

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИТМО»

Информационная безопасность

Отчет по лабораторной работе №4

«Демонстрационная версия криптостойкого блочного алгоритма
Rijndael»

Выполнил:

Студент группы №Р34101

Патутин В.М.

Преподаватель:

Маркина Т. А.

Санкт-Петербург

2022

1. Цель работы

Ознакомление с принципами шифрования, используемыми в алгоритме симметричного шифрования **AES RIJNDAEL**

2. Вариант задания

Задание по варианту 22

Сравните эквивалентность прямого и обратного преобразований в алгоритмах Rijndael и ГОСТ 28147-89.

3. Листинг программы с комментариями

а. Входные данные для работы алгоритма

В качестве текста для шифрования я взял строку вида:

[InformationSecurityInAction](#)

И её представление в 16-ричном виде согласно таблице ASCII

49 6E 66 6F 72 6D 61 74 69 6F 6E 53 65 63 75 72 69 74 79 49 6E 41
63 74 69 6F 6E

Ограничиваем строку до 16 символов, чтобы можно было воспользоваться программой:

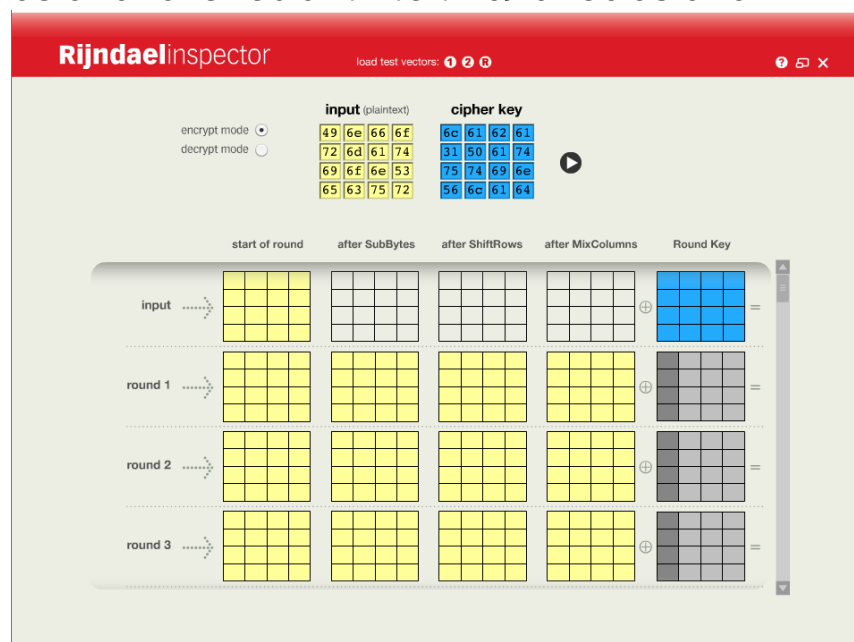
49 6E 66 6F 72 6D 61 74 69 6F 6E 53 65 63 75 72

Также нам необходим ключ, который будет в текстовом формате иметь вид: [laba1PatutinVladimir2022](#)

6C 61 62 61 31 50 61 74 75 74 69 6E 56 6C 61 64 69 6D 69 72 32 30
32 32

И её представление с учетом ограничений будет:

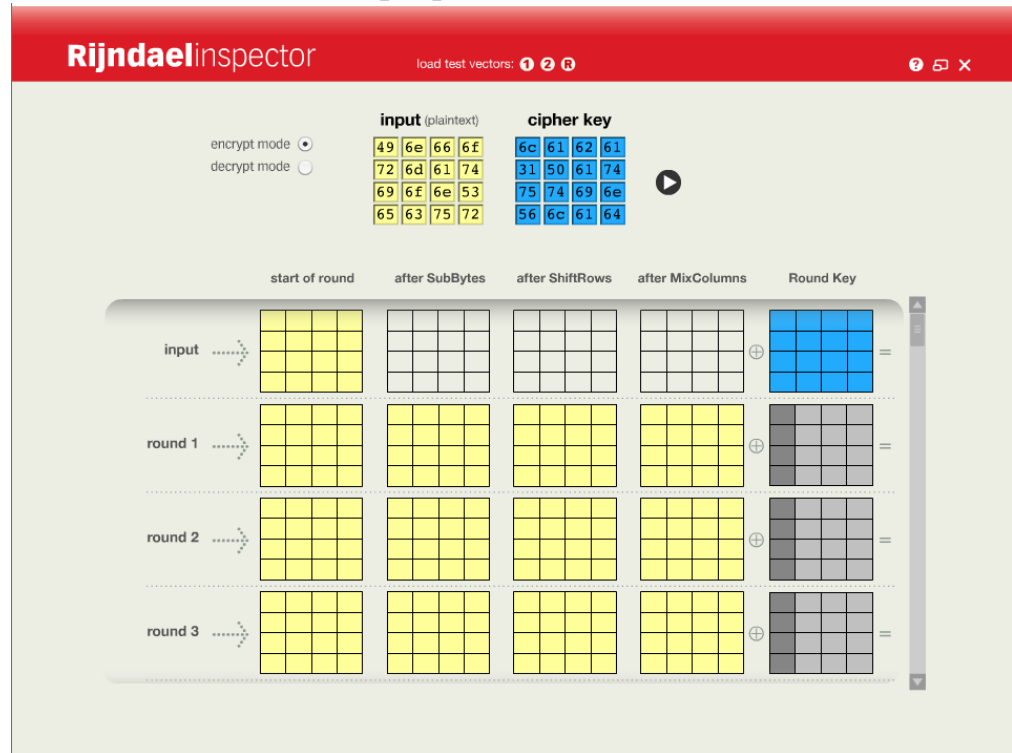
6C 61 62 61 31 50 61 74 75 74 69 6E 56 6C 61 64



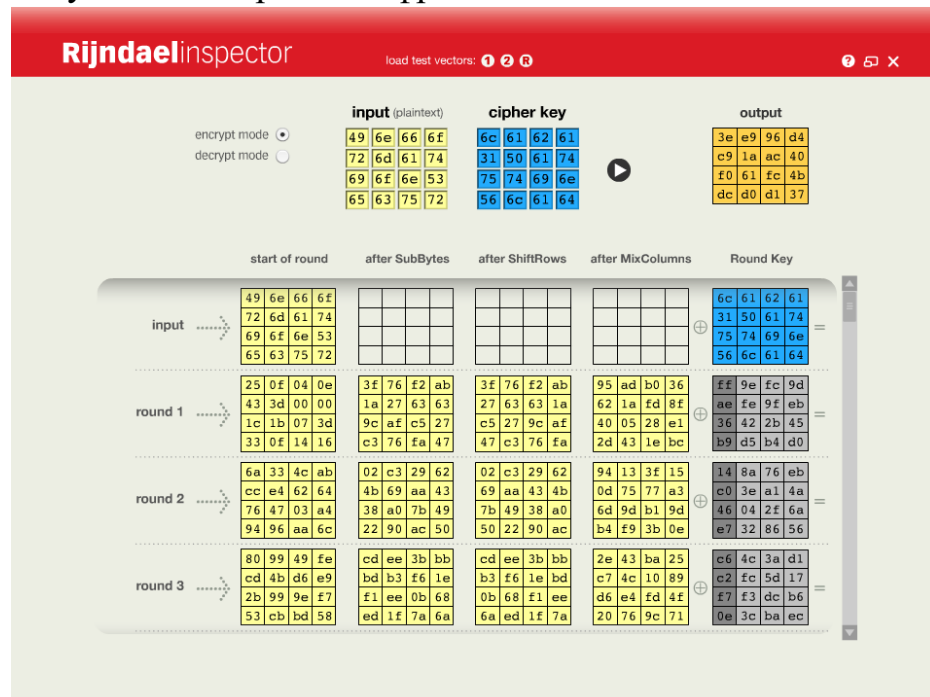
в. Пошаговое выполнение алгоритма шифрования

Загоним данные в программу Rijndael Inspector

Исходное положение программы:

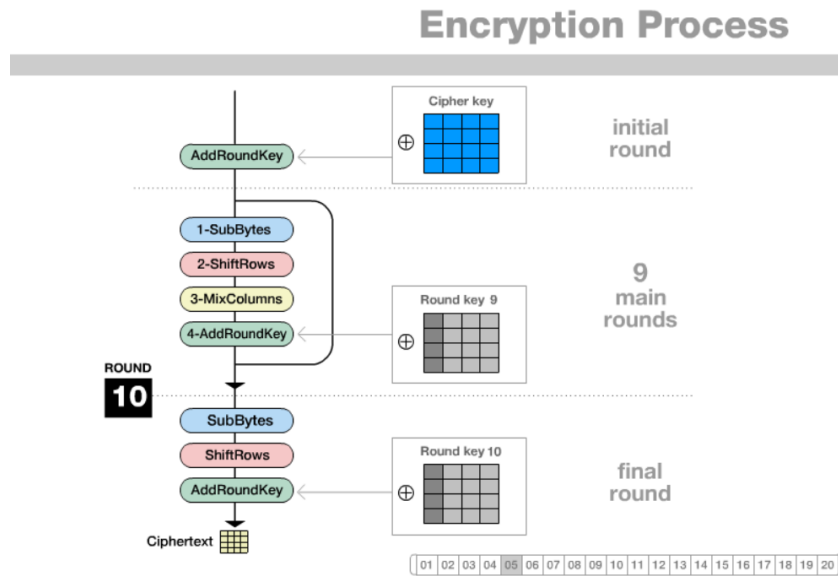


Запускаем алгоритм шифрования



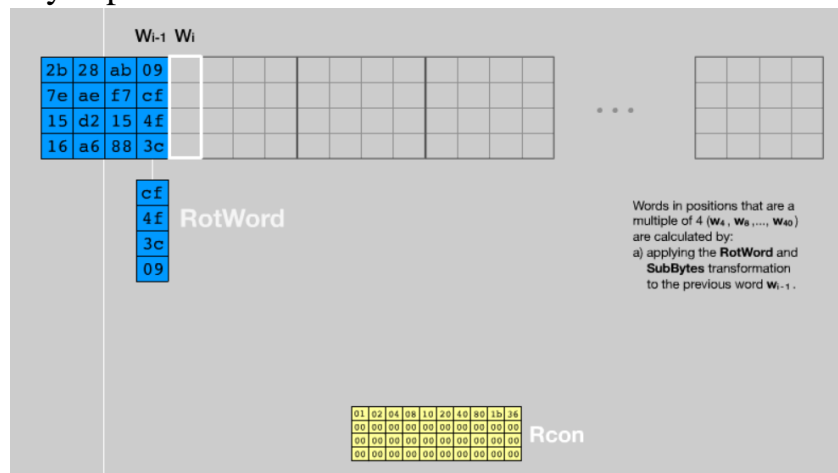
На примере входного значения и первого раунда разберем работу алгоритма.

Общий процесс:



Первый этап сразу после начала работы программы **addRoundKey** – на этом этапе производится матричный XOR с **roundKey**, **roundKey** вычисляется путем применения процедуры **keyExpansion** к ключу шифрования

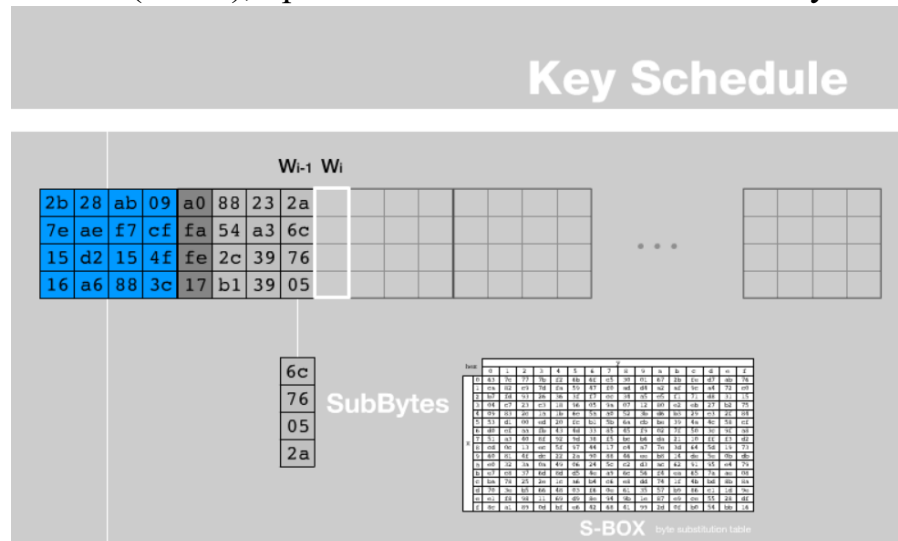
keyExpansion



Алгоритм берет w_{i-1} слово делает **RotWord** и **SubBytes** и производит XOR с w_{i-4} (разница в индексах зависит от размера матрицы) и XOR со столбцом **Rcon** (константные значения на каждый раунд, по столбцу на каждый раунд)

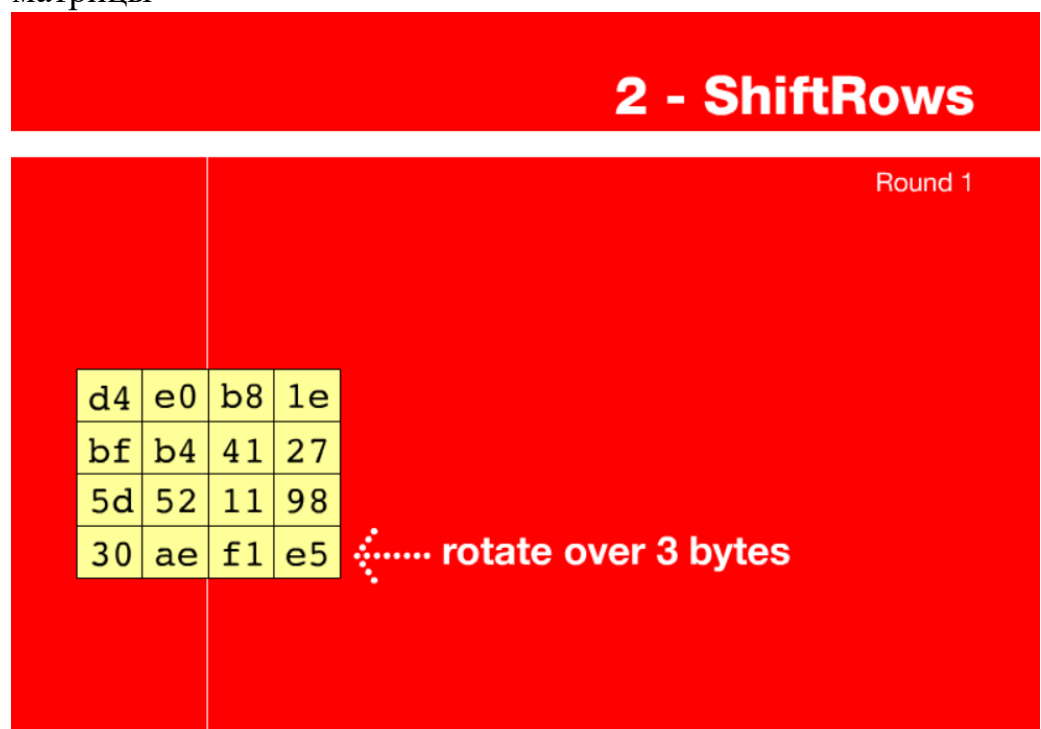
RotWord - функция, использующаяся в процедуре **Key Expansion**, которая берёт 4-байтовое слово и производит над ним циклическую перестановку

SubBytes - трансформации при шифровании, которые обрабатывают State, используя нелинейную таблицу замещения байтов (S-box), применяя её независимо к каждому байту State



После этого начинается несколько раундов, каждый из которых обозначен четырьмя операциями, несколько из них мы разобрали выше

ShiftRows – поэлементный сдвиг влево на +1 каждой строки матрицы



MixColumns - трансформация при шифровании, которая берёт

все столбцы State и смешивает их данные (независимо друг от друга), чтобы получить новые столбцы

с. Пошаговое выполнение дешифрования

Основные операции выполняются по инвертированному механизму от основных операций

The RijndaelInspector application interface shows the decryption process. The top section displays the input (ciphertext), cipher key, and output. The bottom section shows the state of the Rijndael state after three inverse rounds (inv. round 1, 2, 3).

Initial State (Top Screenshot):

- Input (ciphertext):

3e	e9	96	d4
c9	1a	ac	40
f0	61	fc	4b
dc	d0	d1	37
- Cipher key:

6c	61	62	61
31	50	61	74
75	74	69	6e
56	6c	61	64
- Output: (empty 4x4 grid)

State after three inverse rounds (Bottom Screenshot):

- Input: (same as above)
- inv. round 1: (empty 4x4 grid)
- inv. round 2: (empty 4x4 grid)
- inv. round 3: (empty 4x4 grid)

The bottom section shows the state of the Rijndael state after three inverse rounds (inv. round 1, 2, 3). The state is shown as a 4x4 grid of bytes, with the input, cipher key, and output. The state after each round is shown, including the Round Key and the result of the AddRoundKey operation.

State after inv. round 1:

- Input: (same as above)
- inv. round 1: (empty 4x4 grid)
- inv. round 2: (empty 4x4 grid)
- inv. round 3: (empty 4x4 grid)

State after inv. round 2:

- Input: (same as above)
- inv. round 1: (empty 4x4 grid)
- inv. round 2: (empty 4x4 grid)
- inv. round 3: (empty 4x4 grid)

State after inv. round 3:

- Input: (same as above)
- inv. round 1: (empty 4x4 grid)
- inv. round 2: (empty 4x4 grid)
- inv. round 3: (empty 4x4 grid)

На каждом раунде выполняются операции:

InvShiftRows – обратная к операции shift rows, следовательно поэлементный сдвиг вправо

InvSubBytes – обратная операция к subBytes, дешифрование через таблицу замещения

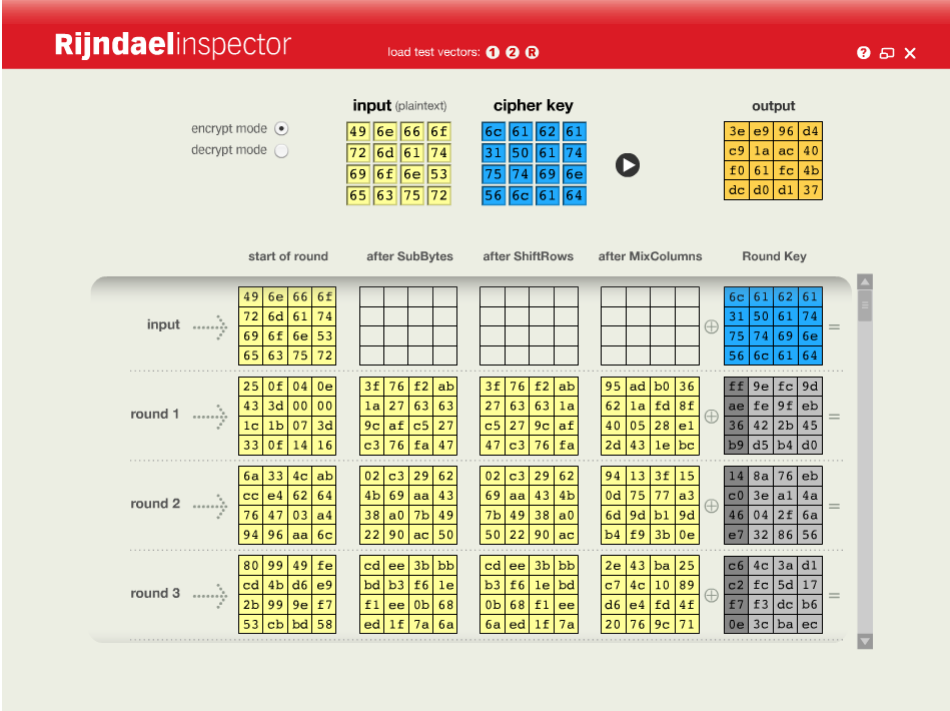
AddRoundKey – описанная в шифровании, не изменяется

InvMixColumns – обратная операция к mixColumns

Перед и после начала раундов выполняется AddRoundKey, чтобы сохранить последовательность, после раундов выполняются по одной операции InvShiftRows, InvSubBytes, AddRoundKey

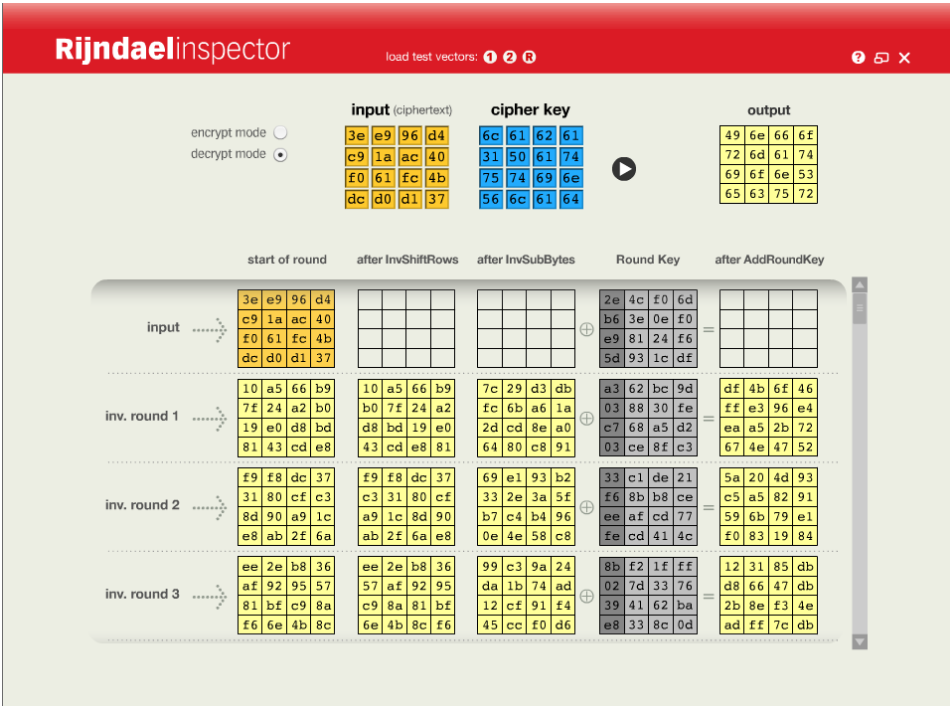
4. Результат работы программы

а. Шифрование



The screenshot shows the RijndaelInspector application in encryption mode. The input (plaintext) is a 4x4 matrix of hex values: 49 6e 66 6f, 72 6d 61 74, 69 6f 6e 53, 65 63 75 72. The cipher key is a 4x4 matrix: 6c 61 62 61, 31 50 61 74, 75 74 69 6e, 56 6c 61 64. The output is a 4x4 matrix: 3e e9 96 d4, c9 1a ac 40, f0 61 fc 4b, dc d0 d1 37. Below the main interface, a detailed view of the encryption rounds is shown, including the start of round, after SubBytes, after ShiftRows, after MixColumns, and the Round Key. The rounds are labeled round 1, round 2, and round 3.

Дешифрование



The screenshot shows the RijndaelInspector application in decryption mode. The input (ciphertext) is a 4x4 matrix of hex values: 3e e9 96 d4, c9 1a ac 40, f0 61 fc 4b, dc d0 d1 37. The cipher key is the same as in the encryption mode. The output is the original plaintext: 49 6e 66 6f, 72 6d 61 74, 69 6f 6e 53, 65 63 75 72. Below the main interface, a detailed view of the decryption rounds is shown, including the start of round, after InvShiftRows, after InvSubBytes, the Round Key, and after AddRoundKey. The rounds are labeled inv. round 1, inv. round 2, and inv. round 3.

b.

5. Ответ на контрольные вопросы

Вариант 22

Сравните эквивалентность прямого и обратного преобразований в алгоритмах Rijndael и ГОСТ 28147-89.

Гост:

В алгоритме ГОСТ28147-89 эквивалентность структуры прямого и обратного криптографического преобразования не обеспечивается специально, а является простым следствием использованного архитектурного решения. В любой однородной сбалансированной сети Файстеля оба эти преобразования алгоритмически идентичны и различаются только порядком использования ключевых элементов: при расшифровании элементы используются в порядке, обратном тому, в котором они используются при зашифровании.

Rijndael:

Шифр Rijndael построен на базе прямых преобразований. Как и для всех подобных алгоритмов, обратное преобразование строится из обращений шагов прямого преобразования, применяемых в обратном порядке. В силу сказанного обеспечить такую же степень идентичности прямого и обратного преобразования, которая достигается в сетях Файстеля, не представляется возможным. Однако специальными конструкторскими решениями достигается близкая степень соответствия: прямое и обратное преобразование получаются идентичными с точностью до используемых в преобразованиях констант.