ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»

(НИУ «БелГУ»)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

КАФЕДРА МАТЕМАТИЧЕСКОГО И ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ

РАЗРАБОТКА ОБУЧАЮЩЕЙ ПРОГРАММЫ ПО РЕЖИМАМ ШИФРОВАНИЯ ГОСТ 28147-89

Выпускная квалификационная работа

обучающегося по направлению подготовки 02.03.01
Математика и компьютерные науки

очной формы обучения группы 07001303

Ряднова Николая Николаевича

Научный руководитель:

к.т.н. доцент

Румбешт Вадим Валерьевич

БЕЛГОРОД, 2017

СОДЕРЖАНИЕ

ВВЕДЕНИЕ……..……...3
1. ПОСТАНОВКА ЗАДАЧИ НА РАЗРАБОТКУ ОБУЧАЮЩЕЙ ПРОГРАММЫ ПО РЕЖИМАМ ШИФРОВАНИЯ ГОСТ 28147-89…….….5
1.1. Обзор криптографического преобразования ГОСТ 28147-89…..5
1.2. Обзор и анализ обучающих программ.…….….20
1.3. Требования к обучающей программе…….28
2. ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ…...…….....32
2.1. Предварительное проектирование.……...32
2.2. Проектирование модульной структуры программы……...…...34
2.3. Программная реализация……..…….…….…...37
3. АПРОБАЦИЯ ОБУЧАЮЩЕЙ ПРОГРАММЫ ПО РЕЖИМАМ ШИФРОВАНИЯ ГОСТ 28147-89..……..44
3.1. Программа и методика испытаний……...44
3.2. Результаты испытаний.……..45
ЗАКЛЮЧЕНИЕ…….50
СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ……..51
ПРИЛОЖЕНИЕ…….54

ВВЕДЕНИЕ

В современном информационном обществе остро стоит вопрос защиты информации. Если информация защищена шифрованием, никто не в силах ее прочесть или изменить. Если мы используем шифрование, то знаем, с кем имеем дело, поэтому шифрование можно интерпретировать и как аутентификацию. Шифрование является важнейшим средством обеспечения безопасности. Обучение студентов методам криптографической защиты также важно: в самом деле, на сегодняшний день криптография используется в цифровых технологиях, что делает представление многих сфер жизни общества без её наличия немыслимым. Этот процесс охватывает всё больше и больше различных сфер жизни: логин и пароль, аутентификация и идентификация, электронная цифровая подпись – использование перечисленных и многих других схожих понятий стало обычным явлением для повседневной жизни.

В данной работе затрагиваются вопросы обучения криптографической защиты информации, происходит обзор основных алгоритмов для решения этой задачи и даётся математическое обоснование правильности их работы. Излагается главная информация об основных направлениях, наиболее актуальных новшествах и примерах классической криптографии. Чёткость и связность изложения помогают студентам достичь максимально полного и комплексного восприятия предмета. Таким образом, студенты, обучающиеся на специальностях, связанными с информационными технологиями, должны быть не только грамотны в вопросах защиты информации, но и способны реализовать и дорабатывать алгоритмы шифрования.

Целью данной выпускной квалификационной работы является разработка обучающей программы по режимам шифрование ГОСТ 28147-89.

Для её достижения были поставлены следующие задачи:

- Провести обзор предметной области и выполнить анализ литературы по теме шифрования на основе ГОСТ 28147-89;
- Раскрыть возможность использования режимов шифрования ГОСТ в качестве средства обучения;
- Изучить средства информационных технологий для проектирования программы учебного назначения и обосновать выбор средств, для разработки собственной компьютерной обучающей программы по режимам шифрования ГОСТ 28147-89;
- Разработать обучающую программу по режимам шифрования ГОСТ 28147-89;
- Экспериментально протестировать обучающую программу по режимам шифрования ГОСТ 28147-89.

В первой главе идет постановка задачи на разработку обучающей программы по режимам шифрования ГОСТ 28147-89,обзор и анализ обучающих программ и составление требований.

Во второй главе происходит предварительное проектирование приложения, проектирование модульной структуры и программная реализация.

В третье главе идет апробация обучающей программы по режимам шифрования ГОСТ 28147-89.

В выпускной квалификационной работе содержится 56 страниц без приложения, 21 рисунков , 5 таблиц, 13 формул. В процессе создания было использовано 20 литературных источников.

1. ПОСТАНОВКА ЗАДАЧИ НА РАЗРАБОТКУ ОБУЧАЮЩЕЙ ПРОГРАММЫ ПО РЕЖИМАМ ШИФРОВАНИЯ ГОСТ 28147-89

1.1 Обзор криптографического преобразования ГОСТ 28147-89

Российский стандарт на криптографию с симметричным ключом определен ГОСТ 28147-89 “Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования”, который был введен в действие 1 июля 1990 г. Принципиальное отличие стандарта от DES – содержание указания на то, что он “по своим возможностям не накладывает ограничений на степень секретности защищаемой информации”. В целом эти алгоритмы схожи, однако между ними есть и существенные отличия. К ним относятся, например, длина ключа и трактовка содержимого узлов замены [которые в схеме DES называются “S-boxes”]. В то время, как внесение узлов изменения DES оптимизировано с точки зрения криптографической защищённости и явно указано в стандарте, заполнение узлов замены ГОСТ 28147 “является секретным элементом и поставляется в установленном порядке”. Учитывая, что оно вместе с тем “является долговременным ключевым элементом, общим для сети ЭВМ”, и что “установленный порядок” поставки может не предусматривать криптографическую оптимизацию, этот пункт стандарта представляется одним из его слабых мест, затрудняющим реализацию и не способствующим криптографической стойкости. В то же время при установке оптимизированных значений для узлов замены криптографическая стойкость алгоритма не сильно отличается от стойкости DES.

Стандарт криптографического преобразования данных ГОСТ 28147-89 может быть использован для защиты произвольной информации,

приведённой к двоичному коду. При разработке этого стандарта использовался мировой опыт: так, например, в процессе его формирования были приняты во внимание недостатки алгоритма DES. В виду сложности этого стандарта будет дано только его смысловое описание.

На различных этапах алгоритмов ГОСТа данные, которыми они оперируют, интерпретируются и используются по-разному. Так, элементы данных могут обрабатываться в виде массивов независимых битов; возможны обработки в виде целого числа без знака, или же в виде имеющего структуру сложного элемента, состоящего из нескольких более простых. В связи с этим следует договориться об используемых обозначениях с целью избегания неоднозначности понимания.

ГОСТ (GOST89) является прекрасным примером шифра, который был разработан, чтобы противостоять вращающийся-подраздел связанные с ключом атаки, но не связанных с ключом дифференциального криптоанализа. Пусть К0..7 быть на восемь 32-разрядных слов ключа. ГОСТ-это 32-круглый Фейстеля шифр; ключ расписание порождает вокруг подразделы k0..31 согласно

$$7 - i \bmod 8 \qquad \text{otherwize} \tag{1.1}$$

Заметим, что ненулевое ΔК0 разница заносится в только sk0, развлекательный центр sk8, sk16, и sk31; разница Δsk0 могут обрабатываться стандартным трюком взаимозачет первый раунд Ключевая разница с правильно подобранным разница открытого текста.

Такой подход позволяет обойти первые восемь раундов бесплатно.

ГОСТ по Ф-функция основывается на параллельном применении восьми 4-разрядных биективных s-блоков, затем левый поворот из 11 бит. Это означает, что это можно атаковать ГОСТ с тройной однотуровой характеристикой. Характеристика имеет только один активный s-блок.

Более подробно, пусть будет вход разница с просто один активный s-блок, а только старший бит установлен в разница входных данных, что s-поле. Выбрать B так, что A → B и младший бит активный s-блок выходной разностью, равна нулю. Функция Round вращается выходной разница влево на 11 бит, который ставит три ненулевых бит B в один информацию sbox по вход во второй тур. Этот второй раунд транслируется с B → C к подобную технику; активный s-блок выходных разница должна быть равна нулю в максимум три бита, так что мы можем покрыть третьем раунде c → A.

Если вероятность всех этих трех отношений p, И они разрешили перекрытия (что разумно, как A, B и C каждый из которых не превышает четыре бита широкий), тогда мы сможем выбрать ключевые разности, получаем 20-круглый характеристика с вероятностью p 32, и провести атаку 4г на 32-круглые ГОСТ. Этот анализ только быть практичным, когда s-коробки имеют очень плохой распределении разницы,

Против 24-круглый вариант ГОСТ, это атака становится более практичным. (A 24-круглый вариант ГОСТ должны иметь ключевой график как полный ГОСТ, но с одним из промежуточных восьмом раунде последовательности удалены. Последние восемь подразделов еще следует поменять местами с первой восемь подразделов.)

Мы исследовали безопасность “стандартный” набор ГОСТ s-блоков мы подсчитали, что 12-раундовый характеристика имеет вероятность около 2-68, который слишком мало, чтобы совершить атаку практические. Случайно выбранные s-блоки были намного слабее. Средняя вероятность более 10000 случайных s-блоков был 2-54; другие значения находились в диапазоне 2-43 по 2-80, имеющих большой разброс между испытаниями[7].

В РФ принят единый алгоритм криптографического преобразования данных, определяемый ГОСТ 28147-89. Данный алгоритм применяется в системах обработки информации, в сетях ЭВМ, отдельных вычислительных комплексах. Стандарт необходим для организаций, корпораций, фирм,

использующих криптографическую защиту информации, которая хранится и передаётся в сетях ЭВМ, а также в отдельных вычислительных комплексах.

Для данного алгоритма предполагается аппаратная и программная реализации. Он отвечает криптографическим требованиям, и в то же время подходит для любой степени секретности защищаемой информации. Алгоритм представляет собой 64-битовый блочный алгоритм с 256-битовым ключом, основанный на сети Фейстеля [1].

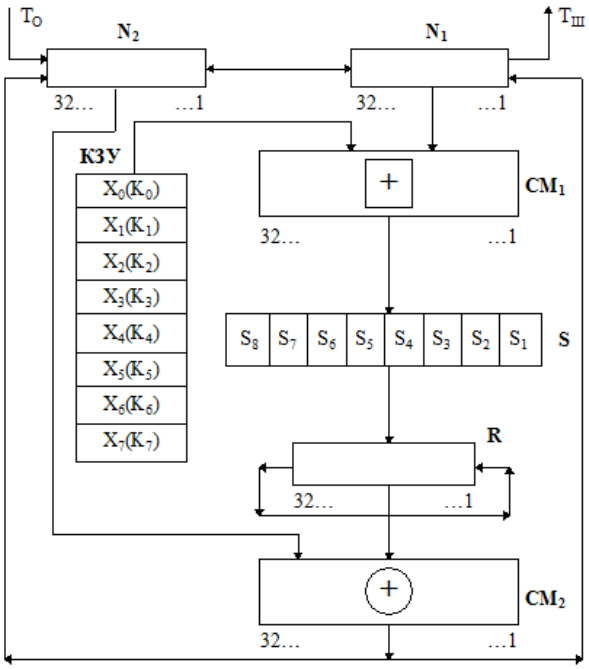

Рис. 1.1. Схема преобразования информации в алгоритме шифрования блока по ГОСТ 28147-89

На схеме используются следующие обозначения:

- N_1, N_2 – 32-разрядные накопители;
- CM_1 – 32-разрядный сумматор по модулю 2^{32} ([+]);
- CM_2 – 32-разрядный сумматор по модулю 2 ($\oplus$);
- R – 32-разрядный регистр циклического сдвига;
- КЗУ – ключевое запоминающее устройство на 256 бит, состоящее из восьми 32-разрядных накопителей X_0, X_1, X_2,…, X_7;

- S–блок подстановки, состоящий из восьми узлов замены (S-блоков замены) $S_1, S_2, S_3, \ldots, S_7, S_8$.

Открытые данные, которые необходимо зашифровать, разделяют на 64-разрядные блоки T_o . Процедура шифрования 64-разрядного блока T_o включает 32 раунда (цикла) (j = 1…32).

В ключевое запоминающее устройство вводят 256 бит ключа K в виде восьми 32-разрядных подключей K_j:

$$K = K_7 K_6 K_5 K_4 K_3 K_2 K_1 K_0 \qquad (1.2)$$

Последовательность битов в блоке

$$T_o = (a_1(0), a_2(0), \ldots, a_{31}(0), a_{32}(0), b_1(0), b_2(0), \ldots, b_{31}(0), b_{32}(0))$$

разбивается на две по 32 бита: b(0) a(0), где b(0) – левые или старшие биты, a(0) – правые или младшие биты.

Эти последовательности вводят в накопители N_1 и N_2; перед началом первого раунда шифрования.

Таблица 1.1.

В результате начальное заполнение накопителя N_1

Номер разряда	32	31	…	2	1
$a(0) = ($	$a_{32}(0),$	$a_{31}(0),$	…,	$a_2(0),$	$a_1(0))$

Таблица 1.2.

Начальное заполнение накопителя N_2

Номер разряда	32	31	…	2	1
$b(0) = ($	$b_{32}(0),$	$b_{31}(0),$	…,	$b_2(0),$	$b_1(0))$

Первый раунд ($j = 1$) процедуры шифрования 64-разрядного блока открытых данных задаётся равенствами:

$$a(1) = f(a(0)[+]K_0) \oplus b(0),\ b(1) = a(0). \qquad (1.3)$$

Где

- a(1) – заполнение N_1,
- b(1) – заполнение N_2 после 1-го раунда шифрования;
- f – функция шифрования.

функция f является суммой по модулю 2^{32} числа a(0) (начального заполнения накопителя N_1) и числа K_0 – подключа, считываемого из накопителя X_0 КЗУ. Каждое из этих чисел равно 32 битам.

Функция f состоит из двух операций, применяющихся над полученной 32-разрядной суммой $(a(0)\,[+]\,K_0)$. Первая из них – подстановка или замена – реализуется с помощью блока подстановки S. Он состоит из восьми узлов замены $S_1, S_2, \ldots, S_8$, память каждого из которых составляет 64 бита. Поступающий из CM_1 на блок подстановки S 32-разрядный вектор разбивают на восемь последовательно идущих 4-разрядных векторов, каждый из которых преобразуется в четырехразрядный вектор соответствующим узлом замены. Каждый узел замены можно представить в виде таблицы-подстановки шестнадцати четырехразрядных двоичных чисел в диапазоне 0000…1111.

В некоторых реализациях, дес ключ планирование применяется с двумя 28-разрядный сдвиговый регистры B, C и D, как в оригинальном определении. Ключ поворачивается в каждом раунде, и восстанавливается в исходное состояние по окончании шифрования, так как общее количество сдвиги в течение 16 раундов 28. Если ошибки влияют на сдвиги этих регистров, затем в следующем шифрования ключа изменяется на соответствующий ключ. Ключ криптоанализ[3], или дифференциальной, связанных с ключом криптоанализ[7] может быть применен с ДКА в таких случаях. Мы ожидаем, что линейный криптоанализ[12] также могут быть объединены с ДФА в некоторых случаях (по аналогии с дифференциально-

линейный криптоанализ[10]), особенно когда выявление неисправности установки отличается высокой надежностью (или когда позиции вина могут быть выбраны злоумышленником) [1].

Структура S-блоков в течение длительного периода времени не выкладывалась в открытом доступе. На данный момент известны S-блоки, которые Центральный Банк РФ использует в своих приложениях. Они являются относительно сильными. Их устройство рассмотрено в табл. 1.3. Любой такой блок может быть представлен в виде строки натуральных чисел от 0 до 15, расположенных в некотором порядке. В этом случае входное значение S-блока – порядковый номер числа, а само число является его выходным значением.

Таблица 1.3.

Рекомендуемые узлы замены для алгоритма ГОСТ 28147-89

№ S-блока	Значение входа															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3
2	14	11	4	12	6	13	15	10	2	3	8	1	0	7	5	9
3	5	8	1	13	10	3	4	2	14	15	12	7	6	0	9	11
4	7	13	10	1	0	8	9	15	14	4	6	12	11	2	5	3
5	6	12	7	1	5	15	13	8	4	10	9	14	0	3	11	2
6	4	11	10	0	7	2	1	13	3	6	8	5	9	12	15	14
7	13	11	4	1	3	15	5	9	0	10	14	7	6	8	2	12
8	1	15	13	0	5	7	10	4	9	2	3	14	6	11	8	12

Второй операцией является циклический сдвиг влево (на 11 разрядов) 32-разрядного вектора, полученного с выхода блока подстановки S. Циклический сдвиг выполняется регистром сдвига R.

Последующие раунды осуществляются аналогично, при этом во втором раунде из КЗУ считывают заполнение X_1 – подключ K_1, в третьем – подключ K_2 и т.д., в восьмом раунде – подключ K_7. В раундах с 9-го по 16-

й, а также в с 17-го по 24-й подключи из КЗУ считываются в том же порядке: $K_0, K_1, K_2, \ldots, K_6, K_7$. В последних восьми раундах с 25-го по 32-й порядок считывания подключей из КЗУ обратный: $K_7, K_6, K_5, \ldots, K_1, K_0$. Следовательно, при шифровании в 32 цикла порядок выборки из КЗУ подключей будет выглядеть так:

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7,$$
$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

В 32-м раунде результат из сумматора CM_2 вводится в накопитель N_2, а в накопителе N_1 сохраняется прежнее заполнение. Полученные после 32-го раунда заполнения накопителей N_1 и N_2 являются блоком зашифрованных данных $T_ш$, соответствующим блоку открытых данных T_o.

Таким образом уравнения шифрования блока имеют вид:

$$\begin{cases} a(j) = f(a(j-1)\,[+]\,K_{j-1(\mathrm{mod}\,8)}) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \text{при j = 1…24,}$$

$$\begin{cases} a(j) = f(a(j-1)\,[+]\,K_{32-j}) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \text{при j = 25…31,} \qquad (1.4)$$

$$\begin{cases} a(32) = a(31) \\ b(32) = f(a(31)\,[+]\,K_0) \oplus b(31) \end{cases} \quad \text{при j = 32,}$$

где $a(j) = (a_{32}(j), a_{31}(j), \ldots, a_1(j))$ – заполнение N_1 после j-го раунда шифрования, $b(j) = (b_{32}(j), b_{31}(j), \ldots, b_1(j))$ – заполнение N_2 после j-го раунда, j = 1…32.

Блок зашифрованных данных $T_ш$ (64 разряда) выводится из накопителей N_1, N_2 в следующем порядке: из разрядов 1…32 накопителя N_1 , затем из разрядов 1…32 накопителя N_2, т.е. начиная с младших разрядов:

$$T_ш = (a_1(32), a_2(32), \ldots, a_{31}(32), a_{32}(32), b_1(32), b_2(32), \ldots, b_{31}(32), b_{32}(32)). \quad (1.5)$$

В КЗУ вводят 256 бит ключа, на котором осуществлялось шифрование. Зашифрованные данные, подлежащие расшифрованию, разбиты на блоки $T_ш$ по 64 бита в каждом. Ввод любого блока $T_ш$ в накопители N_1 и N_2 производят так, чтобы начальное значение накопителя N_1, имело вид:

Таблица 1.4.

Начальное заполнение накопителя N_1

Номер разряда	32	31	…	2	1
a(32) = (	$a_{32}(32)$,	$a_{31}(32)$,	…,	$a_2(32)$,	$a_1(32)$)

Таблица 1.5.

Начальное заполнение накопителя N_2

Номер разряда	32	31	…	2	1
b(32) = (	$b_{32}(32)$,	$b_{31}(32)$,	…,	$b_2(32)$,	$b_1(32)$)

Расшифрование осуществляется по тому же алгоритму, что и шифрование, с тем изменением, что заполнения накопителей $X_0, X_1, X_2, \ldots, X_7$ считываются из КЗУ в раундах расшифрования в следующем порядке:

$$K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0,$$
$$K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0, K_7, K_6, K_5, K_4, K_3, K_2, K_1, K_0.$$

Уравнения расшифрования имеют вид:

$$\begin{cases} a(32-j) = f\left(a(32-j+1)\,[+]\,K_{j-1}\right) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad \text{при j=1…8,}$$

$$\begin{cases} a(32-j) = f\left(a(32-j+1)\,[+]\,K_{32-j(\mathrm{mod}8)}\right) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad \text{при j=9…31,}$$

$$\begin{cases} a(0) = a(1) \\ b(0) = f\left(a(1)\,[+]\,K_0\right) \oplus b(1) \end{cases} \quad \text{при j=32.} \quad (1.6)$$

Полученные результаты после 32 циклов работы заполнения накопителей N_1 и N_2 образуют блок открытых данных

$$T_o = \left(a_1(0), a_2(0), \ldots, a_{31}(0), a_{32}(0), b_1(0), b_2(0), \ldots, b_{31}(0), b_{32}(0)\right) \quad (1.7)$$

На рис. 1.2 изображена схема режима электронной кодовой книги.

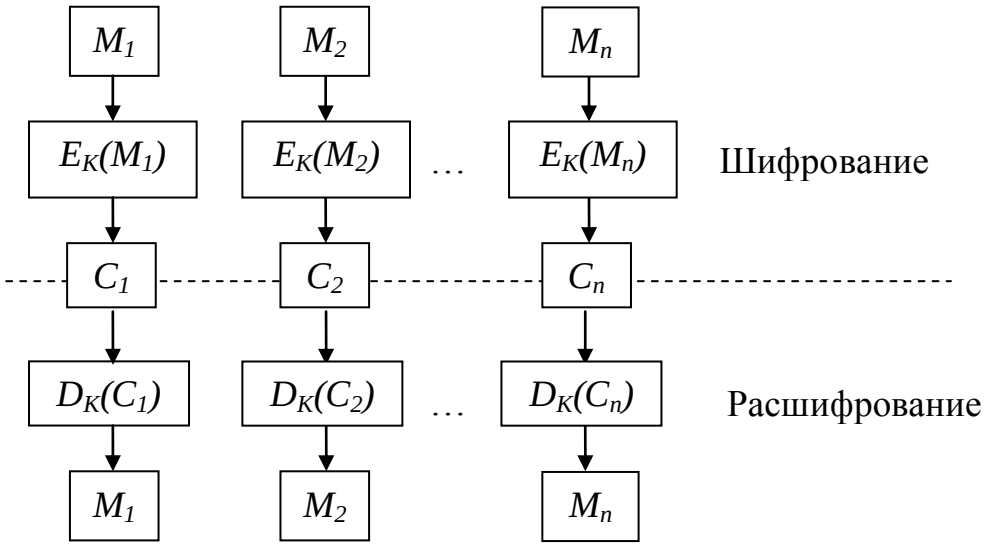

Рис. 1.2. Схема режима ECB

Уравнение шифрования данных в режиме гаммирования выглядит следующим образом:

$$\text{Тш}^{(i)} = \text{T}_0^{(i)} \oplus \text{Гш}^{(i)}, \text{ где } \text{Гш}^{(i)} = \text{A}(\text{Y}_{i-1}\,[+]\,\text{C}_2, \text{Z}_{i-1}\,[+]'\,\text{C}_1), \text{ i=1…m;} \quad (1.8)$$

где:

- $Тш^{(i)}$ – 1-й блок 64-разрядного блока зашифрованного текста;
- $A(\cdot)$ – функция шифрования блока;
- [+] – операция суммы по модулю 2^{32},
- [+]' – операция суммы по модулю 2^{32-1};
- C_1, C_2 – 32-разрядные двоичные константы;
- Y_i , Z_i – 32-разрядные двоичные последовательности.

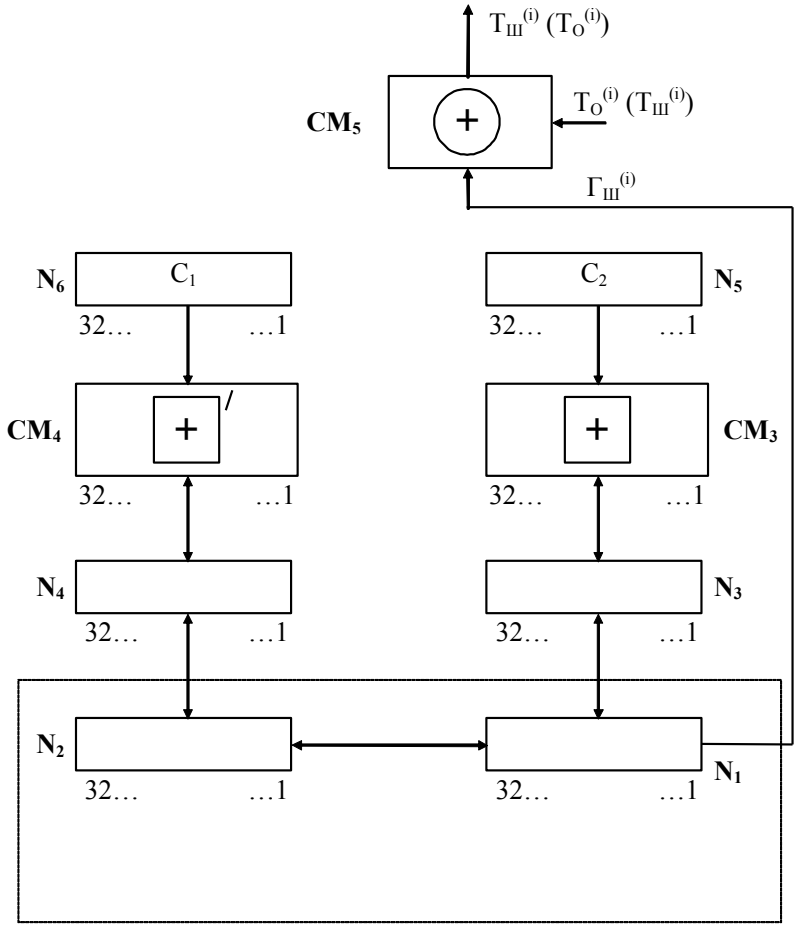

Рис. 1.3.Схема реализации режима гаммирования ГОСТ 28147-89

Разберём реализацию процедуры шифрования в режиме гаммирования. В накопители N_6 и N_5 заранее записаны 32-разрядные двоичные константы C_1 и C_2, имеющие следующие значения (в шестнадцатеричной форме):

$$C_1 = 01010104_{(16)} , C_2 = 01010101_{(16)}. \quad (1.9)$$

В КЗУ вводится 256 бит ключа; в накопители N_1 и N_2 – 64-разрядная двоичная последовательность (синхропосылка) $\check{S} = (S_1, S_2,\ldots, S_{64})$.

Синхропосылка $\check{S}$ является исходным заполнением накопителей N_1 и N_2 для последовательной выработки m блоков гаммы шифра. Исходное заполнение N_1 и N_2 зашифровывается. Результат шифрования $A(\check{S})=(Y_0, Z_0)$ переписывается в 32-разрядные накопители N_3 и N_4 так, что заполнение N_1 переписывается в N_3, а заполнение N_2 – в N_4.

Заполнение накопителя N_4 суммируют по модулю $(2^{32}–1)$ в сумматоре $СМ_4$ с 32-разрядной константой C_1 из накопителя N_6. Результат записывается в N_4. Заполнение накопителя N_3 суммируется по модулю 2^{32} в сумматоре $СМ_3$ с 32-разрядной константой С2 из накопителя N_5. Результат записывается в N_3. Заполнение N_3 переписывают в N_1, а заполнение N_4 – в N_2, при этом заполнения N_3, N_4 сохраняются. Заполнение накопителей N_1 и N_2 зашифровывается.

Полученное в результате шифрования заполнение накопителей N_1, N_2 образует первый 64-разрядный блок гаммы шифра $Гш^{(1)} = (\gamma_1^{(1)}, \gamma_2^{(1)},\ldots, \gamma_{63}^{(1)}, \gamma_{64}^{(1)})$, который суммируют поразрядно по модулю 2 в сумматоре $СМ_5$ с первым 64-разрядным блоком открытых данных $T_0^{(1)} = (t_1^{(1)}, t_2^{(1)},\ldots, t_{63}^{(1)}, t_{64}^{(1)})$.

В результате суммирования по модулю 2 значений $Гш^{(1)}$ и $T_0^{(1)}$ получают первый 64-разрядный блок зашифрованных данных:

$$Тш^{(1)} = Гш^{(1)} \oplus T_0^{(1)} = (\tau_1^{(1)}, \tau_2^{(1)},\ldots, \tau_{63}^{(1)}, \tau_{64}^{(1)}), \qquad (1.10)$$

где $\tau_i^{(1)} = t_i^{(1)} \oplus \gamma_i^{(1)}$, i=1…64.

Для получения следующего 64-разрядного блока гаммы шифра $Гш^{(2)}$ заполнение N_4 суммируется по модулю $(2^{32}–1)$ в сумматоре $СМ_4$ с константой C_1 из N_6. Результат записывается в N_4. Заполнение N_3 суммируется по

модулю 2^{32} в сумматоре $СМ_3$ с константой $С_2$ из N_5. Результат записывается в N_3. Новое заполнение N_3 переписывают в N_1, а новое заполнение N_4 – в N_2, при этом заполнения N_3 и N_4 сохраняют. Заполнения N_1, N_2 зашифровывают.

Полученное в результате зашифрования заполнение накопителей N_1 и N_2 образует второй 64-разрядный блок гаммы шифра $Гш^{(2)}$, который суммируется поразрядно по модулю 2 в сумматоре $СМ_5$ со вторым блоком открытых данных $Т_0^{(2)}$: $Тш^{(2)} = Гш^{(2)} \oplus Т_0^{(2)}$.

Аналогично вырабатываются блоки гаммы шифра $Гш^{(3)}$, $Гш^{(4)}$,…, $Гш^{(m)}$ и зашифровываются блоки открытых данных $Т_0^{(3)}$, $Т_0^{(4)}$,… $Т_0^{(m)}$.

В канал связи или память ЭВМ передаются синхропосылка Š и блоки зашифрованных данных $Тш^{(1)}$, $Тш^{(2)}$,… $Тш^{(m)}$.

При расшифровании криптосхема имеет тот же вид, что и при шифровании. Уравнение расшифрования:

$$Т_0^{(i)} = Тш^{(i)} \oplus Гш^{(i)} = Тш^{(i)} \oplus A(Y_{i-1} [+] C_2, Z_{i-1} [+]' C_1), i=1…m. \quad (1.11)$$

Следует отметить, что расшифровка существует только в том случае, когда имеется синхропосылка, не являющаяся секретным элементом шифра. Она может храниться в памяти ЭВМ или передаваться по каналам связи вместе с зашифрованными данными.

Рассмотрим реализацию процедуры расшифрования. В КЗУ вводят 256 бит ключа, с помощью которого осуществляется зашифрование данных $Т_0^{(1)}$, $Т_0^{(2)}$,… $Т_0^{(m)}$. В накопители N_1 и N_2 вводится синхропосылка, и осуществляется процесс выработки m блоков гаммы шифра $Гш^{(1)}$, $Гш^{(2)}$,…, $Гш^{(m)}$. Блоки зашифрованных данных $Тш^{(1)}$, $Тш^{(2)}$,… $Тш^{(m)}$.суммируются поразрядно по модулю 2 в сумматоре $СМ_5$ с блоками гаммы шифра $Гш^{(1)}$, $Гш^{(2)}$,…, $Гш^{(m)}$. В результате получаются блоки открытых данных $Т_0^{(1)}$, $Т_0^{(2)}$,… $Т_0^{(m)}$. При этом $Т_0^{(m)}$ может содержать меньше 64 разрядов.

Шифрование данных в режиме гаммирования с обратной связью соответствует режиму CFB.

Вид криптосхемы, реализующей алгоритм шифрования в режиме гаммирования с обратной связью, представлен на рис. 1.4.

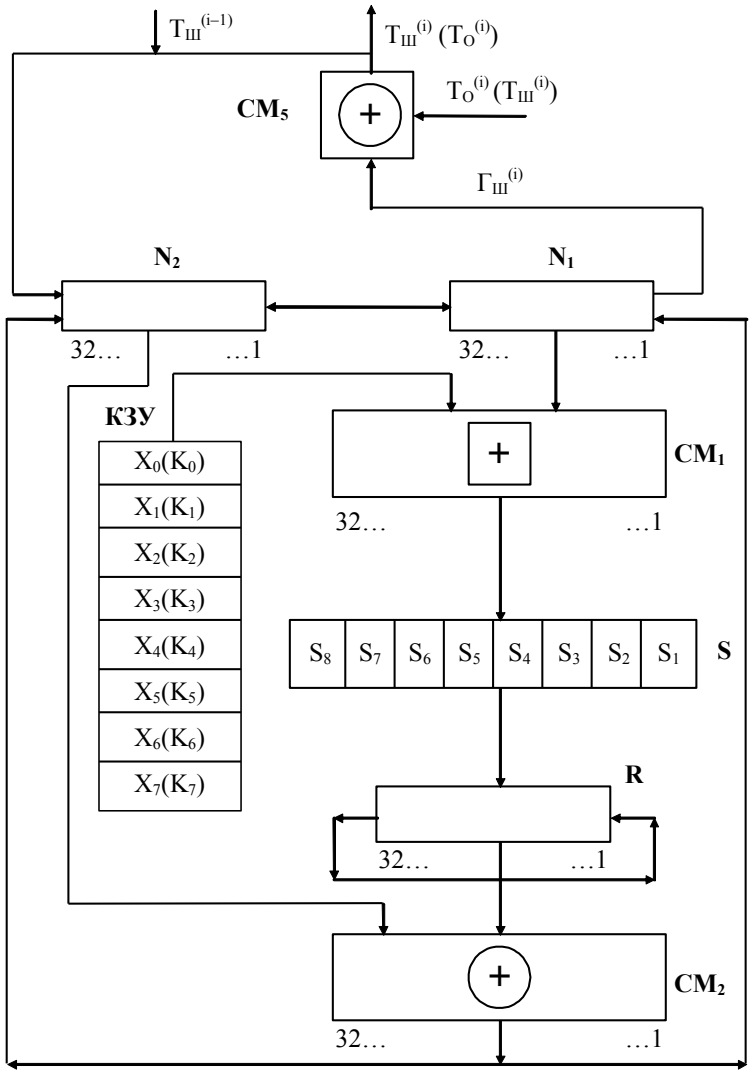

Рис.1.4. Схема реализации режима гаммирования с обратной связью ГОСТ 28147-89

Открытые данные, разбитые на 64-разрядные блоки $T_0^{(1)}$, $T_0^{(2)}$,…, $T_0^{(m)}$, зашифровываются в режиме гаммирования с обратной связью путем поразрядного сложения по модулю 2 с гаммой шифра Гш, которая вырабатывается блоками по 64 бита:

$$Гш=(Гш^{(1)}, Гш^{(2)},\ldots, Гш^{(i)},\ldots, Гш^{(m)}). \quad (1.12)$$

При расшифровании криптосхема имеет тот же вид, что и при шифровании. Уравнения расшифрования:

$$T_0^{(1)} = A(Š) \oplus Тш^{(1)} = Гш^{(1)} \oplus Тш^{(1)},$$
$$T_0^{(i)} = A(Тш^{(i-1)}) \oplus Тш^{(i)} = Гш^{(i)} \oplus Тш^{(i)}, \; i=2\ldots m. \quad (1.13)$$

Для блоков гаммы шифровании производится поразрядное сложение по модулю 2 в сумматоре $СМ_5$ с блоками зашифрованных данных $Тш^{(3)}$, $Тш^{(4)},\ldots, Тш^{(m)}$. В результате получают блоки открытых данных $T_0^{(3)}$, $T_0^{(4)},\ldots, T_0^{(m)}$, при этом последний блок открытых данных $T_0^{(m)}$ может включать меньше 64 разрядов.

Далее применяют сложение по модулю 2 полученного 64-разрядного числа $A(T_0^{(1)})$ со вторым блоком открытых данных $T_0^{(2)}$.

Сумма $(A(T_0^{(1)}) \oplus T_0^{(2)})$ вновь подвергается преобразованию $A(\cdot)$.

Имитовставка $И_Р$ передается по каналу связи или в память ЭВМ в конце зашифрованных данных, т.е. $Тш^{(1)}, Тш^{(2)},\ldots, Тш^{(m)}, И_Р$.

Поступившие к получателю зашифрованные данные $Тш^{(1)}, Тш^{(2)},\ldots, Тш^{(m)}$ расшифровываются, и из полученных блоков открытых данных $T_0^{(1)}$, $T_0^{(2)},\ldots, T_0^{(m)}$ аналогичным образом вырабатывается имитовставка $И_Р$. Эта имитовставка $И_Р$ сравнивается с имитовставкой $И_Р$, полученной вместе с зашифрованными данными из канала связи или из памяти ЭВМ. В случае несовпадения имитовставок полученные при расшифровании блоки открытых данных $T_0^{(1)}, T_0^{(2)},\ldots, T_0^{(m)}$ считают ложными [1].

1.2 Обзор и анализ обучающих программ

Когда мы говорим об обучающих или учебных программах, то имеем в виду программные продукты, специально разработанные для использования в целях обучения независимо от учебного предмета, объёма изучаемого материала, структуры и других показателей как технического, так и методического планов, - и создающих систему учебных компьютерных заданий. К таким программам относятся различные виды компьютерных дидактических материалов: электронные учебники, автоматизированные учебные курсы, тренировочные, контролирующие программы, автоматизированные обучающие системы. Именно наличие системы заданий отличает компьютерные учебные пособия и обучающие программы от разнообразных прикладных программ, использующихся в обучении (например, редакторы текстов, электронные таблицы и т. д.). Обучающие программы и компьютерные учебные пособия занимают центральное место в программном обеспечении той или иной дисциплины.

- функциональность (functionality);
- надёжность (reliability);
- лёгкость и простота использования (usability);
- эффективность (efficiency);
- удобство сопровождения (maintainability);
- переносимость (portability).

Достоинство компьютерных программ учебного назначения в том, что учебный материал – хорошо иллюстрирован, мобилен, вариативен; программы должны помогать преподавателю контролировать и регулировать индивидуальный процесс усвоения, учитывать различные уровни подготовленности обучающихся, в большей степени концентрировать внимание на изучаемом материале.

Криптографический метод защиты, является одним из самых надежных методов защиты, так как защищена сама информация, а не доступ к ней.

Зашифрованный файл нельзя прочесть даже в случае кражи носителя. Данный метод защиты реализуется в виде программ или пакетов программ.

В настоящее время существует множество программ и обучающих программ для защиты и криптографического преобразования информации.

Обучающие программы важны, чтобы студенты, обучающиеся в сфере информационных технологий, могли освоить методы защиты информации, алгоритмы криптографического преобразования, для использования программ так и написания своих собственных.

Основным назначением использования криптографических методов является передача конфиденциальной информации по каналам связи установление подлинности передаваемых сообщений, хранение информации (документов, баз данных) на носителях в зашифрованном виде.

Нами была проанализирована программа шифрования DES. Основными достоинствами алгоритма DES является использование только одного ключа длиной 56 бит. Зашифровав сообщение с помощью одного пакета, для расшифровывания мы можем использовать любой другой, также алгоритм относительно прост и обеспечивает высокую скорость обработки информации, достаточно высокая устойчивость алгоритма.

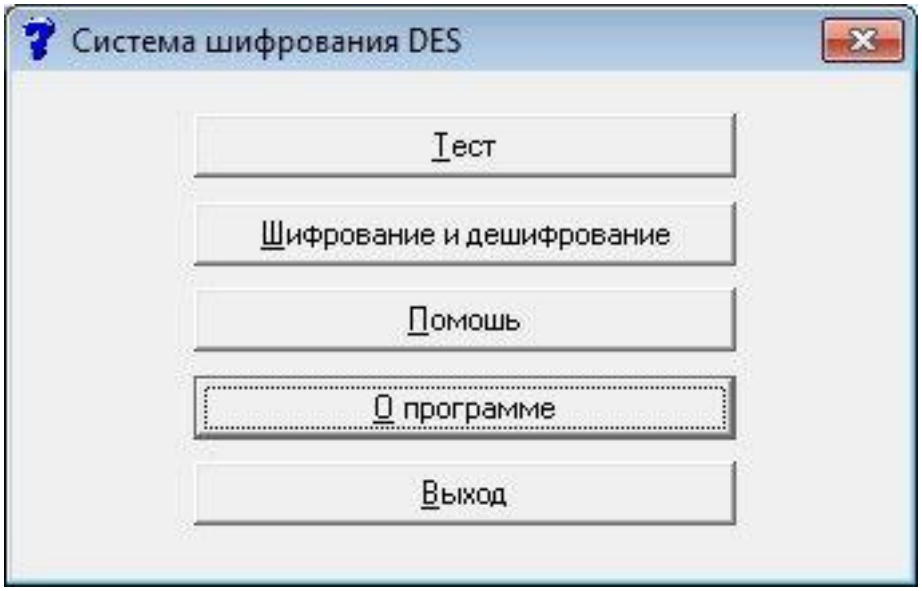

Рис. 1.5. Основное окно программы DES

DES осуществляет шифрование 64-битовых блоков данных с помощью 56-битового ключа, а расшифровывается DES, операциями обратного шифрования и реализуется повторяя операции шифрования в обратной последовательности.

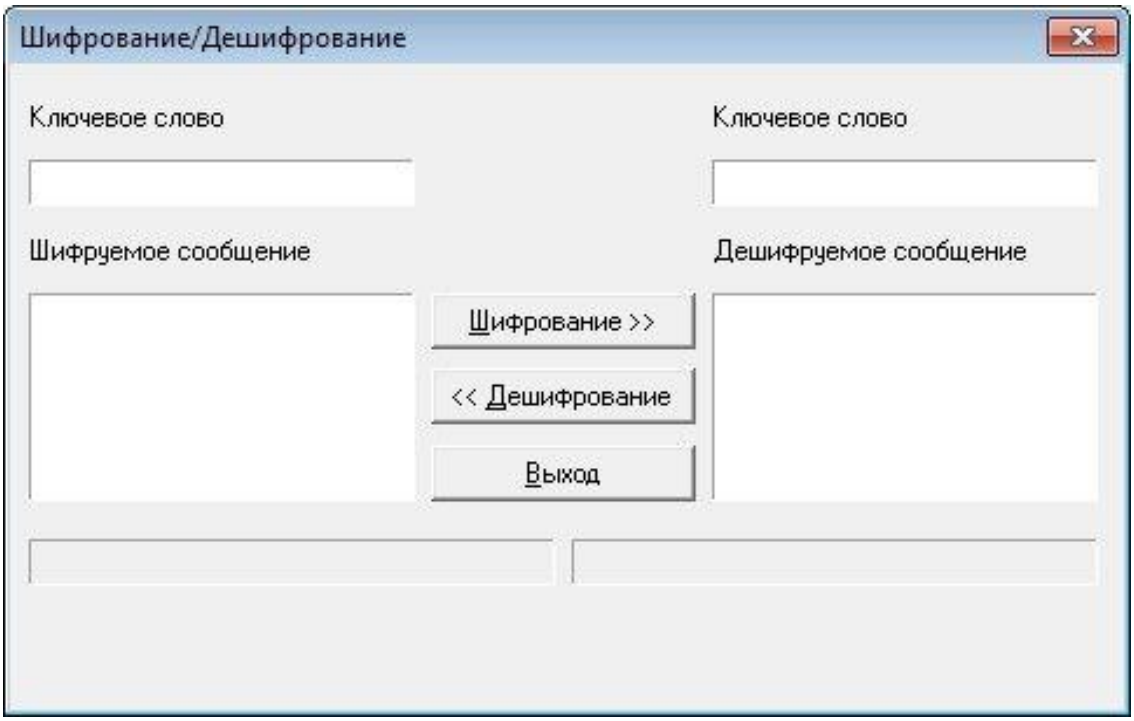

Рис. 1.6. Шифрование и дешифрование

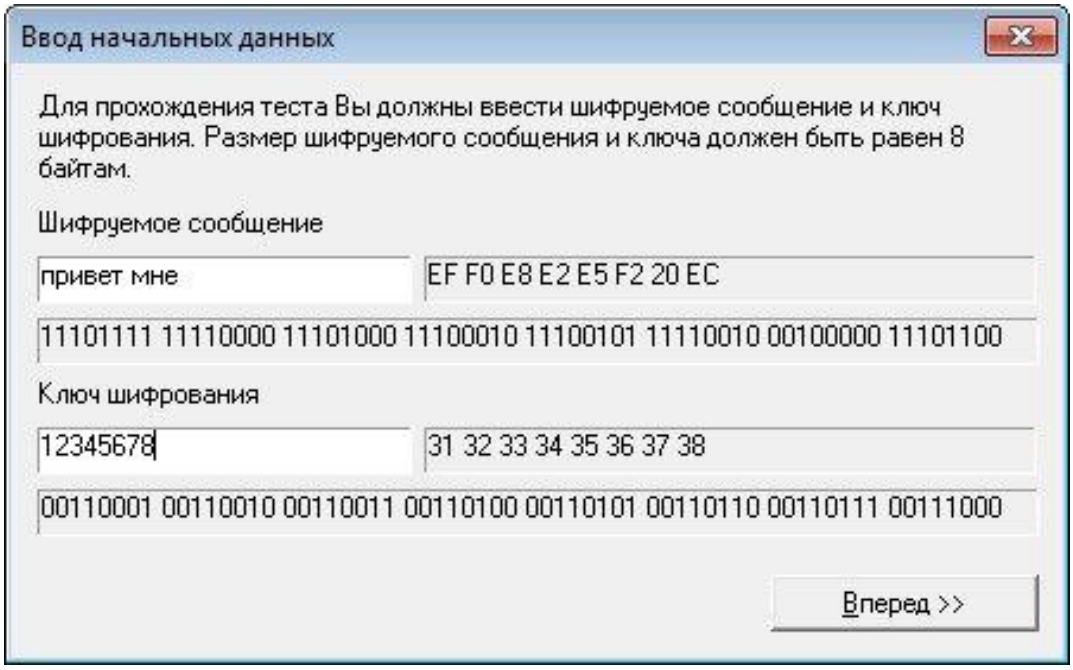

Рис. 1.7 . Ввод исходных данных в тесте

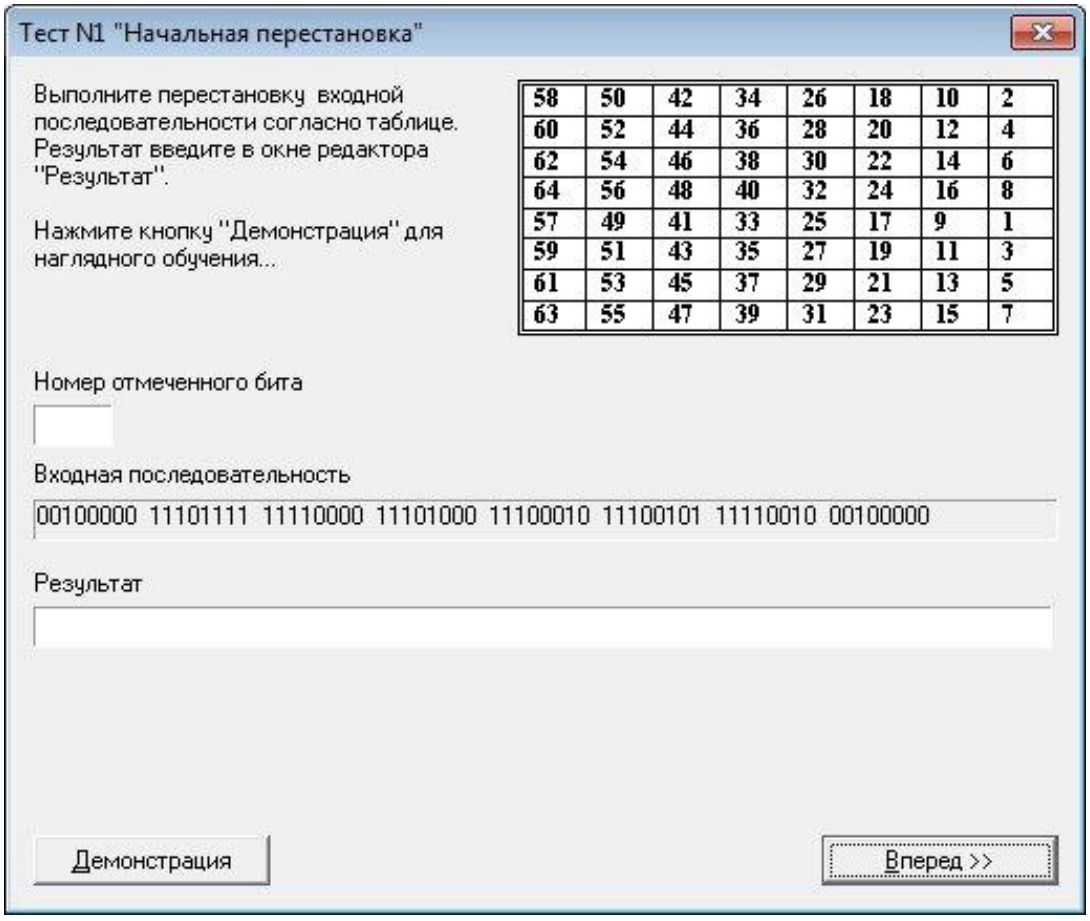

Рис.1.8. Пример теста

Программа DES имеет режим теста, в котором для прохождения теста пользователю необходимо ввести шифруемое сообщение и ключ шифрования. Размер шифруемого сообщения и ключа равен 8 битам. В поле ввода пользователь вносит шифруемое сообщение и ключ, после чего на форме выводится зашифрованный результат. После прохождения теста может вывестись сообщение об ошибке, с просьбой ввести длину шифрования сообщения и ключа шифрования 8 бит. В режиме шифрования и дешифрования, пользователь вводит ключевое слово и сообщение для шифровки или дешифровки. Действия шифрования и дешифрования происходят по соответствующим кнопкам. Результат выводится на форме. Кнопка выход возвращает пользователя в меню.

Кнопка «Помощь» открывает пользователю справку, где можно подробнее изучить функции шифрования в программе.

Таким образом, программа даёт представление пользователю о работе шифрования DES. Наглядно показывает, как происходит шифрование, дешифрование, ввод ключа. К минусам можно отнести то, что программа не достаточно подробно описывает пользователю работу шифрования и дешифрования, имеет мало теоретических данных и не описывает все действия программы.

Мы можем сделать вывод, что необходимо подробнее сделать указания пользователю при шифровании и дешифровании. Более подробно описать функции программы и режима шифрования, сделать более понятный интерфейс и наличие теоретического материала.

В ходе создания данной выпускной квалификационной работы, нами было определено, что защита информации является важным аспектом современных информационных технологий. Программы в области криптографического преобразования как специальных методов шифрования или других видов преобразования информации, используются для защиты информации. В результате преобразования, содержание информации становится недоступным без предъявления ключа криптограммы и обратного преобразования.

Рассмотрим ещё одну программу реализующую обучения по режимам шифрования ГОСТ. Программа разработана для обучения пользователей программы, работе с режимами шифрования. Главное окно представлено на рисунке.

Пользователь может выбрать один из режимов шифрования и перейти к обучению. Режим обучения шифрованию простой замены представлен на рисунке 1.10.

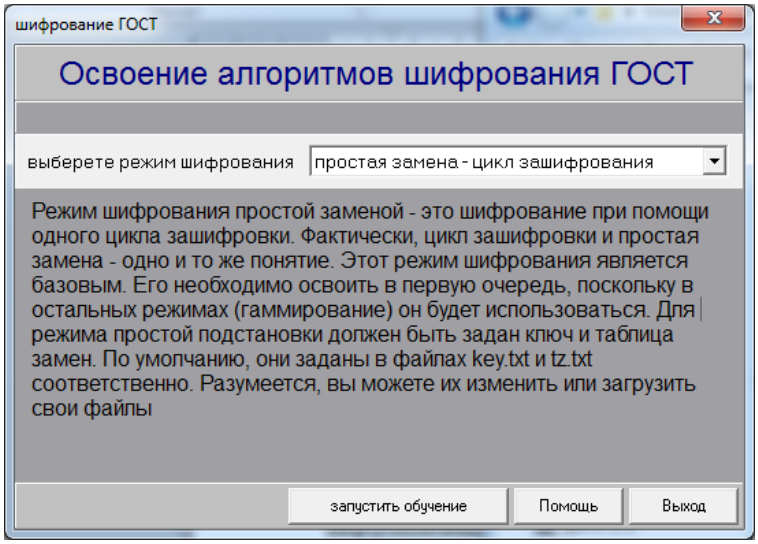

Рис. 1.9.Меню программы

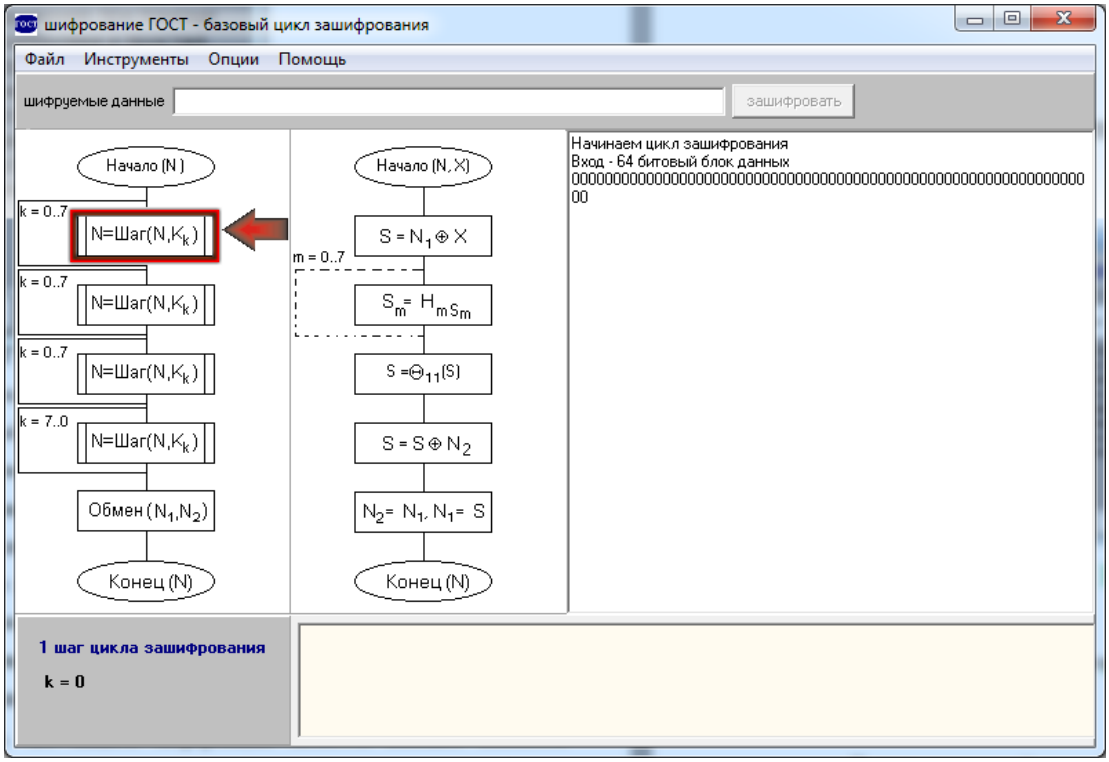

Рис. 1.10. Режим простой замены

Программа реализует так же и режим гаммирования. Он аналогично предыдущему, предоставляет пользователю возможности обучения данному режиму. Шифрование при помощи наложения на данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью криптографического алгоритма, помогает избавится от недостатка

режима простой замены - одни и те же 64 битовые блоки при простой замене будут зашифровываться одинаково.

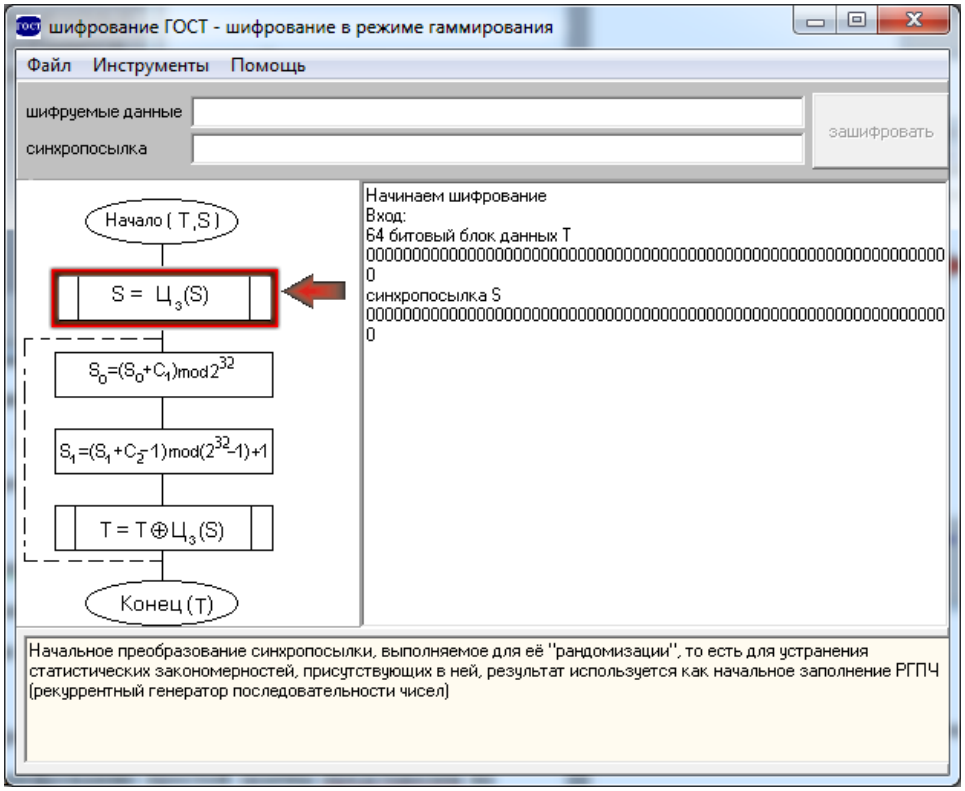

Рис. 1.11. Режим гаммирования

Режим иммитовставки, инициализируется нулевыми битами. Далее, из блока S получится искомая иммитовставка. Для поиска искажений искажений в зашифрованных данных в ГОСТе предусмотрен дополнительный режим криптографического преобразования - выработка имитовставки. Имитовставка - это контрольная комбинация, зависящая от открытых данных и секретной ключевой информации. В программе имеется описание метода и наглядно показана его реализация.

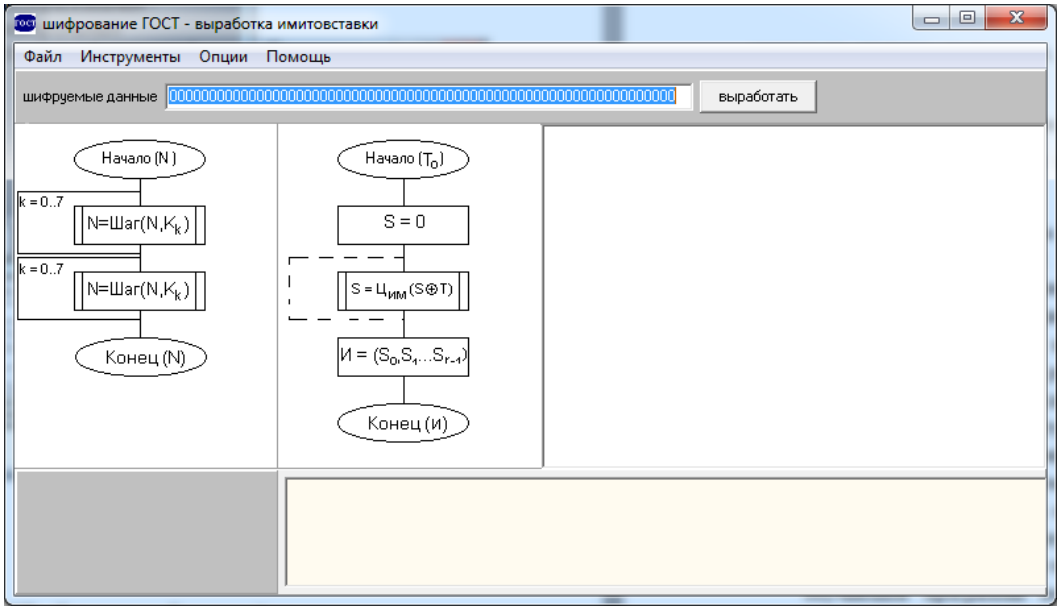

Рис.1.12 Выработка имитовставки

1.3 Требования к обучающей программы

В ходе выпускной квалификационной работы нами были выделены требования к функциональным характеристикам. Программа ГОСТ должна иметь понятный и удобный интерфейс, содержать материал для изучение пользователем. Зашифрованное сообщение должно поддаваться чтению только при наличии ключа. Знание алгоритма шифрования не должно влиять на надежность защиты. Структурные элементы алгоритма шифрования должны быть неизменными. Длина шифрованного текста должна быть равной длине исходного текста. Алгоритм должен допускать как программную, так и аппаратную реализацию, при этом изменение длины ключа не должно вести к качественному ухудшению алгоритма шифрования.

Материал программы должен быть направлен не только на запоминание, но и понимание учебного материала. Программа должна учитывать уровень подготовки, возраст и индивидуальные особенности обучаемых.

Программы должны вести учёт работы каждого обучаемого и давать рекомендации по оптимальной организации его работы в той или иной области знаний.

Разработаем требования на создание обучающей системы по режимам шифрования ГОСТ 28147-89. Система предназначена для освоения студентами режимов шифрования.

Введение технического задания

Настоящее техническое задание распространяется на разработку обучающей программы по режимам шифрования ГОСТ 28147-89, предназначенной обучения студентов. Предполагается, что использовать данную систему будут студент обучающиеся на технических специальностях, связанными с информационными технологиями.

Во время изучения темы криптографического кодирования и шифрования либо подготовки к сессии необходимо получить практические навыки по данным темам. К тому же необходимо иметь удобную среду для изучения практических аспектов темы криптографии и закрепления навыков обучающей программы по режимам шифрования ГОСТ 28147-89 позволит улучшить качество обучения студентов по теме шифрования и криптографии, подходит как для работы в учебной аудитории, так и для самостоятельной работы.

Назначение разработки

Система предназначена для обучения студентов, по режимам шифрования ГОСТ 28147-89. Программа может быть использована для обучения как одного студента, так и группы в целом.

Требования к функциональным характеристикам

Система должна обеспечивать возможность выполнения следующих функций:

- инициализацию системы (материал программы должен соответствовать материалу изучаемых дисциплин в соответствии с учебными планами и т. п.);

- ввод и коррекцию текущей информации содержащейся в программе, возможность дополнения программы;
- хранение информации ключей исходных данных и зашифрованного файла;
- получение сведений об изучаемой теме, помощь работы в программе.

Исходными данными являются:

- файл ключа;
- исходный файл для шифрования сообщения, содержащий массив данных;
- режимы шифрования.

Результаты:

- файл с зашифрованным текстом;
- результаты вычислений программы по режимам шифрования.

Требования к надежности

Требования к обеспечению надежного функционирования программы:

- предусмотреть контроль вводимой информации;
- предусмотреть блокировку некорректных действий пользователя при работе с системой;
- обеспечить целостность хранимой информации.

Требования к составу и параметрам технических средств

Система должна работать на IBM совместимых персональных компьютерах.

Минимальная конфигурация:

- тип процессора – AMD и выше;
- объем оперативного запоминающего устройства – 1024 Мб и более.

Требования к информационной и программной совместимости

Система должна работать под управлением семейства операционных систем Windows.

Требования к программной документации.

Разрабатываемые программные модули должны быть самодокументированы, т. е. тексты программ должны содержать все необходимые комментарии.

Программная система должна включать справочную информацию о работе и подсказки пользователю.

В состав сопровождающей документации должны входить:

- пояснительная записка на 25-30 листах, содержащая описание разработки;
- руководство системного программиста;
- руководство пользователя.

2. ПРОЕКТИРОВАНИЕ И ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

2.1 Предварительное проектные решения

Средствами создания программных приложений являются -локальные средства, обеспечивающие выполнение различных видов работ по созданию программ. Средствами могут быть - языки и системы программирования и инструментальная среда пользователя.

Была реализована обучающая программа ГОСТ, разработанная в среде визуального. Объектно-ориентированного обучения. В этой среде наиболее удобно реализовать необходимые нам функции – шифрование, переход между формами, запись в файл, функция обучения.

В качестве языка программирования для реализации данной выпускной квалификационной работы нами был выбран язык программирования C++ — компилируемый статически типизированный язык программирования общего назначения. Именно этот язык наиболее подходит для автоматизации информационной системы, и реализации режимов шифрования ГОСТ, так как C++ успешно используется во многих областях приложения, далеко выходящих за указанные рамки и позволяет создавать как простые приложения и утилиты, так и коммерческие системы.

Стоит отметить, что язык программирования C++, позволяет более эффективно программировать на уровне битов и шифровать биты. Имеет размер для ключа 256 бит. К примеру, для реализации web приложения данные функции не удобны, так как нужно иметь постоянный доступ к серверу, где хранятся данные для шифрования. Шифрование может быть не стабильно и имеет меньшую скорость, затраты на такое приложение могут быть выше. Для обучения студентов вариант приложения на языке

программирования C++ более удобен, т.к. проще администрировать, не нужно иметь доступ к серверу.

Язык C++ поддерживает как процедурное программирование, так и объектно-ориентированное программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так и низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования.

C++ широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов, а также развлекательных приложений (игр). Существует множество реализаций языка C++, как бесплатных, так и коммерческих и для различных платформ. Например, на платформе x86 это GCC, Visual C++, Intel C++ Compiler, Embarcadero (Borland) C++ Builder и другие. Синтаксис C++ унаследован от языка C. Одним из принципов разработки было сохранение совместимости с C. Тем не менее, C++ не является в строгом смысле надмножеством C; множество программ, которые могут одинаково успешно транслироваться как компиляторами C, так и компиляторами C++, довольно велико, но не включает все возможные программы на C.

Также C++ — чрезвычайно мощный язык, содержащий средства создания эффективных программ практически любого назначения, от низкоуровневых утилит и драйверов до сложных программных комплексов самого различного назначения.

В процессе рассмотрения языков программирования и программных средств, нами была выбрана среда Borland C++ Builder 6. Было решено разрабатывать визуальную программу. Среда имеет удобный интерфейс, части одного приложения могут быть созданы с помощью двух средств C++BuiIder, и Delphi. C++Builder предоставляет программисту широкие возможности повторного использования кода за счет наличия библиотеки компонентов. Эффективность реализации и отладки приложений и программ достигается не только за счет применения средств визуального проектирования, но и за счет, во-первых, высокой производительности самих компиляторов Borland и, во-вторых, так называемой инкрементной компиляции и компоновки исполняемого модуля, когда перекомпиляции и перекомпоновке подвергаются только те модули, в которые были внесены изменения.

2.2. Проектирование модульной структуры программы

При проектировании обучающей программы, были выделены модули программы. Программа имеет следующие модули. Главное меню программы, откуда мы будем переходить к режимам шифрования и страницы с помощью пользователю, режимы простой замены, гаммирования, гаммирования с обратной связью, выработки имитовставки. Данные модули были выбраны исходя из целей и требований к разработке обучающей программы по режимам шифрования. Структура модулей программы представлена виде структурной схемы Константайна (рис. 2.1). С формы модулей режимов пользователь возвращается на главную страницу к главному меню программы.

Нами были разработаны отдельные модули программы. Главное окно, является основным для перехода к режимам шифрования программы и

другим модулям. По нажатию кнопки режима, главное окно формы закрывается и открывается новое окно.

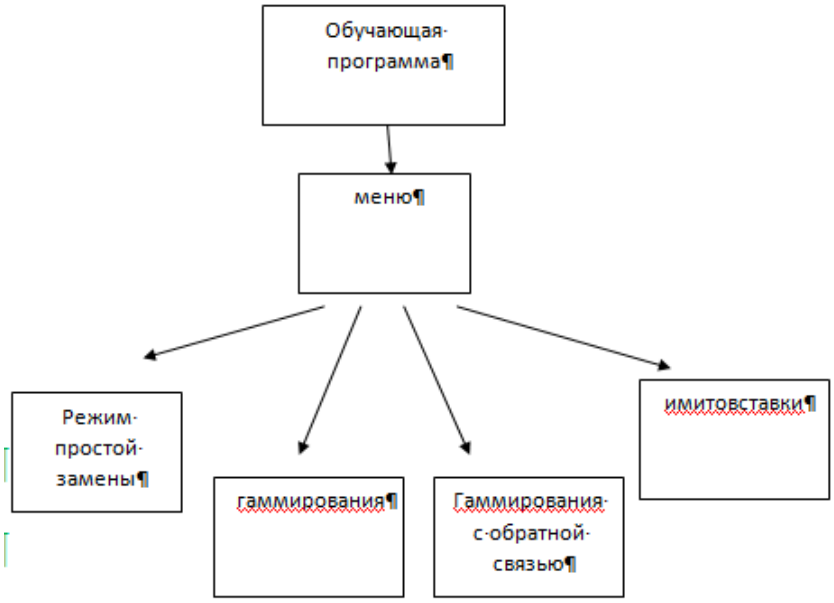

Рис. 2.1. Структурная схема Константайна

Рассмотрим блок-схемы отдельных модулей, алгоритм зашифрования данных в режиме простой замены представлен на рисунке 2.2

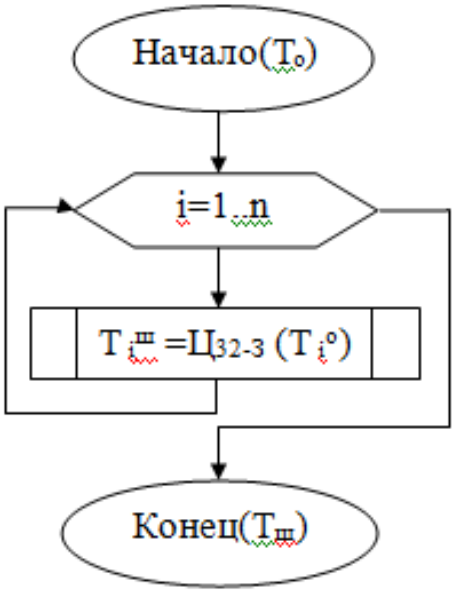

Рис. 2.2 Режим простой замены

Отдельный модуль реализующий зашифрование и расшифрование методом гаммирования, модель метода реализует блок-схема представлен на рисунке 2.3.

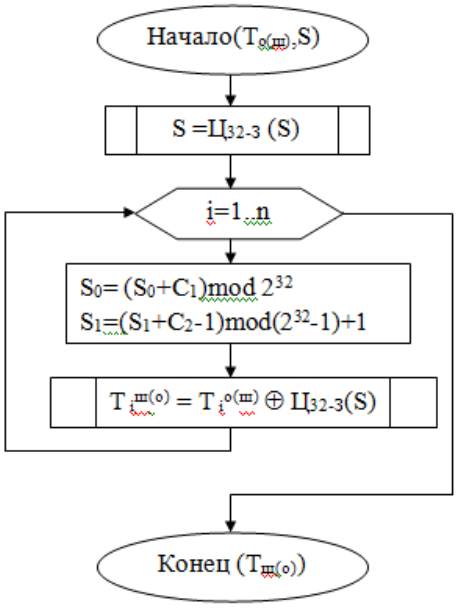

Рис. 2.3 Режим гаммирования

Блок-схема реализации шифрования методом гаммирования с обратной связью представлена на рисунке 2.4.

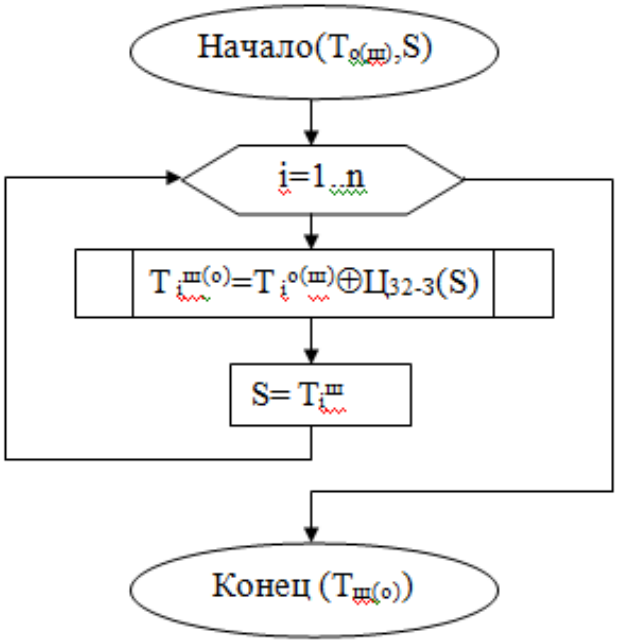

Рис. 2.4 Режим гаммирования с обраной связью

Следующий модуль выработки имитовставки представлен на блок-схеме (рис. 2.5).

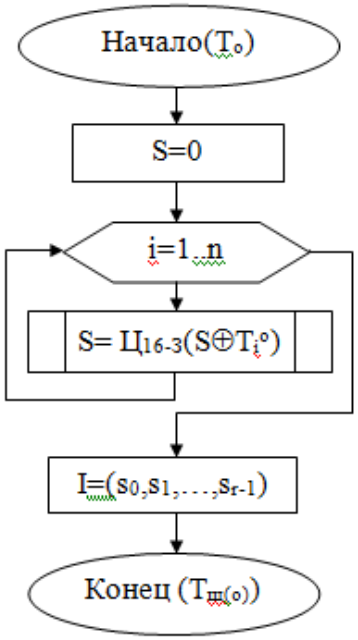

Рис. 2.5 Режим выработки имитовставки

2.3 Программная реализация

В ходе выпускной квалификационной работы, нами была разработана обучающая программа по режимам шифрования ГОСТ 28147-89. В программе реализованы следующие режимы шифрования: режим простой замены, гаммирования, гаммирования с обратной связью, также есть возможность вычисления имитовставки и синхропосылки. Программа считывает шифруемый (исходный) файл, считывает ключ (файл ключа), заносит результат в зашифрованный файл.

Шифрование содержания исходного текстового файла с расширением txt с получением текстового файла, символы которого являются символами исходного файла, преобразованными по алгоритму шифрования методом открытого ключа.

Дешифровка содержания текстового файла по алгоритму шифрования методом открытого ключа с получением текстового файла, символы которого являются символами исходного текстового файла.

Программа предназначена для студентов ВУЗов, изучающих информационные компьютерные технологии, и реализует возможность изучения криптографических преобразований в различных режимах.

Переход к режимам шифрования осуществвляется с главного окна (Рис.3.1).

Режим шифрования простой заменой - это шифрование при помощи одного цикла зашифровки. Фактически, цикл зашифровки и простая замена - одно и то же понятие. Этот режим шифрования является базовым. Его необходимо освоить в первую очередь, поскольку в остальных режимах (гаммирование) он будет использоваться. Для режима простой подстановки должен быть задан ключ и таблица замен. По умолчанию, они заданы в файлах key.txt и tz.txt соответственно. Разумеется, вы можете их изменить или загрузить свои файлы (Рис.3.2).

Шифрование при помощи наложения на данные криптографической гаммы, т.е. последовательности элементов данных, вырабатываемых с помощью криптографического алгоритма. Это позволяет избавится от недостатка простой замены(одни и те же 64 битовые блоки при простой замене будут зашифровываться одинаково).

Отличается от гаммирования тем, что очередной элемент гаммы вырабатывается как результат преобразования по циклу зашифровки предыдущего блока зашифрованных данных, а для зашифрования первого блока массива данных элемент гаммы вырабатывается как результат преобразования по тому же циклу синхропосылки (Рис.3.3).

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

Программа имеет удобный интерфейс, позволяющий работать с ней пользователям любого уровня.

Описание пунктов меню:

Кнопки режимы шифрования: режим простой замены, гаммирования, гаммирования с обратной связью, также есть возможность вычисления имитовставки и синхропосылки.

Ввод ключа.

Ввод исходного текста.

Выход - для выхода из программы.

В программе требуется ввести ключ и нажать на одну из кнопок режима , после нажатия на которую пользователь перейдёт на другую форму с конкретным режимом. Из новой формы есть возможность вернуться назад к меню. В каждом из меню есть текстовое поле с теоретическим материалом, для более подробного изучения и описания режима, и текстовое поле в которое выводится результат.

Листинг 2.1. Фрагмент листинга реализации метода шифрования простой замены

```
char s[50]; char massiv[100][100]; int b,c; int count = 0;
   setlocale(LC_ALL, "rus");
   for(int i=0; i<b; i++)
     for(int j=0; j<c; j++)
       { massiv[i][j]=s[count]; count++;       } }
 for(int j=0; j<c; j++) {
       for(int i=0; i<b; i++) {
          if(massiv[i][j] >= 'a' && massiv[i][j] <= 'z')    }}
```

Зашифрование в режиме простой замены заключается в применении цикла 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 3а и б соответственно, они тривиальны и не нуждаются в комментариях.

Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам: |То|=|Тш|=64·n, после выполнения операции размер полученного массива данных не изменяется.

Алгоритм программы метода шифрования Гаммирования заключается в следующем:

При открытии файла код каждого символа запоминается в целочисленном массиве;

в соответствии с введёнными пользователем значениями p и q (или значениями, используемыми по умолчанию) вычисляются и/или подбираются значения всех остальных чисел, участвующих в формировании ключа и формулах за- и расшифровывания;

Тогда (ar, mod n) может быть вычислено помощью не более k-1 произведений и сведением каждого произведения по модулю n. Таким образом, достаточно вычислить числа, которые требуют k модульных возведения в квадрат и дополнительно не более k –1 модульных произведений. Это означает, что вычисляется не более 2k – 1 произведений с обоими множителями, меньшими, чем n, и сведением произведений по модулю n. Если r является большим и известно m, то r может быть вначале взято по модулю

Этот режим очень похож на режим гаммирования и отличается от него только путем создания элементов шкалы - это еще один элемент диапазона, созданного в результате цикла преобразования 32-S предыдущего блока зашифрованных данных, и для кодирования элемента массива данных первого блока в диапазоне генерируется как результат преобразования в том же синхротронном цикле. Это достигается за счет включения блоков -

каждый блок зашифрованного текста в этом режиме зависит от соответствующего и всех предыдущих блоков открытого текста. Таким образом, этот режим иногда называют блоками крюка для молотка. Сила шифрования заключается в том, что блоки захвата не влияют.

Для решения проблемы обнаружения искажений в зашифрованном массиве данных с заданной вероятностью в Госте, криптографическая трансформация дополнительного режима - разработка. Иммитовставка - это комбинация управления, которая зависит от открытых данных и информации секретного ключа. Цель использования метода - обнаружить любые случайные или преднамеренные изменения данных. Проблема, изложенная в предыдущем абзаце, может быть решена путем добавления к зашифрованным данным. Для потенциального злоумышленника следующие две проблемы практически неразрешимы, если у него нет ключевой информации, такой как расчет иммитовставки для указанной информации открытого массива и подбор открытых данных под заданную имитовставку.

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–|И| на одну попытку подбора. При использовании имитовставки размером 32 бита эта вероятность равна 2–32»0.23·10–9.

Данная программа написана на языке C++ в соответствии с принципами структурного программирования и представляет собой набор процедур и функций, выполняющих определенные действия основной программы, осуществляющей синхронизацию процесса обработки данных и вызов соответствующих функций. Такой подход к написанию программ позволяет с высокой степенью эффективности осуществлять их отладку, а также, изменять их функциональные возможности, так как в данном случае за каждую операцию, выполняемую программой, отвечает относительно автономная подпрограмма (процедура или функция).

Рассматриваемая программа предусматривает работу в диалоге с пользователем.

Основная программа представляет собой три этапа: подготовка к работе, сама работа и её завершение.

Алгоритм основной программы заключается в инициализации модулей программы и, в зависимости от выбора пользователя, управление передаётся одной из процедур или функций, находящейся в определённом модуле, либо осуществляется выход из программы. Все функции работают автономно и требуют для работы вызова основных модулей библиотеки.

Процедуры по обработке данных требуют предварительного введения исходных данных из файла.

Таким образом, с точки зрения пользователя программа представляет собой исполняемый EXE файл и два модуля небольшого размера. Для запуска программы используется EXE файл, действующий под управлением операционной системы Windows. Программный интерфейс программы представляет собой меню, из которого происходит доступ ко всем процедурам и функциям программы.

С точки зрения программиста программный продукт представляет собой структурированную программу на языке C++, хранящуюся в пяти исходных файлах, один из которых является главным и содержит код инициализации модулей, а четыре других являются модулями, содержащими основные процедуры и функции, выполняющие вычисления для преобразования символов. Таким образом, в текст программы, кроме основных подпрограмм управления процессом обработки информации и создающих интерфейс, входят следующие процедуры и функции:

Процедура вычисления n и m; в этой же процедуре находится d – взаимно обратное число с m и вычисляется e;

Функция для работы с большими числами, т.е. функция, осуществляющая алгоритм последовательного возведения в квадрат, описанный выше;

функция возведения в степень (результат не должен превышать значение $9*10^{18}$);

Находясь в программе, пользователь может узнать об объекте, расположенном на форме, направив курсор мыши на данный объект. При этом около курсора мыши появится всплывающее меню с краткой информацией о данном объекте, а более полная информация буде отображаться на информационной панели, находящейся в нижней части формы. В правой части формы находится комплекс для выбора и просмотра того или файла. Просмотр файла осуществляется путём нажатия на клавишу View; при этом данные будут отображаться в окне просмотра информации.

3. АПРОБАЦИЯ ОБУЧАЮЩЕЙ ПРОГРАММЫ ПО РЕЖИМАМ ШИФРОВАНИЯ ГОСТ 28147

3.1 Программа и методика испытаний

Чтобы выявить несоответствия между требованиями и разработанной программой, проведём испытание программного обеспечения.

После того как программное обеспечение было спроектировано и разработано, необходимо его протестировать, чтобы убедиться в отсутствии ошибок, а также проверить функциональность, требуемой постановкой задачи.

Цель проведения испытаний - проверка соответствия характеристик разработанной программного обеспечения функциональным и отдельным иным видам требований, изложенным в требованиях к разрабатываемой программе.

Необходимо проверить:

Соответствует ли программа заданию и требованиям. При проведении испытаний функциональные характеристики или возможности программы подлежат проверке на соответствие требованиям, изложенным в п. «Требования к составу выполняемых функций» технического задания

Проверить комплектность и качества материала программы;

Установить, кто будет испытывать программу;

Проверка соответствия программы требованиям функционального назначения.

Необходимо установить, соответствует ли программа поставленным целям для кого будет использоваться и кем испытываться.

Необходимо проверить требования к функциональным характеристикам. Обеспечивает ли программа возможность выполнения функций инициализации системы (материал программы должен

соответствовать материалу изучаемых дисциплин в соответствии с учебными планами и т. п.), ввод и коррекцию текущей информации содержащейся в программе, возможность дополнения программы, хранение информации ключей исходных данных и зашифрованного файла, получение сведений об изучаемой теме, помощь работы в программе.

Необходимо произвести проверку исходных данных. Файл ключа, исходный файл для шифрования сообщения, содержащий массив данных, режимы шифрования.

Проверить выполнены ли требования к результатам программы, требования к надежности, требования к обеспечению надежного функционирования программы, предусмотреть контроль вводимой информации, предусмотреть блокировку некорректных действий пользователя при работе с системой, обеспечить целостность хранимой информации.

3.2 Результаты испытаний

Проведем последовательно тестирование всех заявленных функций.

Для этого предлагается следующий план действий:

- Открыть программу;
- Отрыть вкладки помощь для изучения работы программы;
- Выбрать режим шифрования;
- Запуск процесса шифрования;
- Сохранение зашифрованного текста в файл;
- Сравнение результатов кодирования и исходной информации;
- Открытие зашифрованного файла;
- Сверить зашифрованный файл исходной;
- Выход в меню;
- Выход из программы

- Повторный запуск программы;
- Выполнить все вышеперечисленные действия с разными режимами шифрования.

Были проверены отдельные модули программы. После запуска программы открывается главное окно программы, в данном окне представлено меню для выбора модулей программы, при нажатии на кнопку режима, пользователь может перейти к нужному режиму шифрования, простой замены, гаммирования, гаммирования с обратной связью, выработки имитовставки, помощь. Кнопка Выход, завершает работу программы и закрывает главное окно.

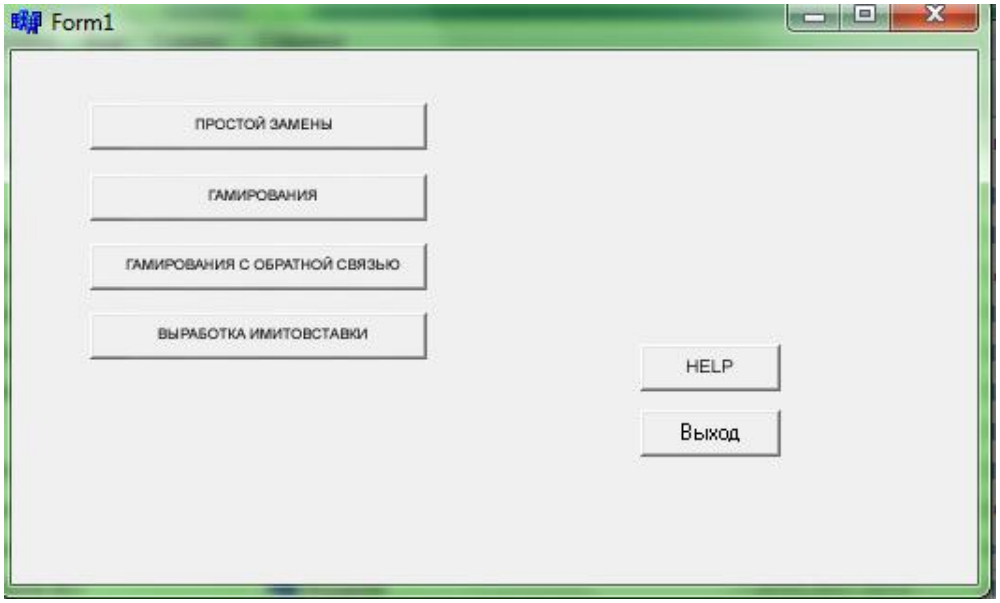

Рис.3.1 основное окно программы

При открытии окна одного из режимов шифрования, например метода шифрования простой замены (рис. 3.2), выводится блок-схема описывающая работу режима, текстовое описание метода, в текстовое поле вводится ключ и происходит зашифрование. Результат выводится в отдельный файл.

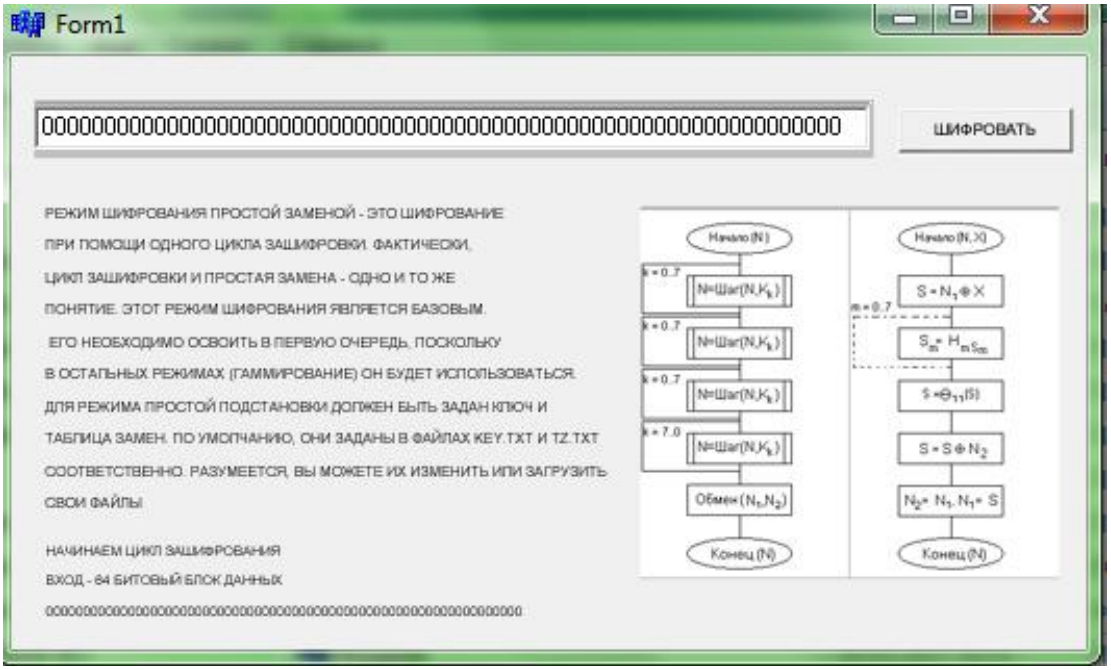

Рис.3.2. Метод простой замены

Все отдельные модули были проверены, выполнения программы прошло в соответствии с поставленными ранее требованиями к обучающей программе. Проверены ключи программы, входные и выходные данные. На рисунках представлены результаты компиляции программы и отдельных модулей.

В методе гаммирования и гаммирования с обратной связью исходными данными является данные для шифровки и синхропосылка. На форме также отображается краткое описание данных методов шифрования и блок-схема, данные выводятся в отдельный файл и на форме в компонент «label».

Рис.3.3. Метод гаммирования с обратной связью.

Форма реализующая режим выработки иммитовставки также была проверена в ходе испытнаий программного обеспечения, данные вводнятся в текстовое поле компонент «Edit», содеожится краткое теоретическое описание метода вывод происходит в файл. Данная форма соответвтует заданным требоаниям к программному обеспечению.

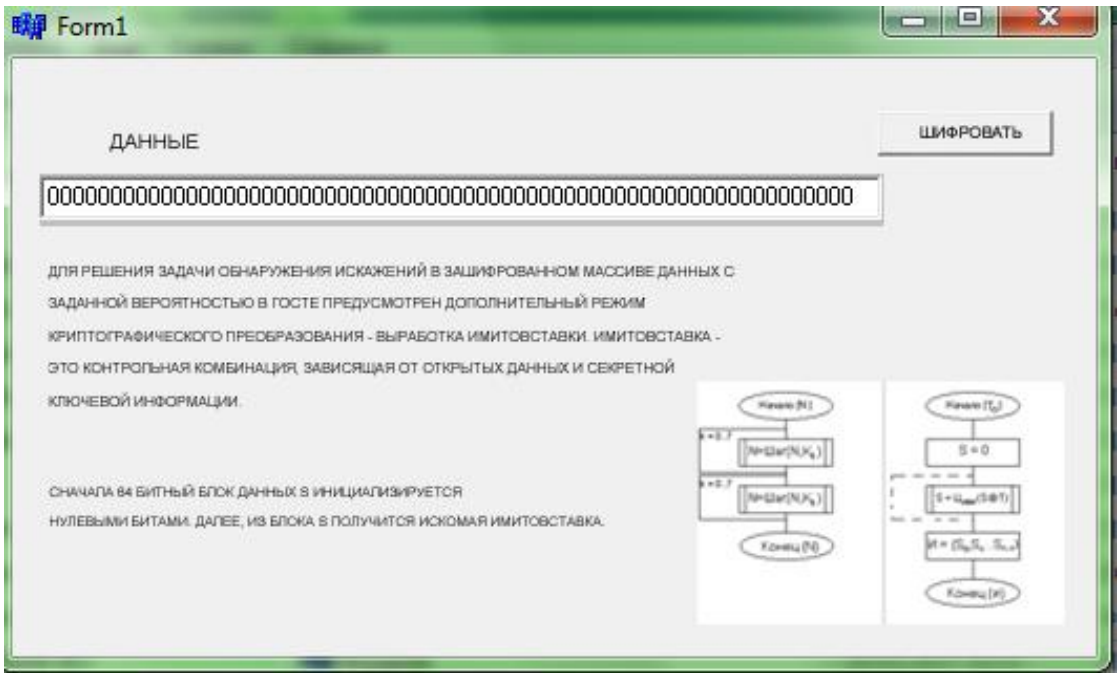

Рис.3.4. Метод выработки имитовставки.

Таким образом, можно сделать вывод, что программа работает корректно, как и все заявленные функции данного программного обеспечения.

Были соблюдены требования к обеспечению надежного функционирования программы, предусмотрен контроль работы в программе, блокировка некорректных действий пользователя при работе с системой, была обеспечена целостность хранимой информации.

Материал программы соответствует материалу изучаемых дисциплин, дополняя традиционное обучение;

Программа имеет удобный дружественный интерфейс, легка в использовании, имеется помощь при работе с программой и содержит небольшую часть теоретического материала.

Хранение информации ключей исходных данных и зашифрованного файла, файл с зашифрованным текстом, хранятся в отдельных файлах, пользователи программы имеют возможность получения сведений об изучаемой теме, и помощи работы с программным обеспечением.

ЗАКЛЮЧЕНИЕ

Данная программа разработана в соответствии с постановкой задачи разработка обучающей программы по режимам шифрования ГОСТ 28147-89. При написании программы использовалась информация ГОСТа 28147-89, был сделан его обзор, изучены методические указания. В ходе работы были раскрыты возможности режимов ГОСТ в качестве средств обучения, изучены средства информационных технологий для проектирования обучающей программы ГОСТ 28147-89, были рассмотрены средства для разработки компьютерной обучающей программы, выявлены и реализованы требования для реализации обучающей программы, обоснован выбор средств, для собственно программы. Было произведено сравнение с другими программами, например программа DES или обучающая программа ГОСТ, находящаяся в свободных источниках.

Разработана и экспериментально протестирована обучающая программа по режимам шифрования ГОСТ 28147-89. Интерфейс программы удобен для использования. Выходные данные представлены в виде текстового файла. По своей структуре программа хорошо организована, что позволяет в случае необходимости легко ее модифицировать. Для проверки работоспособности программы и правильности обработки входных данных разработан тестовый пример. Тестирование программы подтвердило, что программа правильно выполняет обработку данных и выдаёт верные результаты.

Всё это свидетельствует о работоспособности программы и позволяет сделать вывод о пригодности программы к использованию её в целях обучения шифрованию данных.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Винокуров А. Алгоритм шифрования ГОСТ 28147-89, его использование и реализация для компьютеров платформы Intel x86.

2. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.

3. Ross Anderson, Markus Kuhn, *Tamper Resistance - a Cautionary Note,* proceedings of the Second Usenix Workshop on Electronic Commerce, pp. 1-11, November 1996.

4. Ross Anderson, Markus Kuhn, *Low Cost Attacks on Tamper Resistant Devices*, proceedings of the 1997 Security Protocols Workshop, Paris, April 7-9, *1997.*

5. Eli Biham, *New Types of Cryptanalytic Attacks Using Related Keys*, Journal of Cryptology, Vol. 7, No. 4, pp. 229-246, 1994.

6. Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.

7. John Kelsey, Bruce Schneier, David Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES,* Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'96, pp. 237-251,1996. Paul C. Kocher, *Timing Attack's on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems,* Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'96, pp. 104-113, 1996.

8. Dan Boneh, Richard A. Demillo, Richard J. Lipton, *On the Importance of Checking Cryptographic Protocols for Faults,* Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'97, pp. 37-51, 1997.

9. Brown, Josef Pieprzyk, Jennifer Seberry, *LOKI - A Cryptographic*

Primitive for Authentication and Secrecy Applications, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of AUSCRYPT'90, pp. 229-236,1990.

10. Xuejia Lai, James L. Massey, Scan Murphy, *Markov Ciphers and Differential Cryptanalysis,* Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'91. pp. 17-38, 1991.

11. Susan K. Langford, Martin E. Hellman, *Differential-linear cryptanalysis*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'94, pp. 17-25, 1994.

12. Mitsuru Matsui, *Linear Cryptanalysis Method for DES Cipher,* Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'93, pp. 386-397, 1993.

13. Ralph C. Merkle, *Fast Software Encryption Functions*, Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'90, pp. 476-501, 1990.

14. Shoji Miyaguchi, *FEAL-N specifications,* technical note, NTT, 1989.

15. Advances in Cryptology, proceedings of CRYPTO'90, pp. 627-638, 1990.

16. National Bureau of Standards, *Data Encryption Standard,* U.S. Department of

Commerce, FIPS pub. 46, January *1977.*

17. Bart Preneel, Marnix Nuttin, Vincent Rijmen, Johan Buelens, *Cryptanalysis of the CFB Mode of the DES with a Reduced Number of Rounds,* Lecture Notes in Computer Science, Advances in Cryptology, proceedings of CRYPTO'93, pp. 212-223, 1993.

18. Ronald L. Rivest, *The RC5 Encryption Algorithm,* proceedings of Fast Software Encryption, Leuven, Lecture Notes in Computer Science, pp. 86-96, 1994.

19. Bruce Schneier, *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*, proceedings of Fast Software Encryption, Cambridge, Lecture Notes in Computer Science, pp. 191-204, 1993.

20. Akihiro Shimizu, Shoji Miyaguchi, *Fast Data Encryption Algorithm FEAL*,

Lecture Notes in Computer Science, Advances in Cryptology, proceedings of EUROCRYPT'87, pp. 267-278. 1987.

ПРИЛОЖЕНИЕ

Листинг 1. Главное меню программы.

```
#include <vcl.h>
#pragma hdrstop

#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
#include "Unit5.h"
#include "Unit6.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "sBitBtn"
#pragma link "sButton"
#pragma link "sLabel"
#pragma resource "*.dfm"
TForm1 *Form1;

//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm1::sButton1Click(TObject *Sender)
```

```
{
Form2->Show();
Form1->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::sButton2Click(TObject *Sender)
{
  Form3->Show();
Form1->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::sButton3Click(TObject *Sender)
{
Form4->Show();
Form1->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::sButton4Click(TObject *Sender)
{
Form5->Show();
Form1->Hide();
}
//---------------------------------------------------------------------------
void __fastcall TForm1::sButton5Click(TObject *Sender)
{
 Form6->Show();
Form1->Hide();
}
//---------------------------------------------------------------------------
```

Листинг 2. Метод простой замены

```
{
    char s[50];
    char massiv[100][100];
    int b,c;
  StrToInt(Edit1->Text);
    int count = 0;
    setlocale(LC_ALL, "rus");
    for(int i=0; i<b; i++)
    {
       for(int j=0; j<c; j++)
       {
          massiv[i][j]=s[count];
          count++;
       }
    }
     for(int j=0; j<c; j++)
    {
       for(int i=0; i<b; i++)
       {
          if(massiv[i][j] >= 'a' && massiv[i][j] <= 'z')
                }
Edit1->Text =  IntToStr(a)

    }
```

Листинг 3. Метод гамирования

```
oid GammaCoding( string &input, string &gamma, string &result )
{
```

```
  result.clear();
  for( string::iterator i=input.begin(), j=gamma.begin(); i<input.end(); i++, j++ )
  {
   if(j==gamma.end()) j=gamma.begin();
   int Ti = *i - FIRST_SYMBOL;
   int Gi = *j - FIRST_SYMBOL;
   result.push_back(FIRST_SYMBOL+(Ti+Gi)%SYMBOL_NUMBER);
  }
 }

 void GammaDecoding( string &input, string &gamma, string &result )
 {
  result.clear();
  for( string::iterator i=input.begin(), j=gamma.begin(); i<input.end(); i++, j++ )
  {
   if(j==gamma.end()) j=gamma.begin();
   int Ci = *i - FIRST_SYMBOL;
   int Gi = *j - FIRST_SYMBOL;
   result.push_back(FIRST_SYMBOL+(Ci-
Gi+SYMBOL_NUMBER)%SYMBOL_NUMBER);
  }}
 int main()
 {
    setlocale(LC_ALL, "rus");
    string str1, gamma, rez;
    string str1_v, gamma_v, rez_v;
    GammaCoding( str1, gamma, rez );
    GammaDecoding( rez, gamma, str1 );}
```

Листинг 4. Метод гаммирования с обратной связью

```
SetConsoleCP(1251);
SetConsoleOutputCP(1251);

setlocale( LC_CTYPE,"Russian" ); //включаем поддержку русского языка
```

```
FILE*f,*f1; //описываем переменные f1, типа file в виде указателей, через них мы обратимся к файлам типа TXT
f=fopen("in.txt","r");  //открываем файл, с которого будем считывать текст
f1=fopen("out.txt","w");  //открываем файл в который будем записывать текст
char d;
 while(d=fgetc(f)!=EOF)
{//поместить сюда цикл гаммирования
    fputc(d,f1);
}
fclose(f);
fclose(f1);
```

Листинг5. Режим выработки имитовставки

```
//---------------------------------------------------------------------------

#include <vcl.h>
#pragma hdrstop

#include "Unit5.h"
#include "Unit1.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm5 *Form5;
//---------------------------------------------------------------------------
__fastcall TForm5::TForm5(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------

void __fastcall TForm5::Button1Click(TObject *Sender)
{
 Edit1->Text =  IntToStr(e);

v=v<<<1

e=v xor w
```

```
x=((((e+y) %pow(2,32))ˇA^C)*(x xor Mi))% 2^32-1;

y=((((e+x) % pow(2,32))ˇB^D)*(y xor Mi))% 2^32-1;

StrToInt(Edit2->Text);
Memo1->Lines->Add(IntToStr(e);
}
//---------------------------------------------------------------------------
void __fastcall TForm5::Button2Click(TObject *Sender)
{
Form5->Hide();
Form1->Show();
}
//---------------------------------------------------------------------------
```