

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
Факультет радиофизики и компьютерных технологий

Лабораторная работа 4  
Алгоритм криптографического преобразования данных  
ГОСТ28147-89

Минск 2021

## Оглавление

Теоретический материал .....	3
Глава 1. блочные алгоритмы.....	3
Глава 2. цикл фейстеля .....	7
2.1 Криптоанализ .....	8
2.2 Дифференциальный и линейный криптоанализ .....	9
2.3 Используемые критерии при разработке алгоритмов.....	10
глава 3. Описание алгоритма шифрования гост28147-89 .....	12
3.1 Логика построения шифра и структура ключевой информации госта ...	12
3.2 Основной шаг криптопреобразования .....	13
3.3 Базовые циклы криптографических преобразований .....	14
глава 4. Пошаговая визуализация всех режимов шифрования гост28147-89.....	16
4.1 Основные режимы шифрования .....	16
4.2 Простая замена.....	16
4.3 Гаммирование .....	17
4.4 Гаммирование с обратной связью.....	21
4.5 Режим выработки имитовставки.....	22
глава 5. Шифрование по алгоритму гост 28147-89. пошаговый режим.....	23
5.1 Основные режимы шифрования .....	23
5.2 Основные функции.....	23
5.3 Ввод входных данных .....	24
5.4 Работа в пошаговом режиме.....	24
5.5 Запись данных в файл .....	28
5.6 Очистка форм .....	28
Порядок выполнения работы .....	29
Содержание отчета.....	30
Контрольные вопросы .....	31
Литература .....	32

**Цель работы** – изучить способы реализации алгоритма криптографического преобразования по алгоритму ГОСТ28147-89, научиться использовать возможности АПК «КриптоЛаб» при криптографическом преобразовании данных по алгоритму ГОСТ28147-89.

## ТЕОРЕТИЧЕСКИЙ МАТЕРИАЛ

### Глава 1. БЛОЧНЫЕ АЛГОРИТМЫ

Блочными называются шифры, в которых логической единицей шифрования является некоторый блок открытого текста, после преобразований которого получается блок шифрованного текста такой же длины. Обычно используются блоки размером 64 бита. Большинство сетевых приложений, в которых применяется схема традиционного шифрования, использует блочные шифры.

Блочный шифр предполагает преобразование  $n$ -битового блока открытого текста в блок шифрованного текста такого же размера. Число различных блоков при этом равно  $2^n$ , и чтобы шифрование было обратимым (т. е. чтобы обеспечивалась возможность дешифрования), каждый из таких блоков должен преобразовываться в свой уникальный блок шифрованного текста. Такие преобразования называются обратимыми, или несингулярными. Примеры несингулярного и сингулярного преобразований для  $n = 2$  представлены на Рисунке 1.

<i>Открытый</i>	<i>Шифрованный</i>	<i>Открытый</i>	<i>Шифрованный</i>
<i>текст</i>	<i>текст</i>	<i>текст</i>	<i>текст</i>
00	11	00	11
01	10	01	10
10	00	10	01
11	01	11	01

Рисунок 1. Примеры несингулярного и сингулярного преобразований для  $n = 2$

В данном примере в случае необратимого отображения зашифрованный текст соответствует двум разным блокам открытого текста. Если ограничиться рассмотрением только обратимых отображений, число различных допустимых преобразований окажется равным  $2^n$ .

На рис.2 показана общая схема подстановочного шифра для  $n = 4$ . Поступающее на вход 4-битовое значение определяет одно из 16 возможных начальных состояний, которое отображается подстановочным шифром в одно из 16 конечных состояний, каждое из которых представляется 4-битовым отображением зашифрованного текста. Схемы шифрования и дешифрования можно представить в виде таблиц (Рисунок 3). Это самая общая схема процесса блочного шифрования, которая может служить для разработки любого обратимого соответствия между открытым и зашифрованным текстом.

Однако при практической реализации данного подхода возникает следующая проблема. Если использовать блок небольшого размера, например,  $n = 4$ , система оказывается эквивалентной классическому подстановочному шифру. Как уже было отмечено ранее, такие системы уязвимы и могут быть раскрыты с помощью статистического анализа открытого текста. Этот недостаток порождается не самой природой подстановочного шифра, а использованием коротких блоков. При достаточно больших значениях  $n$  и при отсутствии каких-либо ограничений на обратимую подстановку, задающую преобразование открытого текста в зашифрованный, статистические характеристики исходного открытого текста маскируются настолько хорошо, что такого рода криптоанализ становится практически бесполезным.

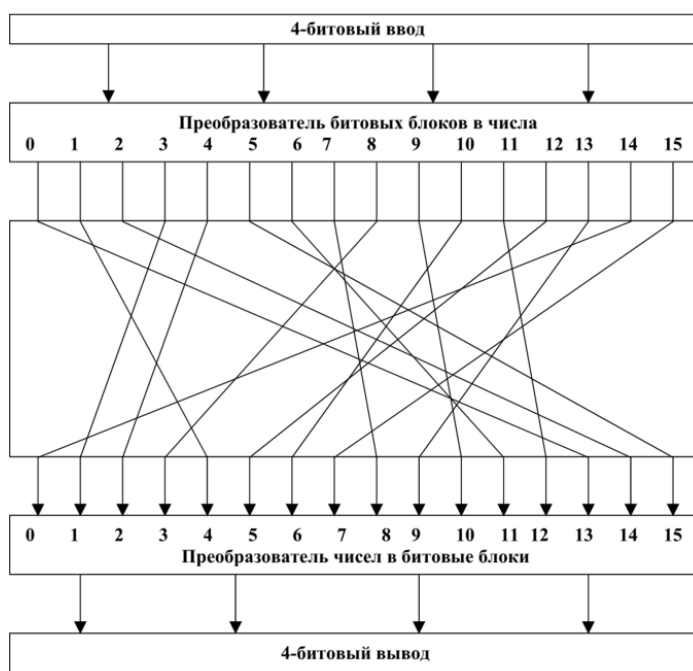


Рисунок 2 Общая схема подстановочного шифра для  $n = 4$

В то же время предположение о допустимости любых обратимых подстановок при больших размерах блоков оказывается весьма непрактичным в отношении реализации алгоритма и скорости выполнения соответствующего приложения. Для такого преобразования ключом является само отображение.

Открытый текст	Шифрованный текст	Открытый текст	Шифрованный текст
0000	1110	0000	1110
0001	0100	0001	001
0010	1101	0010	0100
001	0001	001	1000
0100	0010	0100	0001
0101	1111	0101	1100
010	1011	010	1010
0111	1000	0111	1111
1000	001	1000	0111
1001	1010	1001	1101
1010	010	1010	1001
1011	1100	1011	010
1100	0101	1100	1011
1101	1001	1101	0010
1110	0000	1110	0000
1111	0111	1111	0101

Рисунок 3 Таблица шифрования и дешифрования для блочного шифра

Обратимся снова к таблице на рисунке 3, которая определяет конкретное обратимое отображение пространства открытых текстов в пространство шифрованных для  $n = 4$ . Отображение можно задать элементами из второго столбца таблицы, в котором приведены значения шифрованного текста для соответствующих значений открытого текста. Это, по сути, и есть ключ, который отличает данное конкретное отображение от всех других допустимых отображений. В рассматриваемом случае длина ключа оказывается равной 64. Обычно для блочного шифра с  $n$ -битовым подстановочным блоком размер ключа равен  $n \times 2$ .

Занимаясь изучением данной проблемы, Файстель предложил для больших  $n$  аппроксимировать такую идеальную систему блочного шифрования набором простых для реализации компонентов. Но, прежде чем приступить к непосредственному обсуждению предложенного Файстелем подхода, необходимо обратить внимание на следующий момент. Невозможно позволить себе работать с блочным подстановочным шифром общего вида, но для реализации соответствующего алгоритма программным методом можно вместо всех  $2^n$  допустимых отображений рассмотреть некоторое более узкое подмножество. Например, предположим, что отображения задаются системой линейных уравнений. Для  $n = 4$  получим:

$$\begin{aligned} Y_1 &= K_{11}X_1 + K_{12}X_2 + K_{13}X_3 + K_{14}X_4 \\ Y_2 &= K_{21}X_1 + K_{22}X_2 + K_{23}X_3 + K_{24}X_4 \\ Y_3 &= K_{31}X_1 + K_{32}X_2 + K_{33}X_3 + K_{34}X_4 \\ Y_4 &= K_{41}X_1 + K_{42}X_2 + K_{43}X_3 + K_{44}X_4 \end{aligned}$$

где  $X_i$  обозначает четыре двоичные цифры блока открытого текста,  $Y_i$  — четыре двоичные цифры блока шифрованного текста,  $K_{ij}$  — двоичные коэффициенты, а все арифметические операции выполняются по модулю 2.

Длина ключа равна  $n^2$ , и в данном случае это всего 16 битов. Опасность задания отображений формулами заключается в том, что в случае, когда структура такого алгоритма оказывается доступной криптоаналитику, система может стать весьма уязвимой. В данном примере система шифрования, по сути, эквивалентна шифру Хилла, но в применении не к текстовым символам, а к двоичным данным. Подобные простые линейные системы защищены весьма слабо.

## Глава 2. ЦИКЛ ФЕЙСТЕЛЯ

Блочный алгоритм преобразовывает  $n$ -битный блок незашифрованного текста в  $n$ -битный блок зашифрованного текста. Число блоков длины  $n$  равно  $2^n$ . Для того чтобы преобразование было обратимым, каждый из таких блоков должен преобразовываться в свой уникальный блок зашифрованного текста. При маленькой длине блока такая подстановка плохо скрывает статистические особенности незашифрованного текста. Если блок имеет длину 64 бита, то он уже хорошо скрывает статистические особенности исходного текста. Но в данном случае преобразование текста не может быть произвольным в силу того, что ключом будет являться само преобразование, что исключает эффективную как программную, так и аппаратную реализации.

Наиболее широкое распространение получили сети Фейстеля, так как, с одной стороны, они удовлетворяют всем требованиям к алгоритмам симметричного шифрования, а с другой стороны, достаточно просты и компактны.

Сеть Фейстеля имеет следующую структуру. Входной блок делится на несколько равной длины подблоков, называемых ветвями. В случае, если блок имеет длину 64 бита, используются две ветви по 32 бита каждая. Каждая ветвь обрабатывается независимо от другой, после чего осуществляется циклический сдвиг всех ветвей влево. Такое преобразование выполняется несколько циклов или раундов. В случае двух ветвей каждый раунд имеет структуру, показанную на рисунке:

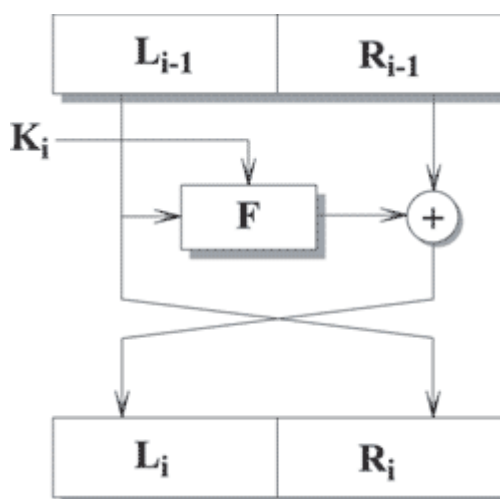


Рисунок 4.  $i$ -ый раунд сети Фейстеля

Функция  $F$  называется образующей. Каждый раунд состоит из вычисления функции  $F$  для одной ветви и побитового выполнения операции  $XOR$  результата  $F$  с другой ветвью. После этого ветви меняются местами. Считается, что оптимальное число раундов - от 8 до 32. Важно то, что увеличение количества раундов значительно увеличивает криптостойкость алгоритма. Возможно, эта особенность и повлияла на столь активное распространение сети Фейстеля, так как

для большей криптостойкости достаточно просто увеличить количество *раундов*, не изменяя сам *алгоритм*. В последнее время количество *раундов* не фиксируется, а лишь указываются допустимые пределы.

*Сеть Фейстеля* является обратимой даже в том случае, если *функция F* не является таковой, так как для *дешифрования* не требуется вычислять  $F^{-1}$ . Для *дешифрования* используется тот же *алгоритм*, но на вход подается зашифрованный текст, и ключи используются в обратном порядке. В настоящее время все чаще используются различные разновидности *сети Фейстеля* для 128-битного блока с четырьмя ветвями. Увеличение количества ветвей, а не размерности каждой ветви связано с тем, что наиболее популярными до сих пор остаются процессоры с 32-разрядными словами, следовательно, оперировать 32-разрядными словами эффективнее, чем с 64-разрядными.

Основной характеристикой алгоритма, построенного на основе *сети Фейстеля*, является *функция F*. Различные варианты касаются также начального и конечного преобразований. Подобные преобразования, называемые *забеливанием* (*whitening*), осуществляются для того, чтобы выполнить начальную рандомизацию входного текста.

## 2.1 Криптоанализ

Процесс, при котором предпринимается попытка узнать  $X$ ,  $K$  или и то, и другое, называется *криптоанализом*. Одной из возможных атак на *алгоритм шифрования* является *лобовая атака*, т. е. простой перебор всех возможных *ключей*. Если множество *ключей* достаточно большое, то подобрать *ключ* нереально. При *длине ключа n бит* количество возможных *ключей* равно  $2^n$ . Таким образом, чем *длиннее ключ*, тем более стойким считается *алгоритм* для лобовой атаки.

Существуют различные типы атак, основанные на том, что противнику известно определенное количество пар незашифрованное сообщение - зашифрованное сообщение. При анализе зашифрованного текста противник часто применяет *статистические методы* анализа текста. При этом он может иметь общее *представление* о типе текста, например, английский или русский текст, выполнимый *файл* конкретной ОС, исходный текст на некотором конкретном языке программирования и т. д. Во многих случаях криптоаналитик имеет достаточно много информации об исходном тексте. Криптоаналитик может иметь возможность перехвата одного или нескольких незашифрованных сообщений вместе с их зашифрованным видом. Или криптоаналитик может знать основной формат или основные характеристики сообщения. Говорят, что криптографическая схема абсолютно безопасна, если зашифрованное сообщение не содержит никакой информации об исходном сообщении. Говорят, что криптографическая схема вычислительно безопасна, если:

1. Цена расшифровки сообщения больше цены самого сообщения.



2. Время, необходимое для расшифровки сообщения, больше срока жизни сообщения.

## 2.2 Дифференциальный и линейный криптоанализ

Рассмотрим в общих чертах основной подход, используемый при *дифференциальном* и *линейном криптоанализе*. И в том, и в другом случае предполагается, что известно достаточно большое количество пар (незашифрованный текст, зашифрованный текст).

Понятие *дифференциального криптоанализа* было введено Эли Бихамом (Biham) и Ади Шамиром (Shamir) в 1990 году. Конечная задача дифференциального криптоанализа - используя свойства алгоритма, в основном свойства S-box, определить подключ раунда. Конкретный способ *дифференциального криптоанализа* зависит от рассматриваемого алгоритма шифрования.

Если в основе алгоритма лежит *сеть Фейстеля*, то можно считать, что блок  $m$  состоит из двух половин —  $m_0$  и  $m_1$ . *Дифференциальный криптоанализ* рассматривает отличия, которые происходят в каждой половине при шифровании. (Для алгоритма *DES* "отличия" определяются с помощью операции *XOR*, для других алгоритмов возможен иной способ). Выбирается пара незашифрованных текстов с фиксированным отличием. Затем анализируются отличия, получившиеся после шифрования одним *раундом* алгоритма, и определяются вероятности различных ключей. Если для многих пар входных значений, имеющих одно и то же отличие  $X$ , при использовании одного и того же *подключа* одинаковыми ( $Y$ ) оказываются и отличия соответствующих выходных значений, то можно говорить, что  $X$  влечет  $Y$  с определенной вероятностью. Если эта вероятность близка к единице, то можно считать, что *подключ раунда* найден с данной вероятностью. Так как *раунды* алгоритма независимы, вероятности определения *подключа* каждого *раунда* следует перемножать. Как мы помним, считается, что результат *шифрования* данной пары известен. Результаты *дифференциального криптоанализа* используются как при разработке конкретных *S-box*, так и при определении оптимального числа *раундов*.

Другим способом *криптоанализа* является линейный криптоанализ, который использует линейные приближения преобразований, выполняемых алгоритмом шифрования. Данный метод позволяет найти ключ, имея достаточно большое число пар (незашифрованный текст, зашифрованный текст). Рассмотрим основные принципы, на которых базируется *линейный криптоанализ*. Обозначим

$P[1], \dots, P[n]$  - незашифрованный блок сообщения.

$C[1], \dots, C[n]$  - зашифрованный блок сообщения.

$K[1], \dots, K[m]$  - ключ.

$$A[i, j, \dots, k] = A[i] \oplus A[j] \oplus \dots \oplus A[k]$$

Целью *линейного криптоанализа* является поиск *линейного уравнения* вида

$$P[\alpha_1, \alpha_2, \dots, \alpha_a] \oplus C[\beta_1, \beta_2, \dots, \beta_b] = K[\gamma_1, \dots, \gamma_c]$$

Выполняющееся с вероятностью  $p \neq 0.5$ .  $\alpha_i$ ,  $\beta_i$  и  $\gamma_i$  - фиксированные позиции в блоках сообщения и ключе. Чем больше  $p$  отклоняется от 0.5, тем более подходящим считается уравнение.

Это уравнение означает, что, если выполнить операцию *XOR* над некоторыми битами незашифрованного сообщения и над некоторыми битами зашифрованного сообщения, получится бит, представляющий собой *XOR* некоторых бит ключа. Это называется линейным приближением, которое может быть верным с вероятностью  $p$ .

Уравнения составляются следующим образом. Вычисляются значения левой части для большого числа пар соответствующих фрагментов незашифрованного и зашифрованного блоков. Если результат оказывается равен нулю более чем в половине случаев, то полагают, что  $K[\gamma_1, \dots, \gamma_c] = 0$ . Если в большинстве случаев получается 1, полагают, что  $K[\gamma_1, \dots, \gamma_c] = 1$ . Таким образом получают систему уравнений, решением которой является ключ.

Как и в случае *дифференциального криптоанализа*, результаты *линейного криптоанализа* должны учитываться при разработке алгоритмов *симметричного шифрования*.

### 2.3 Используемые критерии при разработке алгоритмов

Принимая во внимание перечисленные требования, обычно считается, что *алгоритм симметричного шифрования* должен:

- Манипулировать данными в больших блоках, предпочтительно размером 16 или 32 бита.
- Иметь размер блока 64 или 128 бит.
- Иметь масштабируемый ключ до 256 бит.
- Использовать простые операции, которые эффективны на *микропроцессорах*, т. е. *исключающее или*, сложение, табличные подстановки, умножение по модулю. Не должно использоваться сдвигов переменной длины, побитных перестановок или *условных переходов*.
- Должна быть возможность реализации алгоритма на 8-битном процессоре с минимальными требованиями к памяти.
- Использовать заранее вычисленные *подключи*. На системах с большим количеством памяти эти *подключи* могут быть заранее вычислены для ускорения работы. В случае невозможности заблаговременного вычисления *подключей* должно произойти только замедление выполнения. Всегда должна быть возможность *шифрования данных* без каких-либо предварительных вычислений.
- Состоять из переменного числа итераций. Для приложений с маленькой *длиной ключа* нецелесообразно применять большое число итераций для противостояния дифференциальным и другим атакам. Следовательно,

должна быть возможность уменьшить число итераций без потери безопасности (не более чем уменьшенный размер ключа).

- По возможности не иметь слабых ключей. Если это невозможно, то количество слабых ключей должно быть минимальным, чтобы уменьшить вероятность случайного выбора одного из них. Тем не менее, все слабые ключи должны быть заранее известны, чтобы их можно было отбраковать в процессе создания ключа.
- Задействовать *подключи*, которые являются односторонним хэшем ключа. Это дает возможность использовать большие парольные фразы в качестве ключа без ущерба для безопасности.
- Не иметь линейных структур, которые уменьшают комплексность и не обеспечивают исчерпывающий поиск.
- Использовать простую для понимания разработку. Это дает возможность анализа и уменьшает закрытость алгоритма.
- 

Большинство блочных алгоритмов основано на использовании *сети Фейстеля*, все имеют плоское *пространство* ключей, с возможным исключением нескольких слабых ключей.

## Глава 3. ОПИСАНИЕ АЛГОРИТМА ШИФРОВАНИЯ ГОСТ28147-89

### 3.1 Логика построения шифра и структура ключевой информации гостя

Элементы данных обозначаются  $X$ .  $|X|$  - размер элемента данных  $X$  в битах. Если интерпретировать элемент данных  $X$  как целое неотрицательное число, то верно следующее неравенство  $0 \leq X < 2^{|X|}$ . Если элемент данных состоит из нескольких элементов меньшего размера, то обозначается следующим образом:  $X = (X_0, X_1, \dots, X_{n-1}) = X_0 \parallel X_1 \parallel \dots \parallel X_{n-1}$ . Процедура объединения нескольких элементов данных в один называется **конкатенацией** данных и обозначается символом « $\parallel$ ». Для размеров элементов данных должно выполняться следующее соотношение:  $|X| = |X_0| + |X_1| + \dots + |X_{n-1}|$ . Если интерпретировать составной элемент и все входящие в него элементы данных как целые числа без знака, то можно записать следующее равенство:

$$(X_0, X_1, \dots, X_{n-1}) = X_0 \parallel X_1 \parallel \dots \parallel X_{n-1} = X_0 + 2^{|X_0|} (X_1 + 2^{|X_1|} (\dots (X_{n-2} + 2^{|X_{n-2}|} X_{n-1}) \dots)).$$

В алгоритме элемент данных может интерпретироваться как массив отдельных:

$$X = (x_0, x_1, \dots, x_{n-1}) = x_0 + 2^1 \cdot x_1 + \dots + 2^{n-1} \cdot x_{n-1}.$$

Если над элементами данных выполняется некоторая операция, имеющая логический смысл, то предполагается, что данная операция выполняется над соответствующими битами элементов.

$A \bullet B = (a_0 \bullet b_0, a_1 \bullet b_1, \dots, a_{n-1} \bullet b_{n-1})$ , где  $n = |A| = |B|$ , символом « $\bullet$ » обозначается произвольная бинарная логическая операция (**исключающего или**)

ГОСТ 28147–89 содержит описание алгоритмов нескольких уровней. На самом верхнем находятся практические алгоритмы, предназначенные для шифрования массивов данных и выработки для них имитовставки. Все они опираются на три алгоритма низшего уровня, называемые **базовые циклами**. Они имеют следующие названия и обозначения

- Цикл зашифрования (32–3);
- Цикл расшифрования (32 - P);
- Цикл выработки имитовставки (16-3).

Базовых циклов представляет собой многократное повторение одной единственной процедуры: **основным шагом криптопреобразования**.

В ГОСТе ключевая информация состоит из двух структур данных. Помимо собственно **ключа**, необходимого для всех шифров, она содержит еще и **таблицу замен**.

**Ключ** – массив из восьми 32-битовых элементов кода, обозначается символом **K**:  $K = \{K_i\}_{0 \leq i \leq 7}$ . В ГОСТе элементы ключа используются как 32-разрядные целые числа без знака:  $0 \leq K_i \leq 2^{32}$ . Размер ключа составляет  $32 \cdot 8 = 256$  бит или 32 байта.

**Таблица замен** может быть представлена в виде матрицы размера 8 16, содержащей 4-битовые элементы, которые можно представить в виде целых чисел от 0 до 15. Строки **таблицы замен** – **узлами замен**, каждый **узел замен** должен содержать 16 различных чисел от 0 до 15 в произвольном порядке. Таблица обозначается символом **H**:  $H = \{H_{i,j}\}_{0 \leq i \leq 7, 0 \leq j \leq 15}$ ,  $0 \leq H_{i,j} \leq 15$ . Общий объем таблицы замен равен: 8 узлов  $\times$  16 элементов/узел  $\times$  4 бита/элемент = 512 бит или 64 байта.

### 3.2 Основной шаг криптопреобразования

Основной шаг криптопреобразования по своей сути является оператором, определяющим преобразование 64-битового блока данных. Дополнительным параметром этого оператора является 32-битный блок, в качестве которого используется какой-либо элемент ключа. Схема алгоритма основного шага приведена на рисунке 5:

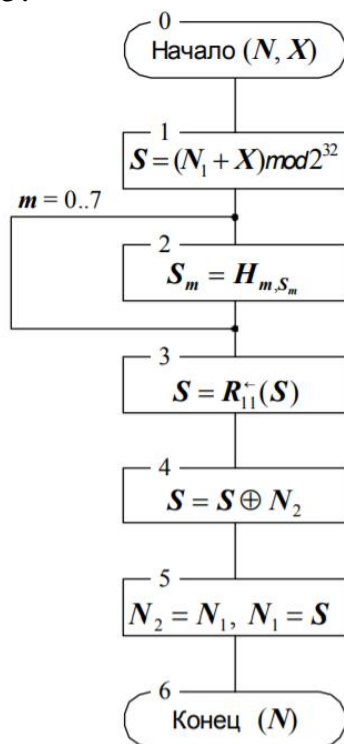


Рисунок 5. Схема основного шага криптопреобразования алгоритма ГОСТ 28147-89

**Шаг 0.** Определяет исходные данные для основного шага криптопреобразования:

- **N** – преобразуемый 64-битовый блок данных, в ходе выполнения шага его

младшая ( $N_1$ ) и старшая ( $N_2$ ) части обрабатываются как отдельные 32-битовые целые числа без знака. Таким образом, можно записать  $N = (N_1, N_2)$ .

- $X$  – 32-битовый элемент ключа;

Шаг 1. Сложение с ключом. Младшая половина преобразуемого блока складывается по модулю  $2^{32}$  с используемым на шаге элементом ключа, результат передается на следующий шаг;

Шаг 2. Поблочная замена. 32-битовое значение, полученное на предыдущем шаге, интерпретируется как массив из восьми 4-битовых блоков кода:  $S = (S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7)$ . Далее значение каждого из восьми блоков заменяется новым, которое выбирается по таблице замен следующим образом: значение блока  $S_i$  меняется на  $S_i$ -тый по порядку элемент (нумерация с нуля)  $i$ -того узла замен (т. е.  $i$ -той строки таблицы замен, нумерация также с нуля). Другими словами, в качестве замены для значения блока выбирается элемент из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа. Теперь становится понятным размер таблицы замен: число строк в ней равно числу 4-битовых элементов в 32-битовом блоке данных, то есть восьми, а число столбцов равно числу различных значений 4-битового блока данных, равному как известно  $2^4$ , шестнадцати.

Шаг 3. Циклический сдвиг на 11 бит влево. Результат предыдущего шага сдвигается циклически на 11 бит в сторону старших разрядов и передается на следующий шаг. На схеме алгоритма символом  $\overleftarrow{R}_{11}$  обозначена функция циклического сдвига своего аргумента на 11 бит влево, т. е. в сторону старших разрядов.

Шаг 4. Побитовое сложение: значение, полученное на шаге 3, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.

Шаг 5. Сдвиг по цепочке: младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего шага.

Шаг 6. Полученное значение преобразуемого блока возвращается как результат выполнения алгоритма основного шага криптопреобразования.

### 3.3 Базовые циклы криптографических преобразований

Базовые циклы построены из основных шагов криптографического преобразования. В процессе выполнения основного шага используется только один 32-битовый элемент ключа, в то время как ключ ГОСТа содержит восемь таких элементов. Следовательно, чтобы ключ был использован полностью, каждый из базовых циклов должен многократно выполнять основной шаг с различными его элементами. Вместе с тем, в каждом базовом цикле все элементы ключа должны быть использованы одинаковое число раз, по соображениям стойкости шифра это число должно быть больше одного.

Цикл расшифрования должен быть обратным циклу зашифрования. Другими словами, зашифрование блока данных теоретически может быть выполнено

с помощью цикла расшифрования, в этом случае расшифрование блока данных должно быть выполнено циклом зашифрования. Из двух взаимно обратных циклов любой может быть использован для зашифрования, тогда второй должен быть использован для расшифрования данных, однако стандарт ГОСТ28147-89 закрепляет роли за циклами и не предоставляет пользователю права выбора в этом вопросе. Цикл выработки имитовставки вдвое короче циклов шифрования, порядок использования ключевых элементов в нем такой же, как в первых 16 шагах цикла зашифрования.

Схемы базовых циклов приведены на рисунках ба-в. Каждый из них принимает в качестве аргумента и возвращает в качестве результата 64-битовый блок данных, обозначенный на схемах  $N$ . Символ Шаг( $N, X$ ) обозначает выполнение основного шага криптопреобразования для блока данных  $N$  с использованием ключевого элемента  $X$ .

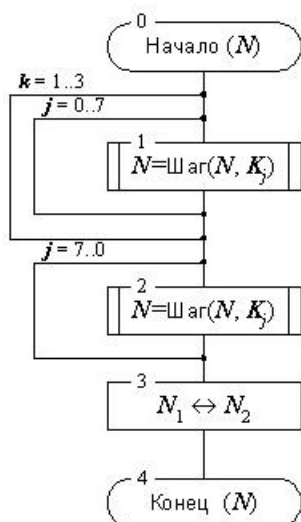


Рисунок ба. Схема цикла зашифрования 32-3

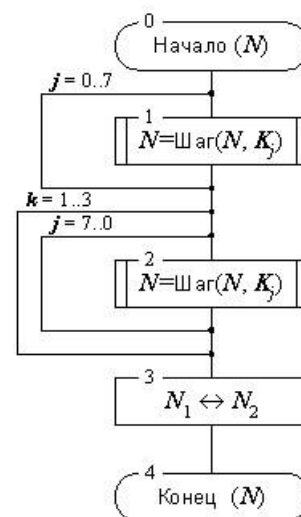


Рисунок бб. Схема цикла расшифрования 32-Р

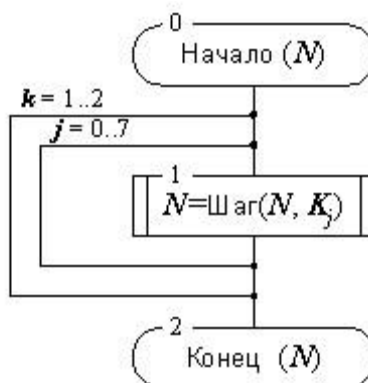


Рисунок бв. Схема цикла выработки имитовставки 16-3

## Глава 4. ПОШАГОВАЯ ВИЗУАЛИЗАЦИЯ ВСЕХ РЕЖИМОВ ШИФРОВАНИЯ ГОСТ28147-89

### 4.1 Основные режимы шифрования

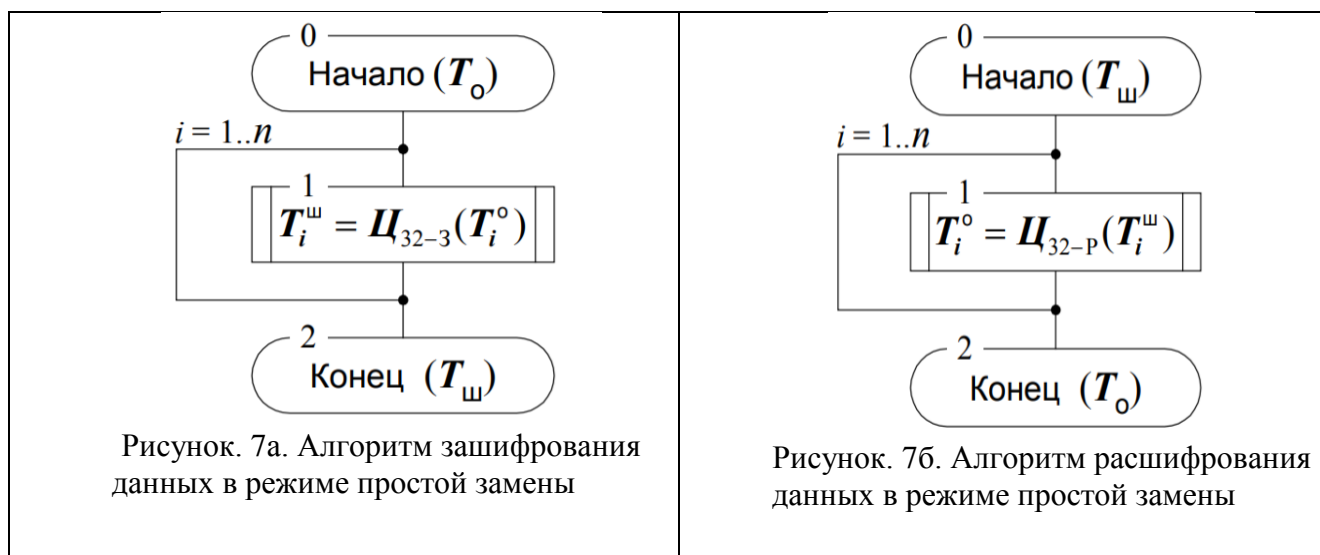
ГОСТ 28147-89 предусматривает три следующих режима шифрования данных:

- Простая замена,
- Гаммирование,
- Гаммирование с обратной связью,
- Один дополнительный режим выработки имитовставки.

В любом из этих режимов данные обрабатываются блоками по 64 бита, на которые разбивается массив, подвергаемый криптографическому преобразованию, именно поэтому ГОСТ относится к блочным шифрам. Однако в двух режимах гаммирования есть возможность обработки неполного блока данных размером меньше 8 байт, что существенно при шифровании массивов данных с произвольным размером, который может быть не кратным 8 байтам.

### 4.2 Простая замена

Зашифрование в данном режиме заключается в применении цикла 32-З к блокам открытых данных, расшифрование – цикла 32-Р к блокам зашифрованных данных. Это наиболее простой из режимов, 64-битовые блоки данных обрабатываются в нем независимо друг от друга. Схемы алгоритмов зашифрования и расшифрования в режиме простой замены приведены на рисунках 7а и б соответственно.





Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам:  $|T_o| = |T_{ш}| = 64 \cdot n$ , после выполнения операции размер полученного массива данных не изменяется. Данный режим шифрования имеет следующие особенности:

- Так как блоки данных шифруются независимо друг от друга и от их позиции в массиве, при зашифровании двух одинаковых блоков открытого текста получаются одинаковые блоки шифртекста и наоборот.
- Если длина шифруемого массива данных не кратна 8 байтам или 64 битам, возникает проблема, чем и как дополнять последний неполный блок данных массива до полных 64 бит.

Перечисленные выше особенности делают практически невозможным использование режима простой замены, ведь он может применяться только для шифрования массивов данных с размером кратным 64 битам, не содержащим повторяющихся 64-битовых блоков. Это почти так, но есть одно очень важное исключение: размер ключа составляет 32 байта, а размер таблицы замен – 64 байта. Кроме того, наличие повторяющихся 8-байтовых блоков в ключе или таблице замен будет говорить об их весьма плохом качестве, поэтому в реальных ключевых элементах такого повторения быть не может. Таким образом, режим простой замены подходит для шифрования ключевой информации.

### 4.3 Гаммирование

От недостатков режима простой замены можно избавиться, для этого необходимо сделать возможным шифрование блоков с размером менее 64 бит и обеспечить зависимость блока шифртекста от его номера, иными словами, рандомизировать процесс шифрования. В ГОСТе это достигается двумя различными способами в двух режимах шифрования, предусматривающих гаммирование. Гаммирование – это снятие на зашифрованные данные криптографической гаммы, то есть последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных открытых данных. Для наложения гаммы при зашифровании и ее снятия при расшифровании должны использоваться взаимно обратные бинарные операции. В ГОСТе для этой цели используется операция побитного сложения по модулю 2.

Описание режима гаммирования. Гамма получается следующим образом: с помощью некоторого алгоритмического рекуррентного генератора последовательности чисел (РГПЧ) вырабатываются 64-битовые блоки данных, которые далее подвергаются преобразованию по циклу 32-3, то есть зашифрованию в режиме простой замены, в результате получают блоки гаммы. Благодаря тому, что наложение и снятие гаммы осуществляется при помощи одной и той же операции побитового исключающего или, алгоритмы зашифрования и расшифрования в режиме гаммирования идентичны, их общая схема приведена на рисунке 8.

РГПЧ, используемый для выработки гаммы, является рекуррентной функцией:

$$\Omega_{i+1} = f(\Omega_i),$$

где  $\Omega_i$  – элементы рекуррентной последовательности,  $f$  – функция преобразования.

В действительности  $\Omega_0$  элемент данных является параметром алгоритма для режимов гаммирования, на схемах он обозначен как  $S$ , и называется в ГОСТе – начальным заполнением (синхропосылка) одного из регистров шифрователя. Разработчики ГОСТа использовали для инициализации РГПЧ результат преобразования синхропосылки по циклу 32-3:  $\Omega_0 = C_{32-3}(S)$ . Последовательность элементов, вырабатываемых РГПЧ, целиком зависит от его начального заполнения. РГПЧ:  $\Omega_i = f_i(\Omega_0)$ , где  $f_i(X) = f(f_{i-1}(X))$ ,  $f_0(X) = X$ . С учетом преобразования по алгоритму простой замены добавляется еще и зависимость от ключа:

$$\Gamma_i = C_{32-3}(\Omega_i) = C_{32-3}(f_i(\Omega_0)) = C_{32-3}(f_i(C_{32-3}(S))) = \varphi_i(S, K),$$

где  $\Gamma_i$  –  $i$ -тый элемент гаммы,  $K$  – ключ.

Таким образом, последовательность элементов гаммы для использования в режиме гаммирования однозначно определяется ключевыми данными и синхропосылкой.

Для обратимости процедуры шифрования в процессах за- и расшифрования должна использоваться одна и та же синхропосылка. Из требования уникальности гаммы, невыполнение которого приводит к катастрофическому снижению стойкости шифра, следует, что для шифрования двух различных массивов данных на одном ключе необходимо обеспечить использование различных синхропосылок. Это приводит к необходимости хранить или передавать синхропосылку по каналам связи вместе с зашифрованными данными, хотя в отдельных особых случаях она может быть предопределена или вычисляться особым образом, если исключается шифрование двух массивов на одном ключе.

Для РГПЧ, используемого в ГОСТе для генерации элементов гаммы, не предъявляются требования обеспечения каких-либо статистических характеристик вырабатываемой последовательности чисел. РГПЧ спроектирован исходя из необходимости выполнения следующих условий:

- Период повторения последовательности чисел, вырабатываемой РГПЧ, не должен сильно (в процентном отношении) отличаться от максимально возможного при заданном размере блока значения  $2^{64}$ ;
- Соседние значения, вырабатываемые РГПЧ, должны отличаться друг от друга в каждом байте, иначе задача криптоаналитика будет упрощена;
- РГПЧ должен быть достаточно просто реализуем как аппаратно, так и программно на наиболее распространенных типах процессоров, большинство из которых, как известно, имеют разрядность 32 бита.

Исходя из перечисленных принципов создатели ГОСТа спроектировали весьма удачный РГПЧ, имеющий следующие характеристики:

- в 64-битовом блоке старшая и младшая части обрабатываются независимо друг от друга:

$$\Omega_i = (\Omega_i^0, \Omega_i^1), |\Omega_i^0| = |\Omega_i^1| = 32, \Omega_{i+1}^0 = \hat{f}(\Omega_i^0), \Omega_{i+1}^1 = \tilde{f}(\Omega_i^1);$$

фактически, существуют два независимых РГПЧ для старшей и младшей частей блока.

- рекуррентные соотношения для старшей и младшей частей следующие:

$$\Omega_{i+1}^0 = (\Omega_i^0 + C_1) \bmod 2^{32}, \text{ где } C_1 = 1010104_{16};$$

$$\Omega_{i+1}^1 = (\Omega_i^1 + C_2 - 1) \bmod (2^{32} - 1) + 1,$$

где  $C_2 = 1010104_{16}$ ;

Второе выражение нуждается в комментариях, так как в тексте ГОСТа приведено нечто другое:  $\Omega_{i+1}^1 = (\Omega_i^1 + C_2) \bmod (2^{32} - 1)$ , с тем же значением константы  $C_2$ . В тексте стандарта дается комментарий, что под операцией взятия остатка по модулю понимается не то же самое, что и в математике. Отличие заключается в том, что  $(2^{32}-1) \bmod (2^{32}-1) = (2^{32}-1)$ , а не 0. И это упрощает реализацию формулы, а математически корректное выражение для нее приведено выше. Схема алгоритма шифрования в режиме гаммирования приведена на рисунке 8, ниже изложены пояснения к схеме:

Шаг 0. Определяет исходные данные для основного шага криптопреобразования:

- $T_{o(ш)}$  – массив открытых (зашифрованных) данных произвольного размера, подвергаемый процедуре зашифрования (расшифрования), по ходу процедуры массив подвергается преобразованию порциями по 64 бита;
- $S$  – *синхропосылка*, 64-битовый элемент данных, необходимый для инициализации генератора гаммы;

Шаг 1. Начальное преобразование синхропосылки, выполняемое для ее «рандомизации», то есть для устранения статистических закономерностей, присутствующих в ней, результат используется как начальное заполнение РГПЧ;

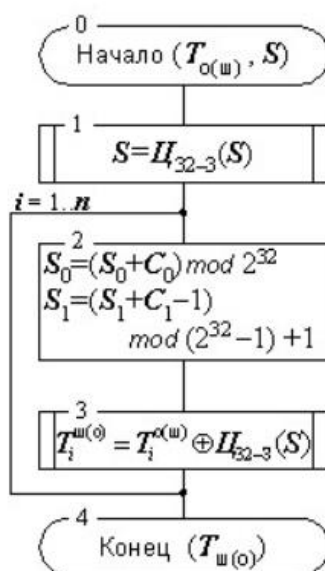


Рисунок. 8. Алгоритм зашифрования (расшифрования) данных в режиме гаммирования.

Шаг 2. Один шаг работы РГПЧ, реализующий его рекуррентный алгоритм. В ходе данного шага старшая ( $S_1$ ) и младшая ( $S_0$ ) части последовательности данных вырабатываются независимо друг от друга;

Шаг 3. Гаммирование. Очередной 64-битовый элемент, выработанный РГПЧ, подвергается процедуре за- шифрования по циклу 32–3, результат используется как элемент гаммы для зашифрования (расшифрования) очередного блока открытых (зашифрованных) данных того же размера.

Шаг 4. Результат работы алгоритма – зашифрованный (расшифрованный) массив данных.

Ниже перечислены особенности гаммирования как режима шифрования:

- Одинаковые блоки в открытом массиве данных дадут при зашифровании различные блоки шифртекста, что позволит скрыть факт их идентичности
- Поскольку наложение гаммы выполняется побитно, шифрование неполного блока данных легко выполнимо как шифрование битов этого неполного блока, для чего используется соответствующие биты блока гаммы.
- Синхропосылка, использованная при зашифровании, каким-то образом должна быть передана для использования при расшифровании.
- Использовать predetermined значение синхропосылки или вырабатывать ее синхронно источником и приемником по определенному закону, в этом случае изменение размера передаваемого или хранимого массива данных отсутствует;

Оба способа дополняют друг друга, и в тех редких случаях, где не работает первый, наиболее употребительный из них, может быть использован второй, более экзотический.

Режим гаммирования имеет еще одну особенность. В этом режиме биты массива данных шифруются независимо друг от друга. Таким образом,

- каждый бит шифртекста зависит от соответствующего бита открытого текста и, естественно, порядкового номера бита в массиве:  $t_i^{\text{III}} = t_i^0 \oplus \gamma_i = f(t_i^0, i)$ . Из этого вытекает, что изменение бита шифртекста на противоположное значение приведет к аналогичному изменению бита открытого текста на противоположный:  $\bar{t}_i^{\text{III}} = t_i^{\text{III}} \oplus 1 = (t_i^0 \oplus \gamma_i) \oplus 1 = (t_i^0 \oplus 1) \oplus \gamma_i = \bar{t}_i^0 \oplus \gamma_i$ , где  $\bar{t}$  обозначает инвертированное по отношению к  $t$  значение бита ( $\bar{0} = 1, \bar{1} = 0$ ).

Данное свойство дает злоумышленнику возможность воздействуя на биты шифр- текста вносить предсказуемые и даже целенаправленные изменения в соответствующий открытый текст, получаемый после его расшифрования, не обладая при этом секретным ключом.

#### 4.4 Гаммирование с обратной связью

Данный режим очень похож на режим гаммирования и отличается от него только способом выработки элементов гаммы – очередной элемент гаммы вырабатывается как результат преобразования по циклу 32-3 предыдущего блока зашифрованных данных, а для зашифрования первого блока массива данных элемент гаммы вырабатывается как результат преобразования по тому же циклу синхропосылки. Этим достигается зацепление блоков – каждый блок шифртекста в этом режиме зависит от соответствующего и всех предыдущих блоков открытого текста. Поэтому данный режим иногда называется гаммированием с зацеплением блоков. На стойкость шифра факт зацепления блоков не оказывает никакого влияния.

Схема алгоритмов за- и расшифрования в режиме гаммирования с обратной связью приведена на рисунке 9 и ввиду своей простоты в комментариях не нуждается. Шифрование в режиме гаммирования с обратной связью обладает теми особенностями, что и шифрование в режиме обычного гаммирования, за исключением влияния искажений шифртекста на соответствующий открытый текст. Для сравнения запишем функции расшифрования блока для обоих упомянутых режимов:

$$T_i^{\circ} = T_i^{\text{III}} \oplus \Gamma_i, \text{ гаммирование.}$$

$$T_i^{\circ} = T_i^{\text{III}} \oplus \text{Ц}_{32-3}(T_{i-1}^{\text{III}}), \text{ гаммирование с обратной связью;}$$

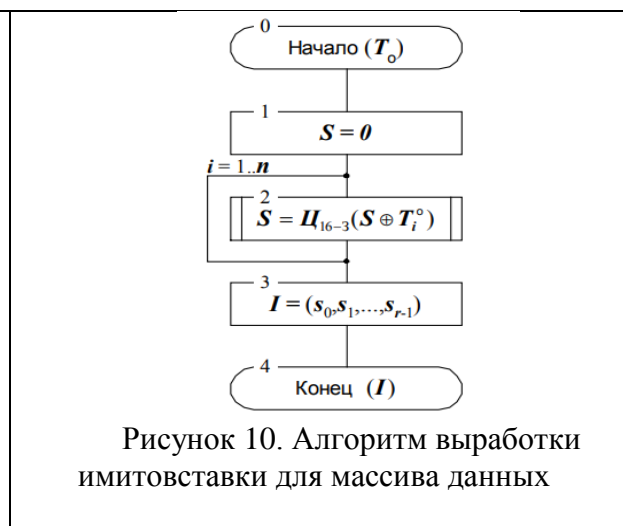
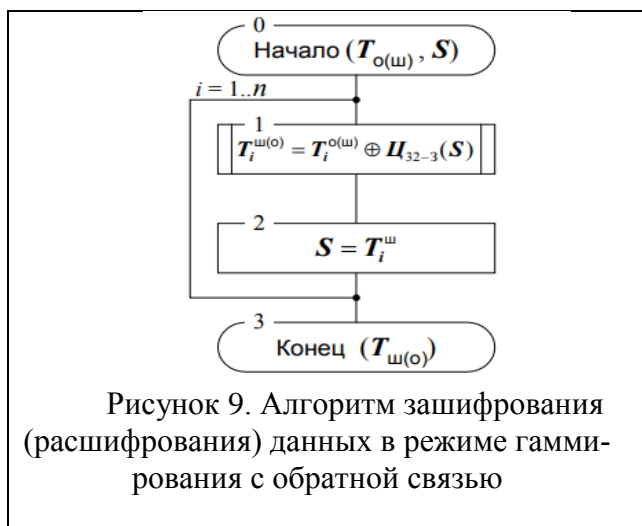
Если в режиме обычного гаммирования изменения в определенных битах шифр текста влияют только на соответствующие биты открытого текста, то в режиме гаммирования с обратной связью картина несколько сложнее. Как видно из соответствующего уравнения, при расшифровании блока данных в режиме гаммирования с обратной связью, блок открытых данных зависит от соответствующего и предыдущего блоков зашифрованных данных. Поэтому, если внести искажения в зашифрованный блок, то после расшифрования искаженными окажутся два блока открытых данных – соответствующий и следующий за ним, причем искажения в первом случае будут носить тот же характер, что и в режиме гаммирования, а во втором случае – как в режиме простой замены.

## 4.5 Режим выработки имитовставки

Для решения задачи обнаружения искажений в зашифрованном массиве данных с заданной вероятностью в ГОСТе предусмотрен дополнительный режим криптографического преобразования – выработка имитовставки. Имитовставка – это контрольная комбинация, зависящая от открытых данных и секретной ключевой информации. Целью использования имитовставки является обнаружение всех случайных или преднамеренных изменений в массиве информации. Проблема, изложенная в предыдущем пункте, может быть успешно решена с помощью добавления к шифрованным данным имитовставки. Для потенциального злоумышленника две следующие задачи практически неразрешимы, если он не владеет ключевой информацией:

- вычисление имитовставки для заданного открытого массива информации;
- подбор открытых данных под заданную имитовставку;

Схема алгоритма выработки имитовставки приведена на рисунке 10. В качестве имитовставки берется часть блока, полученного на выходе, обычно – 32 его младших бита. При выборе размера имитовставки надо принимать во внимание, что вероятность успешного навязывания ложных данных равна величине  $2^{-l}$  на одну попытку подбора, если в распоряжении злоумышленника нет более эффективного метода подбора, чем простое угадывание. При использовании имитовставки размером 32 бита эта вероятность равна  $2^{-32} \approx 0,23 \cdot 10^{-9}$ .



## Глава 5. ШИФРОВАНИЕ ПО АЛГОРИТМУ ГОСТ 28147-89. ПОШАГОВЫЙ РЕЖИМ

### 5.1 Основные режимы шифрования

Запуск модуля осуществляется как из панели управления программы «Криптолаб» (см. пункт 5 настоящего руководства, Рис. 5.2), так и при помощи стандартных средств ОС. После запуска модуля будет открыто его основное окно (рис. 11.1), которое имеет два основных поля: входные/выходные данные и поле для работы в пошаговом режиме.

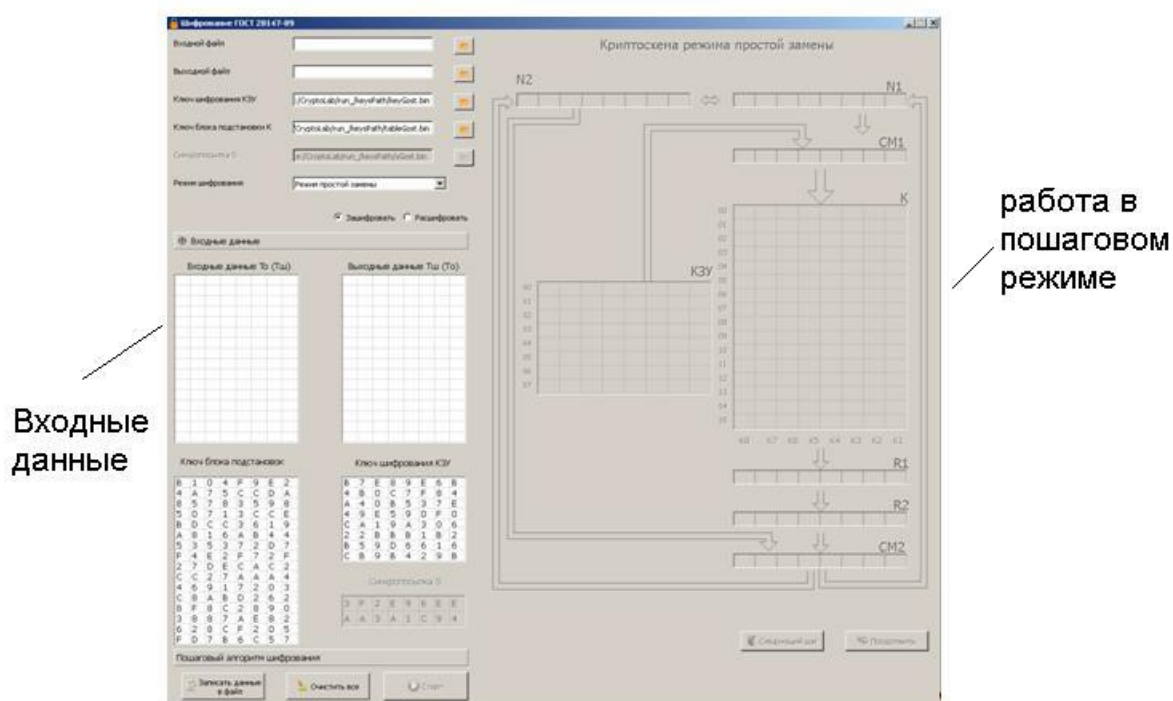


Рисунок 11. Основное окно модуля шифрования по алгоритму ГОСТ 28147-89

Выход из модуля «Шифрование по алгоритму ГОСТ 28147-89» программы «Криптолаб» осуществляется при закрытии его основного окна, нажатии клавиши «Esc» либо комбинации «горячих» клавиш «Alt+F4». Если модуль был открыт из панели управления, то при закрытии модуля, произойдет автоматический переход в панель управления программы «Криптолаб».

### 5.2 Основные функции

Данный модуль позволяет проводить зашифрование/расшифрование данных в режимах простой замены, гаммирования, гаммирования с обратной связью, а также вычисление имитовставки в пошаговом режиме. Также модуль позволяет записать данные в файл.

### 5.3 Ввод входных данных

Для всех режимов необходимо задать входной файл для шифрования, ключ шифрования, таблицу замены. Для режимов гаммирования и гаммирования с обратной связью необходимо задать синхропосылку. Размеры данных входных параметров отражены в Приложении Б. Для смены режима шифрования необходимо раскрыть выпадающий список и выбрать нужный режим (рис. 11.2). При выборе режима шифрования гаммирования или гаммирования с обратной связью станет доступным окно для ввода синхропосылки. Для того чтобы задать входные данные из файла необходимо нажать на кнопку. Все ошибки, которые возникают при установке входных данных, описаны в Приложении А.

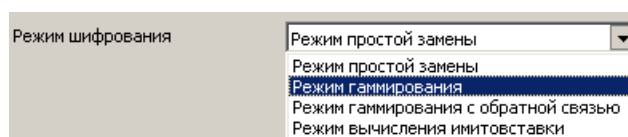


Рисунок 12. Выбор режима шифрования

Все введенные данные отображаются в соответствующих таблицах в поле входных данных (рис. 11.1). Данные отображаются в байтах. В модуле стоит ограничение на размер входных данных. Для шифрования файла в пошаговом режиме необходимо выбирать файл для зашифрования/расшифрования не более чем 64 байта. Необходимо отметить, что для режима шифрования простой замены размер входных данных должен быть кратен 8 байтам, как это заявлено в стандарте. По умолчанию установлено зашифрование данных в режиме простой замены. После того, как все файлы будут введены, станет доступной кнопка «Старт».

### 5.4 Работа в пошаговом режиме

Для начала работы в пошаговом режиме нужно нажать кнопку «Старт». Станет активным поле для работы в пошаговом режиме (рис. 11.3–11.6).



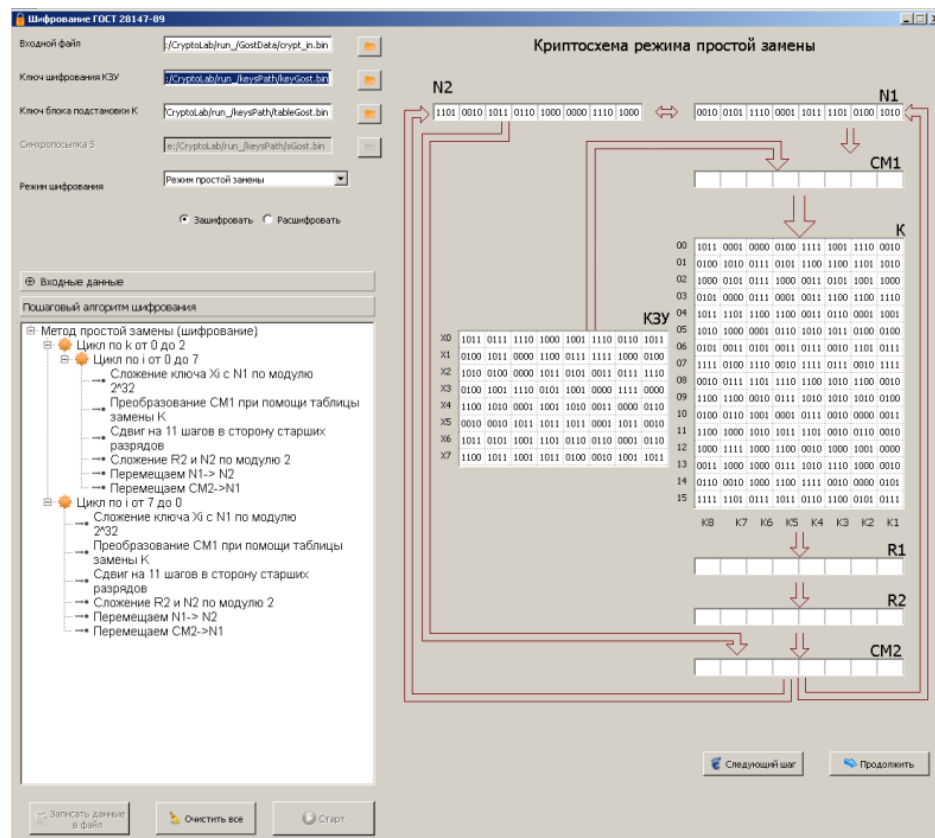


Рисунок 13. Начало работы в режиме простой замены

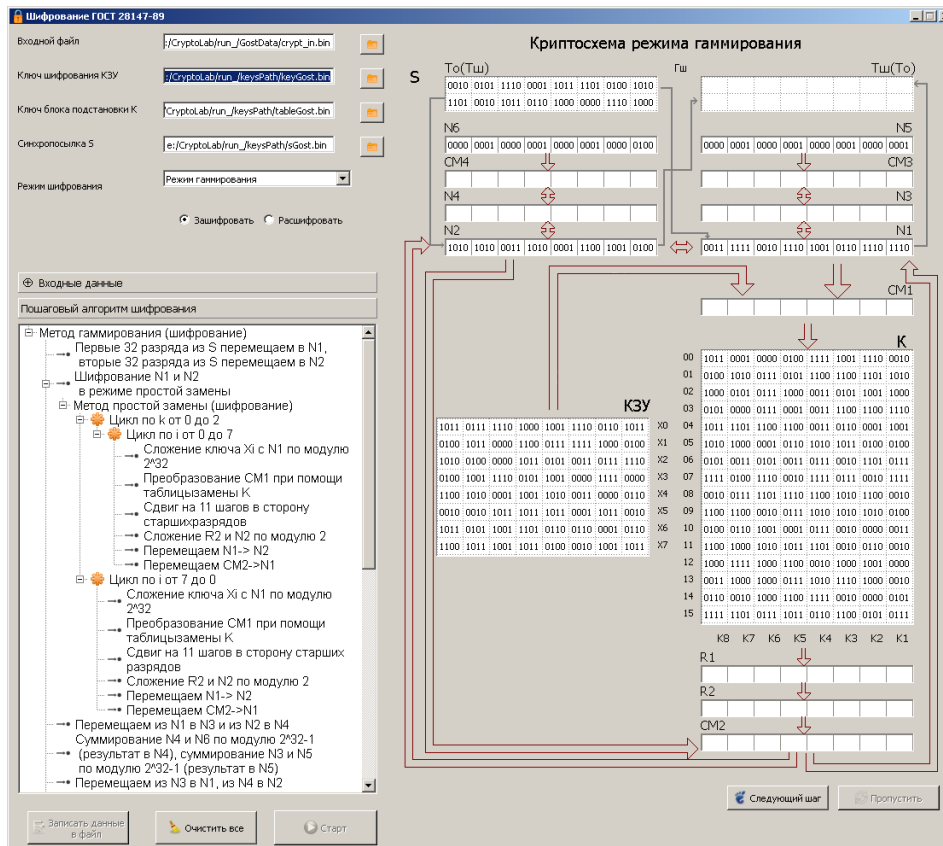


Рисунок 14. Начало работы в режиме гаммирования

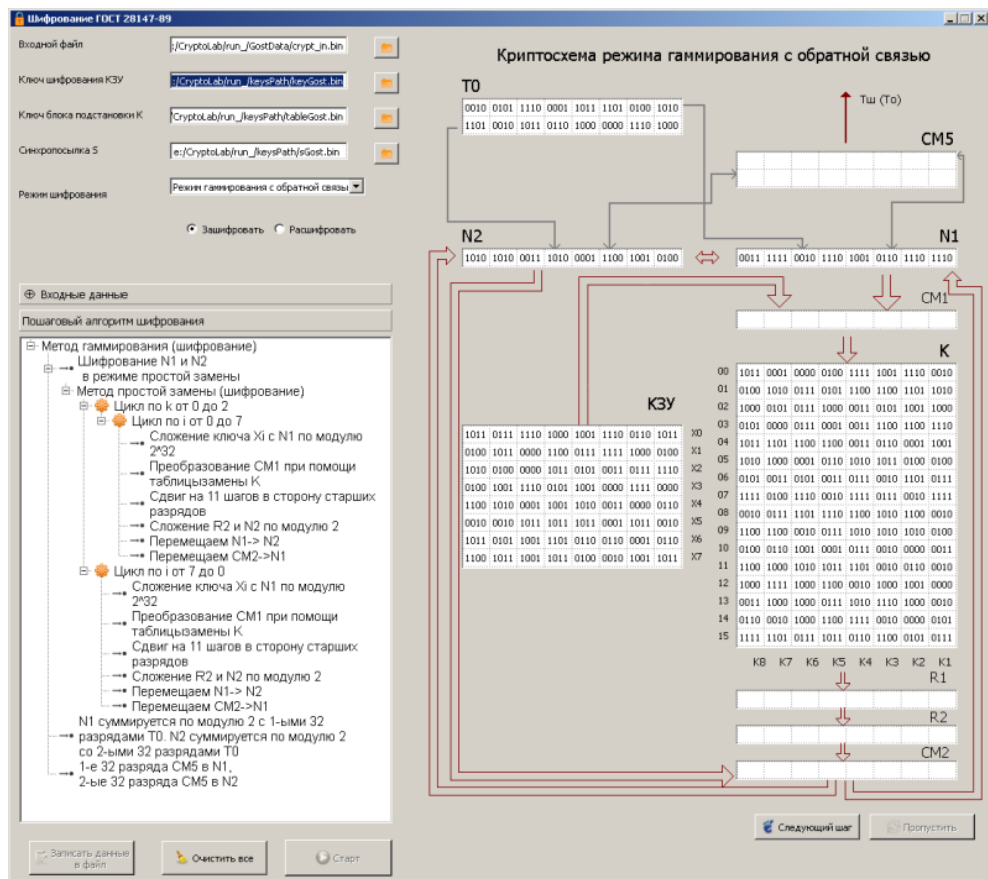


Рисунок 15. Начало работы в режиме гаммирования с обратной связью

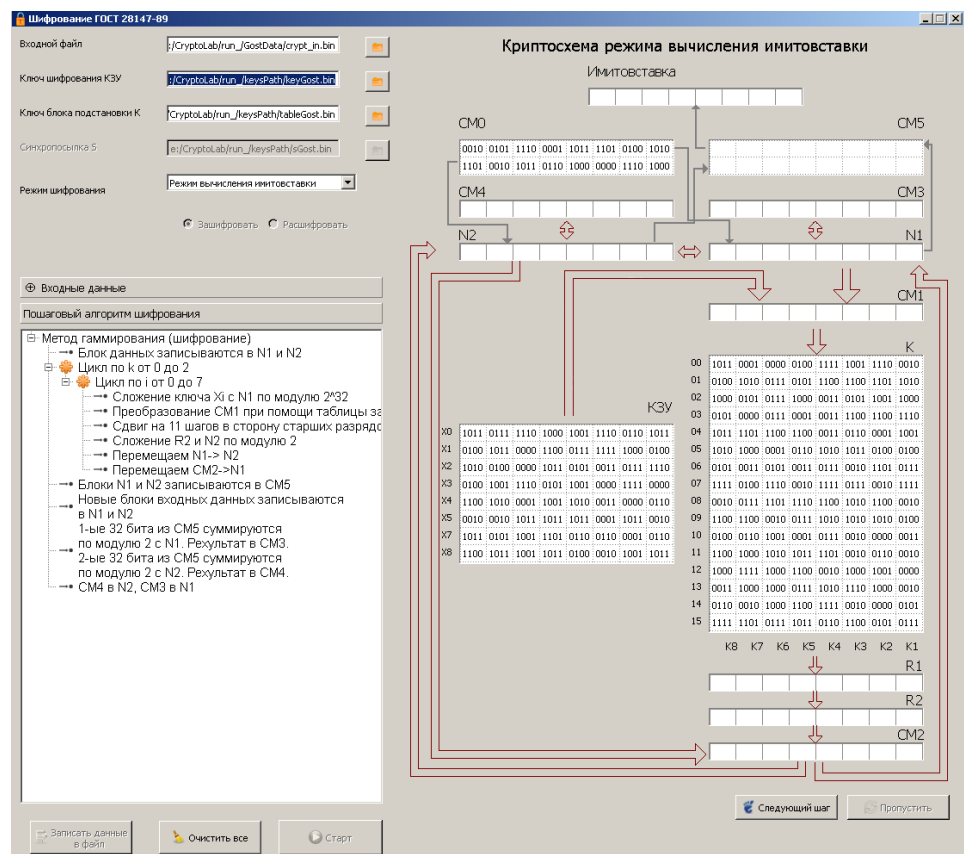


Рисунок 16. Начало работы в режиме выработки имитовставки

Вкладка «Пошаговый алгоритм шифрования» с текстовым описанием алгоритма открывается по умолчанию при запуске модуля. Для того чтобы просмотреть входные данные, необходимо нажать на вкладку «Входные данные».

Оператору доступны для работы в пошаговом режиме две кнопки «Следующий шаг» и «Продолжить». При нажатии на кнопку «Следующий шаг» происходит операция одного хода шифрования. Активные элементы данного шага выделяются синим цветом. Также во вкладке с описанием режима шифрования активное действие также выделяется синим цветом (рис. 11.7).

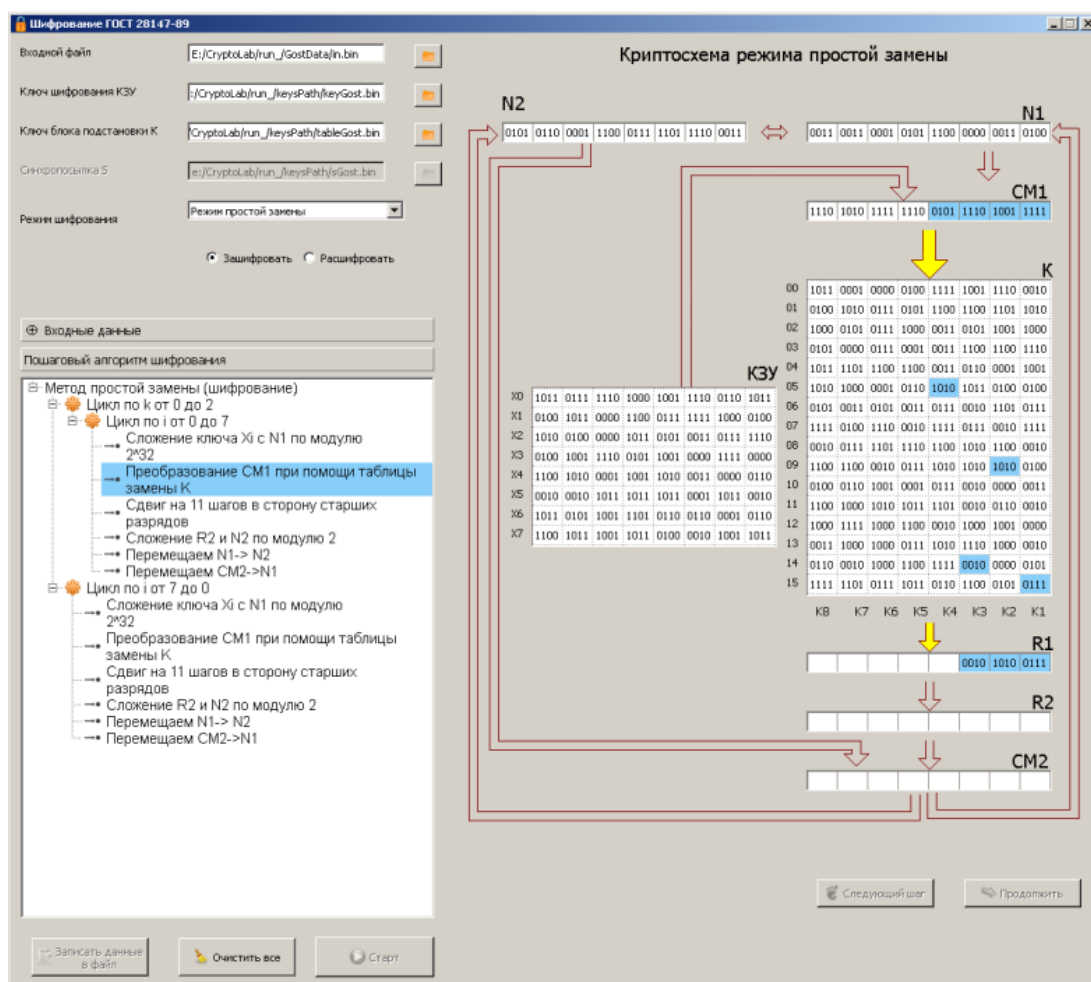


Рисунок 17. Пошаговый ход работы программы


Алгоритм шифрование простой замены состоит в 32 циклах шифрования. Для того чтобы не повторять эту процедуру каждый раз предусмотрена кнопка «Пропустить». Ее нажатие, пропускает только алгоритм шифрования данных в режиме простой замены. Стоит отметить, что алгоритм режима шифрования блока данных входит во все остальные режимы шифрования ГОСТ 28147-89. Причем в режимах гаммирования, гаммирования с обратной связью и выработки имитовставки процесс шифрования блока

данных с использованием алгоритма шифрования в режиме простой замены, реализован анимационным образом, что позволяет пользователю не нажимать постоянно на кнопку «Следующий шаг».

### 5.5 Запись данных в файл

Кнопка «Запись в файл» становится активной только в том случае, когда алгоритм шифрования данных доведен до конца и во входных данных отобразятся зашифрованные данные. Процедура записи зашифрованных данных в файл подробно описана в Приложении В.

### 5.6 Очистка форм

Для того, чтобы очистить все входные данные, необходимо нажать на кнопку . При этом если проводилась криптографическая операция в поле пошагового режима, то данная операция завершится, данные сохранены не будут. Поле пошагового режима станет неактивным. Обратить операцию нельзя.

## **ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ**

1. Изучить теоретическую информацию о работе алгоритма шифрования ГОСТ28147-89.
2. Изучить пошаговую визуализацию работы алгоритма в программе «Криптолаб».
3. Выполнить реализацию алгоритма шифрования ГОСТ28147-89 на одном из языков программирования.

## **СОДЕРЖАНИЕ ОТЧЕТА**

1. Краткое описание алгоритма шифрования ГОСТ28147-89.
2. Описание основных шагов алгоритма шифрования ГОСТ28147-89.
3. Описание разработанной программы.
4. Выводы.

## **КОНТРОЛЬНЫЕ ВОПРОСЫ**

1. Дайте определение блочным алгоритмам.
2. Опишите структуру Цикла Фейстеля.
3. Опишите логику построения шифра и структуру ключевой информации ГОСТа 28147-89.
4. Опишите основной шаг криптопреобразования.
5. Опишите базовые циклы криптографических преобразований.
6. Опишите основные режимы шифрования.

## ЛИТЕРАТУРА

1. Бабенко, Л. К. Современные алгоритмы блочного шифрования и методы их анализа: учеб, пособие / Л. К. Бабенко, Е. А. Ищукова. — М.: Гслиос-АРВ, 2006.
2. Черемушкин А.В. Криптографические протоколы. Основные свойства и уязвимости. - М.: Издательский центр "Академия", 2009, 272 с.
3. Хорев, П.Б. Методы и средства защиты информации в компьютерных системах / П.Б. Хорев. -М.: АСАСЕМ А, 2005. 254с.
4. ГОСТ 28147–89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. - Москва: ИПК Издательство стандартов, 1996. - 26 с.
5. Ссылка на ГОСТ:  
<https://shop.belgiss.by/ru/gosudarstvennye-standarty/gost-28147-89>