Существует 2 основных варианта шифрования алгоритмом 3DES: 3-key Triple DES и 2-key Triple DES. Как видно из названия, принципиальное различие этих методов - количество ключей (три и два соответственно). В свою очередь, каждый из этих алгоритмов имеет по 2 разных типа: EEE (encryption-encryption-encryption) и EDE (encryption-decryption-encryption) шифрование.

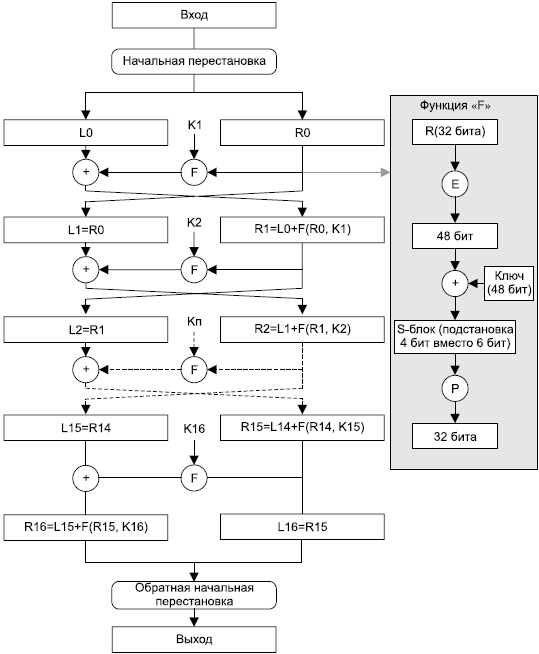
Начнём с трёх ключей:

* DES-EEE3: Исходный текст шифруется три раза, используя разные ключи.
* DES-EDE3: Исходный текст шифруется, затем дешифруется (уже другим ключом), затем снова шифруется (третьим ключом).
* DES-EEE2: Исходный текст шифруется три раза, однако ключи на первом и последнем шаге одинаковые.
* DES-EDE2: Исходный текст шифруется, затем дешифруется (другим ключом), затем снова шифруется (ключом, используемым при первом шифровании).

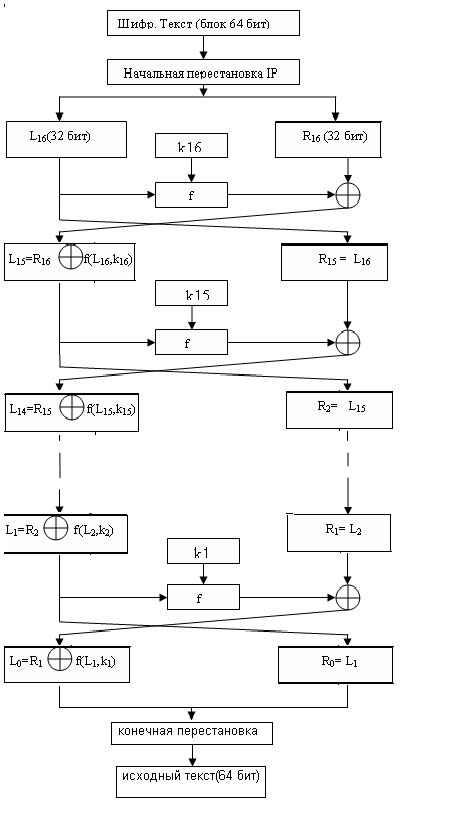
На практике, самый используемый тип 3DES шифрования - DES-EDE3.

# **3. Блок-схема алгоритма**

Общая схема алгоритма *кодирования* для DES:



Общая схема алгоритма *декодирования* для DES:



## **Стандарт шифрования ГОСТ 28147-89**

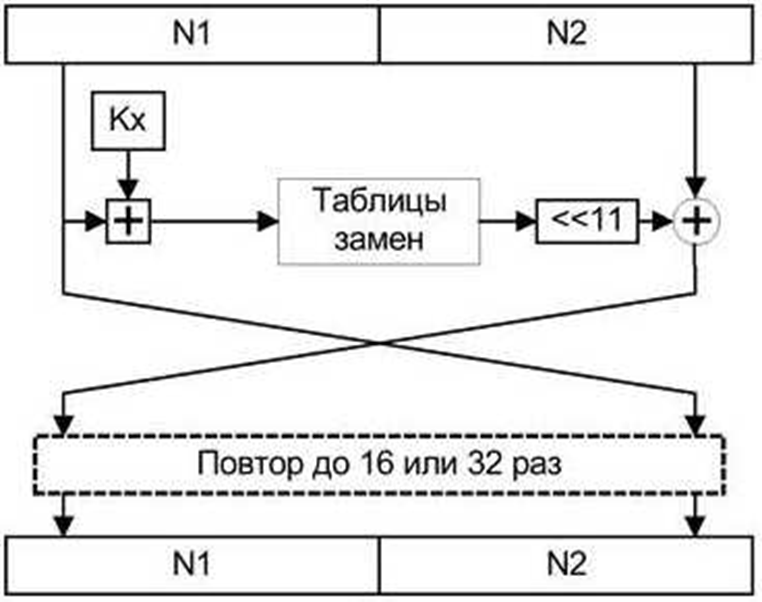
ГОСТ 28147-89 представляет собой симметричный 64-битовый блочный алгоритм с 256-битовым ключом.

Этот алгоритм криптографического преобразования данных предназначен для аппаратной и программной реализации, удовлетворяет криптографическим требованиям и не накладывает ограничений на степень секретности защищаемой информации.

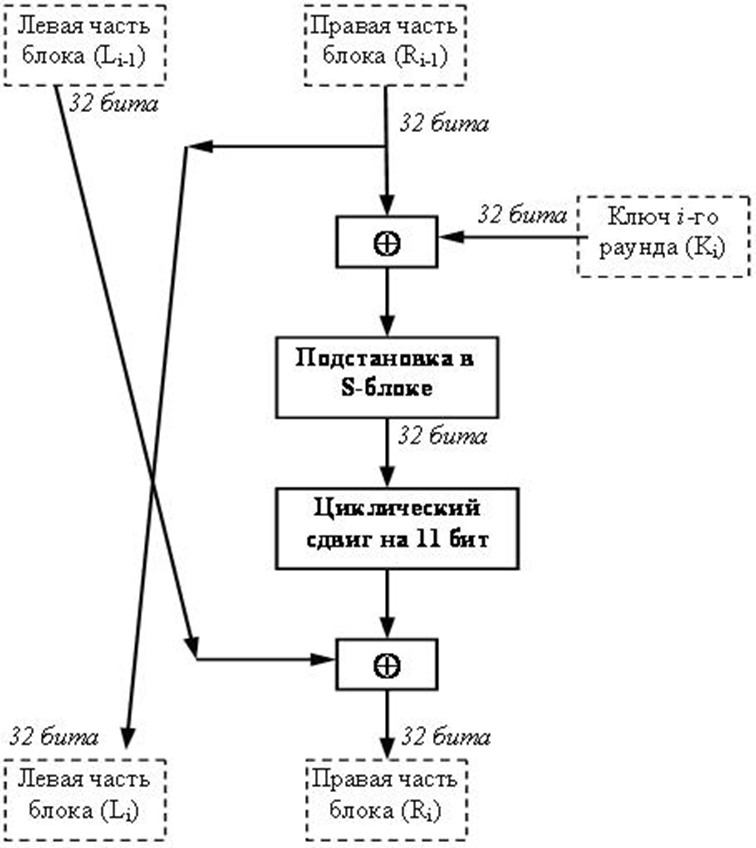
### **Схема алгоритма ГОСТ 28147-89**

Данные, подлежащие зашифровке, разбивают на 64-разрядные блоки.

Эти блоки разбиваются на два субблока N1 и N2 по 32 бит



Структура одного раунда ГОСТ 28147-89



### **Описание работы ГОСТ 28147-89**

Шифруемый блок данных разбивается на две части, которые затем обрабатываются как отдельные 32-битовые целые числа без знака.

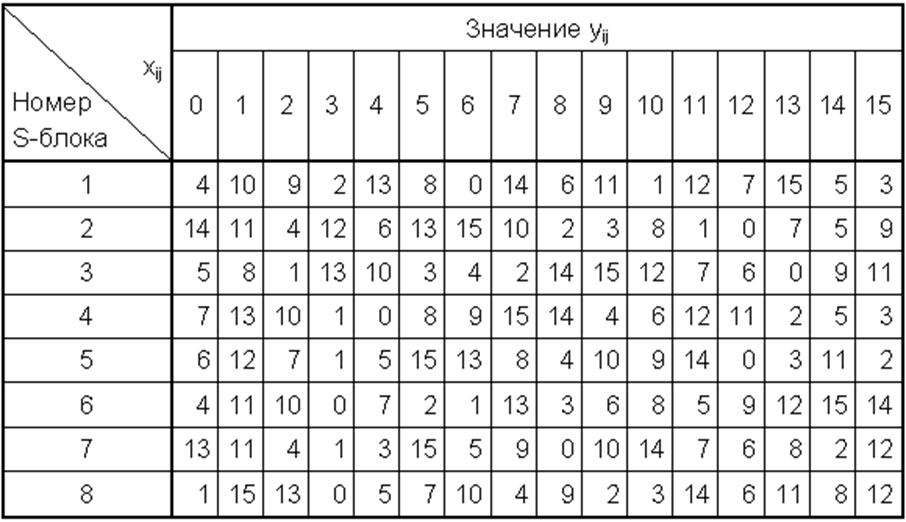
Сначала правая половина блока и подключ раунда складываются по модулю 232.

Затем производится поблочная подстановка.

32-битовое значение, полученное на предыдущем шаге (обозначим его S), интерпретируется как массив из восьми 4-битовых блоков кода: S=(S0,S1,S2,S3,S4,S5,S6,S7).

Далее значение каждого из восьми блоков заменяется на новое, которое выбирается по таблице замен

Таблица замен ГОСТ 28147-89



В каждой строке таблицы замен записаны числа от 0 до 15 в произвольном порядке без повторений.

Значения элементов таблицы замен взяты от 0 до 15, так как в четырех битах, которые подвергаются подстановке, может быть записано целое число без знака в диапазоне от 0 до 15.

Значение блока S1 (четыре младших бита 32-разрядного числа S) заменится на число, стоящее на позиции, номер которой равен значению заменяемого блока.

Например, в этом случае S1=0 заменится на 4, если S1=1, то оно заменится на 10 и т.д.

### **Описание ГОСТ 28147-89**

После выполнения подстановки все 4-битовые блоки снова объединяются в единое 32-битное слово, которое затем циклически сдвигается на 11 битов влево.

Наконец, с помощью побитовой операции "сумма по модулю 2" результат объединяется с левой половиной, вследствие чего получается новая правая половина Ri.

Новая левая часть Li берется равной младшей части преобразуемого блока: Li= Ri-1.

Полученное значение преобразуемого блока рассматривается как результат выполнения одного раунда алгоритма шифрования.

**Процедуры шифрования и расшифрования**

ГОСТ 28147-89 является блочным шифром, поэтому преобразование данных осуществляется блоками в так называемых базовых циклах.

Базовые циклы заключаются в многократном выполнении для блока данных основного раунда, рассмотренного нами ранее, с использованием разных элементов ключа и отличаются друг от друга порядком использования ключевых элементов.

В каждом раунде используется один из восьми возможных 32-разрядных подключей.

Рассмотрим процесс создания подключей раундов.  
В ГОСТ эта процедура очень проста, особенно по сравнению с DES.  
256-битный ключ K разбивается на восемь 32-битных подключей, обозначаемых K0, K1, K2, K3, K4, K5, K6, K7.   
Алгоритм включает 32 раунда, поэтому каждый подключ при шифровании используется в четырех раундах в последовательности, представленной в таблице

Таблица Последовательность использования подключей при шифровании

Раунд 1 2 3 4 5 6 7 8

Подключ K0 K1 K2 K3 K4 K5 K6 K7

Раунд 9 10 11 12 13 14 15 16

Подключ K0 K1 K2 K3 K4 K5 K6 K7

Раунд 17 18 19 20 21 22 23 24

Подключ K0 K1 K2 K3 K4 K5 K6 K7

Раунд 25 26 27 28 29 30 31 32

Подключ K7 K6 K5 K4 K3 K2 K1 K0

Процесс расшифрования производится по тому же алгоритму, что и шифрование. Единственное отличие заключается в порядке использования подключей Ki. При расшифровании подключи должны быть использованы в обратном порядке, а именно, как указано в таблице

Таблица Последовательность использования подключей при расшифровании

Раунд 1 2 3 4 5 6 7 8

Подключ K0 K1 K2 K3 K4 K5 K6 K7

Раунд 9 10 11 12 13 14 15 16

Подключ K7 K6 K5 K4 K3 K2 K1 K0

Раунд 17 18 19 20 21 22 23 24

Подключ K7 K6 K5 K4 K3 K2 K1 K0

Раунд 25 26 27 28 29 30 31 32

Подключ K7 K6 K5 K4 K3 K2 K1 K0

### **ГОСТ 28147-89 Режимы работы**

**Режим простой замены:** все блоки шифруются независимо друг от друга с разными подключами в разных раундах. Для одинаковых блоков сообщения М блоки шифртекста будут одинаковыми.

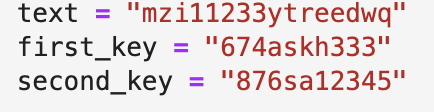
**Режим гаммирования:** В регистры N1 и N2 записывается 64-битовая синхропосылка (вектор инициализации) и шифруется с использованием СК. Результат подается на вход регистров и снова шифруется с использованием ключа. Получается «одноразовый блокнот».

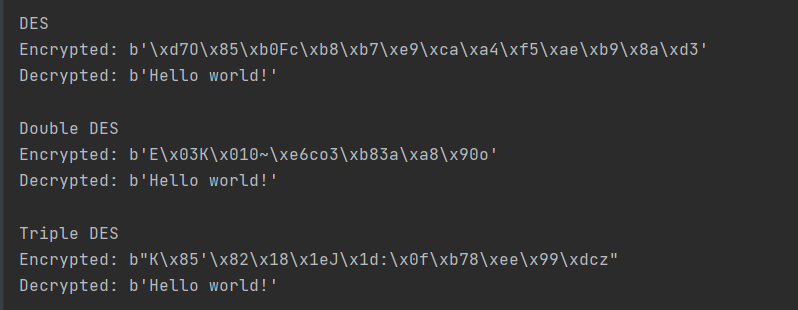
В **режиме гаммирования с обратной связью** для заполнения регистров N1 и N2, начиная со 2-го блока, используется результат зашифрования предыдущего блока открытого текста.

**Работа криптосистемы в режиме гаммирования**



# **4. Пример работы программы**





# **5. Код программы**

from des import Des, DoubleDes, TripleDes  
  
def main():  
 file = open("sample.txt", "r")  
 data = file.read().encode()  
 print("Data: %r\n" % data)  
  
 k = Des(b"DESCRYPT", pad=b"\1")  
 d = k.encrypt(data)  
 print("DES")  
 print("Encrypted: %r" % d)  
 print("Decrypted: %r\n" % k.decrypt(d))  
 assert k.decrypt(d) == data  
  
 k = DoubleDes(b"DESCRYPTREALREAL", pad=b"\1")  
 d = k.encrypt(data)  
 print("Double DES")  
 print("Encrypted: %r" % d)  
 print("Decrypted: %r\n" % k.decrypt(d))  
 assert k.decrypt(d) == data  
  
 k = TripleDes(b"DESCRYPTREALREALHOOLHOOL", pad=b"\1")  
 d = k.encrypt(data)  
 print("Triple DES")  
 print("Encrypted: %r" % d)  
 print("Decrypted: %r\n" % k.decrypt(d))  
 assert k.decrypt(d) == data  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# **6. Вывод**

Сам по себе алгоритм DES уже не является криптостойким, т.к. силами современной̆ вычислительной̆ техники его вполне можно взломать. С попыткой̆ увеличения криптостойкости, используя двойной DES, была выявлена слабость, называемая “встреча посередине”, что тоже делает алгоритм уязвимым. Что касается тройного DES, то на данный момент его можно считать криптостойким. Для успешной атаки на 3DES потребуется около 232 бит известного открытого текста, 2113 шагов, 290 циклов DES-шифрования и 288 бит памяти. На данный момент это непрактично, и, по оценкам НИСТ, алгоритм с выбором трех различных ключей должен остаться надежным до 2030-х.