**2.3 Перечень тем лабораторных занятий по дисциплине**

**Методы защиты информации**

Основная цель проведения лабораторных занятий состоит в закреплении теоретического материала курса, приобретении навыков выполнения эксперимента, обработки экспериментальных данных, анализа результатов, грамотного оформления отчетов.

|  |  |  |
| --- | --- | --- |
| №  темы  по п.1 | Наименование  лабораторной работы | Содержание |
| 3 | Лабораторная работа № 1  Симметричная криптография. Стандарт шифрования ГОСТ 28147-89 | Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи стандарта шифрования ГОСТ 28147-89 в следующих режимах:   1. Простой замены 2. Гаммирования 3. Гаммирования с обратной связью 4. Генерации имитоприставок |
| 3 | Лабораторная работа № 2  Симметричная криптография. СТБ 34.101.31-2011 | Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи алгоритма СТБ 34.101.31-2011 в различных режимах |
| 4 | Лабораторная работа № 3  Асимметричная криптография. Криптосистема Рабина | Реализовать криптостойкое программное средство шифрования и дешифрования текстовых файлов при помощи Криптосистемы Рабина |
| 4 | Лабораторная работа № 4  Асимметричная криптография. Алгоритм Мак-Элиса | Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи алгоритма Мак-Элиса для криптостойких размеров порождающей матрицы |
| 5 | Лабораторная работа № 5  Хеш-функции | Реализовать программное средство контроля целостности сообщений с помощью вычисления хеш-функции и алгоритма ГОСТ 34.11 и SHA 1 |
| 6 | Лабораторная работа № 6  Цифровая подпись | Реализовать программное средство формирования и проверки ЭЦП на базе алгоритма ГОСТ 34.10 |
| 7 | Лабораторная работа № 7  Криптография с ис­пользованием эллипти­ческих кривых | Реализовать схему шифрования (дешифрования) для аналога алгоритма Эль-Гамаля на основе эллиптических кривых |
| 8 | Лабораторная работа № 8  Стеганографические методы | Реализовать программное средство, сокрытия (извлечения) текстового сообщения в (из) JPEG изображение(я) на основе метода сокрытия в частотной области изображения |

По результатам работы студентом должен быть представлен и защищен отчет. Содержание отчета включает:

1. Титульный лист, подписанный студентом и преподавателем с оценкой после защиты работы
2. Введение, содержащее общие теоретические сведения и индивидуальную постановку задачи.
3. Полную блок – схему разработанного алгоритма.
4. Распечатку скриншотов результатов ввода данных и исполнения программы.
5. Распечатку программного кода.
6. Вывод о решении поставленной задачи и иные сведения, относящиеся к процессу ее решения.

**Лабораторная работа № 1. Симметричная криптография. Стандарт шифрования ГОСТ 28147-89**

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

# Общие сведения

**ГОСТ 28147-89** «*Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования*» — устаревший государственный стандарт СССР (позже межгосударственный стандарт СНГ), описывающий алгоритм симметричного блочного шифрования и режимы его работы.

Является примером DES-подобных криптосистем, созданных по классической итерационной схеме Фейстеля.

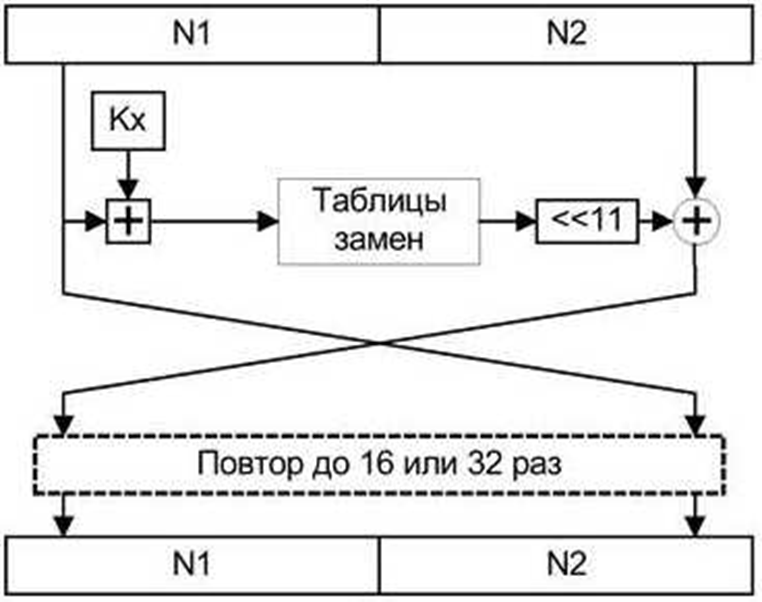
ГОСТ 28147-89 представляет собой симметричный 64-битовый блочный алгоритм с 256-битовым ключом.

Этот алгоритм криптографического преобразования данных предназначен для аппаратной и программной реализации, удовлетворяет криптографическим требованиям и до 1983 года не накладывал ограничений на степень секретности защищаемой информации.

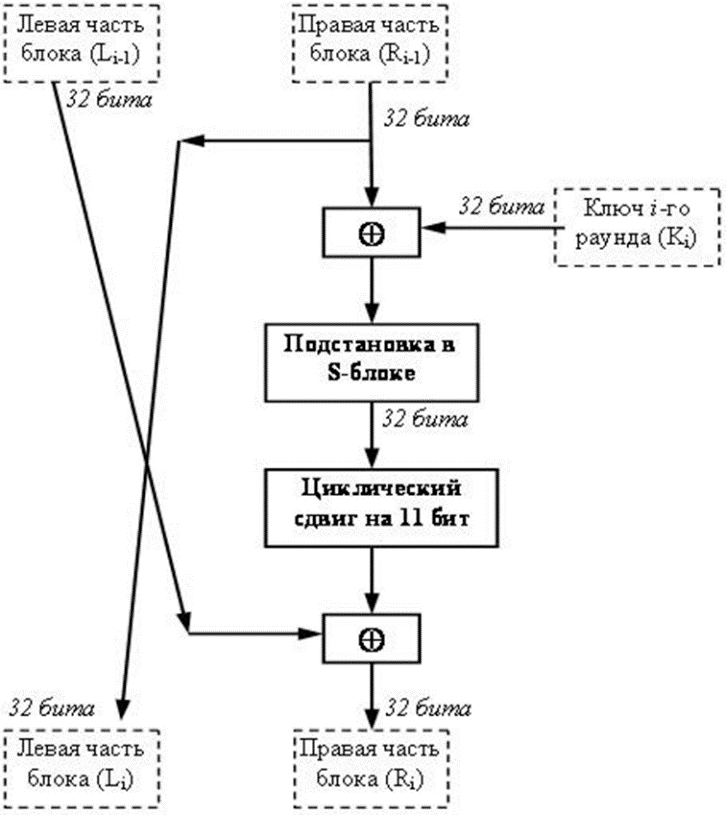
# Описание работы ГОСТ 28147-89

Схема работы алгоритма ГОСТ 28147-89 следующая. Данные, подлежащие зашифровке, разбивают на 64-разрядные блоки.

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



* + Структура одного раунда ГОСТ 28147-89\



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

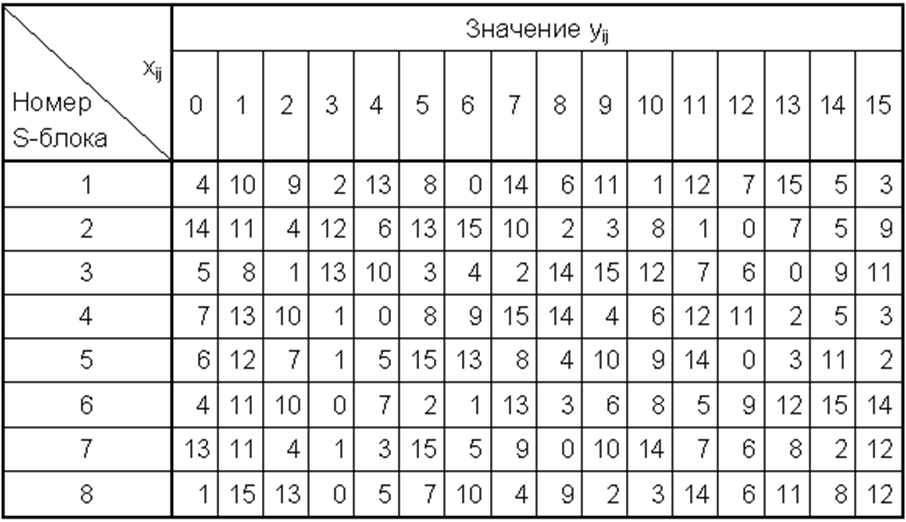
Сначала правая половина блока и подключ раунда складываются по модулю 232.

Затем производится поблочная подстановка.

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

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

Таблица замен ГОСТ 28147-89



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

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

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

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

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

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

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

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

# Процедуры шифрования и расшифрования

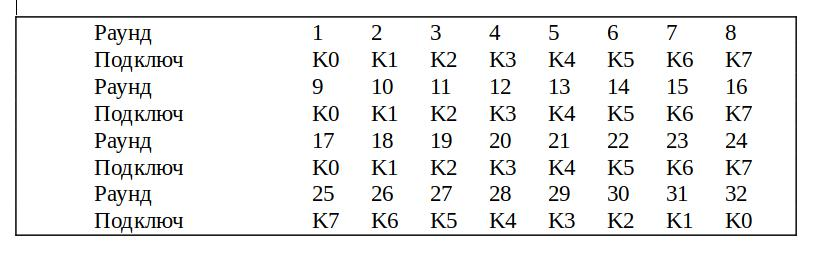
ГОСТ 28147-89 является блочным шифром, поэтому преобразование данных осуществляется блоками в так называемых базовых циклах.

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

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

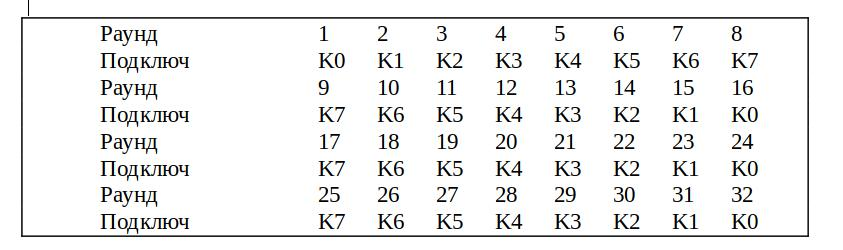
Рассмотрим процесс создания подключей раундов. В ГОСТ эта процедура очень проста, особенно по сравнению с DES. 256-битный ключ K разбивается на восемь 32-битных подключей, обозначаемых K0, K1, K2, K3, K4, K5, K6, K7. Алгоритм включает 32 раунда, поэтому каждый подключ при шифровании используется в четырех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

Таблица 1. Последовательность использования подключей при шифровании



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

Таблица 2. Последовательность использования подключей при расшифровании



# ГОСТ 28147-89 Режимы работы

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

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

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



Рисунок 4. Работа криптосистемы в режиме гаммирования

**ЗАДАНИЕ:**

1. Изучить теоретические сведения.
2. Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи стандарта шифрования ГОСТ 28147-89 в следующих режимах:

* Простой замены
* Гаммирования
* Гаммирования с обратной связью
* Генерации имитоприставок

**Лабораторная работа № 2. Симметричная криптография. СТБ 34.101.31-2011**

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

# 2.1. Общие сведения

Настоящий стандарт определяет семейство криптографических алгоритмов, предназначенных для обеспечения конфиденциальности и контроля целостности данных. Обрабатываемыми данными являются двоичные слова (сообщения).

Криптографические алгоритмы стандарта построены на основе базовых алгоритмов шифрования блока данных.

Криптографические алгоритмы шифрования и контроля целостности делятся на восемь групп:

1) алгоритмы шифрования в режиме простой замены;

2) алгоритмы шифрования в режиме сцепления блоков;

3) алгоритмы шифрования в режиме гаммирования с обратной связью;

4) алгоритмы шифрования в режиме счетчика;

5) алгоритм выработки имитовставки;

6) алгоритмы одновременного шифрования и имитозащиты данных;

7) алгоритмы одновременного шифрования и имитозащиты ключа;

8) алгоритм хеширования.

Первые четыре группы предназначены для обеспечения конфиденциальности сообщений. Каждая группа включает алгоритм зашифрования и алгоритм расшифрования.

Стороны, располагающие общим ключом, могут организовать конфиденциальный обмен сообщениями путем их зашифрования перед отправкой и расшифрования после получения. В режимах простой замены и сцепления блоков шифруются сообщения, которые содержат хотя бы один блок, а в режимах гаммирования с обратной связью и счетчика – сообщения произвольной длины.

Пятый алгоритм предназначен для контроля целостности сообщений с помощью имитовставок – контрольных слов, которые определяются с использованием ключа. Стороны, располагающие общим ключом, могут организовать контроль целостности при обмене сообщениями путем добавления к ним имитовставок при отправке и проверки имитовставок при получении. Проверка имитовставок дополнительно позволяет стороне-получателю убедиться в том, что сторона-отправитель знает ключ, т. е. позволяет проверить подлинность сообщений.

# 2.2. Блок-схема алгоритма шифрования в режиме простой замены

Блок-схема алгоритма на i-ом такте шифрования представлена на рисунке 1.

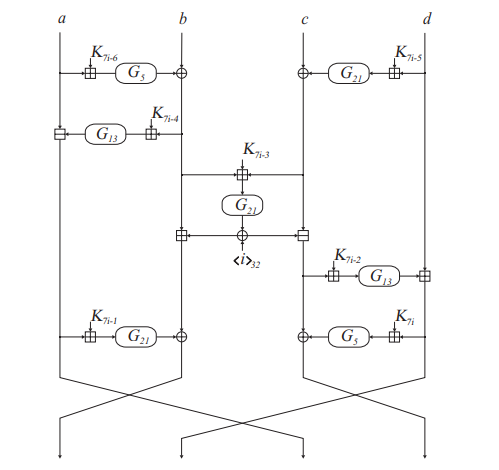


Рисунок 2.1 – Вычисления на i-м такте шифрования

Входными данными алгоритмов зашифрования и расшифрования являются блок  {\displaystyle X\in \{0,1\}^{128}}и ключ {\displaystyle \theta \in \{0,1\}^{256}.}

Выходными данными является блок {\displaystyle Y\in \{0,1\}^{128}}— результат зашифрования либо расшифрования слова на ключе  либо 

Входные данные для шифрования подготавливаются следующим образом:

* Слово {\displaystyle X} записывается в виде {\displaystyle X=X\_{1}\|X\_{2}\|X\_{3}\|X\_{4},X\_{i}\in \{0,1\}^{32}.}
* Ключ записывается в виде  и определяются тактовые ключи

Обозначения и вспомогательные преобразования

Преобразование  ставит в соответствие слову , слово



* циклический сдвиг влево на r бит.

 операция замены 8-битной входной строки подстановкой с рисунка 4.

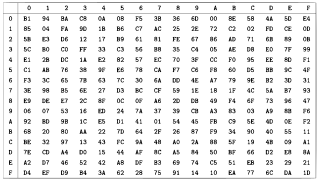


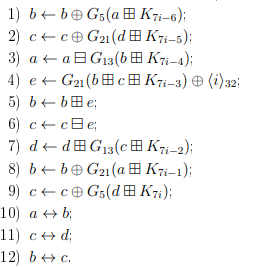
Рисунок 2.2 – Преобразование Н

Подстановка {\displaystyle H:\{0,1\}^{8}\rightarrow \{0,1\}^{8}}задается фиксированной таблицей. В таблице используется шестнадцатеричное представление слов {\displaystyle u\in \{0,1\}^{8}.}

{\displaystyle \boxplus } и  {\displaystyle \boxminus -}операции сложения и вычитания по модулю 232{\displaystyle 2^{32}.}

Для зашифрования блока {\displaystyle X} на ключе {\displaystyle \theta } выполняются следующие шаги:

1. Установить {\displaystyle a\leftarrow X\_{1},b\leftarrow X\_{2},c\leftarrow X\_{3},d\leftarrow X\_{4}.}
2. Для {\displaystyle i}i = 1,2,… ,8 выполнить:

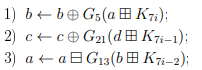


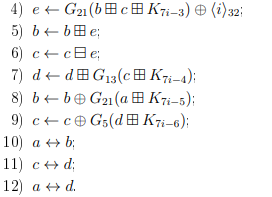
3. Установить {\displaystyle Y\leftarrow b\|d\|a\|c.}

4. Возвратить {\displaystyle Y.}

Для расшифрования блока {\displaystyle X} на ключе {\displaystyle \theta } выполняются следующие шаги:

1. Установить {\displaystyle a\leftarrow X\_{1},b\leftarrow X\_{2},c\leftarrow X\_{3},d\leftarrow X\_{4}.}
2. Для {\displaystyle i}i = 8,7,… ,1 выполнить:





3. Установить {\displaystyle Y\leftarrow c\|a\|d\|b.} 

4. Возвратить {\displaystyle Y.} 

# 2.3 Шифрование в режиме гаммирования с обратной связью

Входными данными алгоритмов зашифрования и расшифрования являются сообщение 𝑋 ∈ {0, 1}\* , ключ O∈{0, 1}256 и синхропосылка 𝑆 ∈ {0, 1}128 .

Выходными данными является слово 𝑌 ∈ {0, 1}|𝑋| — результат зашифрования либо расшифрования 𝑋 на ключе 𝜃 при использовании синхропосылки 𝑆.

Входное сообщение 𝑋 записывается в виде 𝑋 = 𝑋1 ‖ 𝑋2 ‖ . . . ‖ 𝑋𝑛, |𝑋1| = |𝑋2| = . . . = |𝑋𝑛−1| = 128, |𝑋𝑛| <= 128.

При шифровании словам 𝑋𝑖 ставятся в соответствие слова 𝑌𝑖 ∈ {0, 1} |𝑋𝑖| , из которых затем составляется 𝑌 .

При зашифровании используется вспомогательный блок 𝑌0 ∈ {0, 1}128, а при расшифровании — вспомогательный блок 𝑋0 ∈ {0, 1}128

Зашифрование сообщения 𝑋 на ключе 𝜃 при использовании синхропосылки 𝑆 состоит в выполнении следующих шагов:

1. Установить 𝑌0 ← 𝑆.
2. Для 𝑖 = 1, 2, . . . , 𝑛 выполнить: 𝑌𝑖 ← 𝑋𝑖 ⊕ 𝐿|𝑋𝑖|(𝐹(𝑌𝑖−1)).
3. Установить 𝑌 ← 𝑌1 ‖ 𝑌2 ‖ . . . ‖ 𝑌𝑛.
4. Возвратить 𝑌.

Расшифрование сообщения 𝑋 на ключе 𝜃 при использовании синхропосылки 𝑆 состоит в выполнении следующих шагов:

1. Установить 𝑋0 ← 𝑆.
2. Для 𝑖 = 1, 2, . . . , 𝑛 выполнить: 𝑌𝑖 ← 𝑋𝑖 ⊕ 𝐿|𝑋𝑖|(𝐹𝜃(𝑋𝑖−1)).
3. Установить 𝑌 ← 𝑌1 ‖ 𝑌2 ‖ . . . ‖ 𝑌𝑛.
4. Возвратить 𝑌.

**ЗАДАНИЕ:**

1. Изучить теоретические сведения.
2. Реализовать программные средства шифрования и дешифрования текстовых файлов при помощи стандарта шифрования СТБ 34.101.31-2011 в режимах:

* Простой замены
* Гаммирования с обратной связью

**Лабораторная работа № 3**

**Асимметричная криптография. Криптосистема Рабина**

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

# 3.1. Общие сведения

Криптосистема Рабина — криптографическая система с открытым ключом, безопасность которой обеспечивается сложностью поиска квадратных корней в кольце остатков по модулю составного числа. Безопасность системы, как и безопасность метода RSA, обусловлена сложностью разложения на множители больших чисел. Зашифрованное сообщение можно расшифровать 4 способами. Недостатком системы является необходимость выбора истинного сообщения из 4-х возможных.

Общий вид криптосистемы Рабина представлена на рисунке 1.

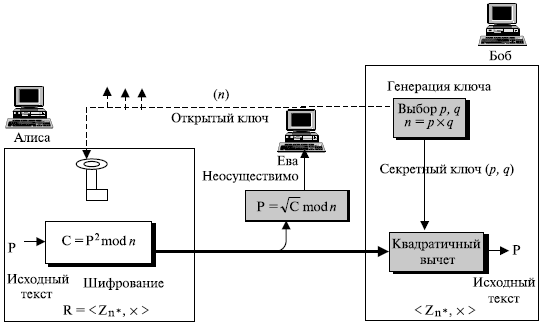
****

Рисунок 1 – Криптосистема Рабина общий вид

# 3.2 Генерация ключа

Система Рабина, как и любая асимметричная криптосистема, использует открытый и закрытый ключи. Открытый ключ используется для шифрования сообщений и может быть опубликован для всеобщего обозрения. Закрытый ключ необходим для расшифровки и должен быть известен только получателям зашифрованных сообщений.

Процесс генерации ключей следующий:

1 выбираются два случайных числа *p* и *q* с учётом требований:

1) числа должны быть большими;

2) числа должны быть простыми;

3) должны выполняться условия: p=4k+3; q=4k+3 и *p* ≠*q*, где kЄN

2 вычисляется число *n* = *p* · *q*;

3 число *n* — открытый ключ; числа *p* и *q* — закрытый.

Пример. Пусть *p* = 7 и *q* = 11. Тогда *n* = *p* · *q* = 7 · 11 = 77. Число *n* = 77 — открытый ключ, а числа *p* = 7 и *q* = 11 — закрытый. Получатель сообщает отправителям число 77. Отправители шифруют сообщение, используя число 77, и отправляют получателю. Получатель расшифровывает сообщение с помощью чисел 7 и 11. Приведённые ключи плохи для практического использования, так как число 77 легко раскладывается на простые множители (7 и 11).

# 3.3 Шифрование

Исходное сообщение *m* (текст) шифруется с помощью открытого ключа — числа *n* по следующей формуле *c* = *m*² mod *n*.

Благодаря использованию умножения по модулю скорость шифрования системы Рабина больше, чем скорость шифрования по методу RSA, даже если в последнем случае выбрать небольшое значение экспоненты.

Пример (продолжение). Пусть исходным текстом является *m* = 20. Тогда зашифрованным текстом будет: *c* = *m*² mod *n* = 20² mod 77 = 400 mod 77 = 15.

# 3.4 Расшифрование

Для расшифровки сообщения необходим закрытый ключ — числа *p* и *q*. Процесс расшифровки выглядит следующим образом:

1 используя алгоритм Евклида, из уравнения

находят числа и ;

2 далее, используя китайскую теорему об остатках, вычисляют четыре числа:

1)

2)

3)

4)

Одно из этих чисел является истинным открытым текстом *m*.

Пример (окончание). В результате расшифровки получаем: m ∈ { 64 , 20 , 13 , 57 }. Видим, что один из корней является исходным текстом *m*.

**ЗАДАНИЕ:**

1. Изучить теоретические сведения
2. Реализовать криптостойкое программное средство шифрования и дешифрования текстовых файлов при помощи Криптосистемы Рабина

**Лабораторная работа № 4**

**Асимметричная криптография. Алгоритм Мак-Элиса**

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

**4.1 Общие сведения**

**McEliece** — криптосистема с открытыми ключами на основе теории алгебраического кодирования, разработанная в 1978 году Робертом Мак-Элисом. Это была первая схема, использующая рандомизацию в процессе шифрования. Алгоритм основан на сложности декодирования полных линейных кодов.

Криптосистема имеет несколько преимуществ, например, над RSA. Шифрование и дешифрование проходит быстрее и с ростом длины ключа степень защиты данных растет гораздо быстрее. McEliece применим также в задачах аутентификации.

Идея, лежащая в основе данной системы, состоит в выборе корректирующего кода, исправляющего определенное число ошибок, для которого существует эффективный алгоритм декодирования. С помощью секретного ключа этот код «маскируется» под общий линейный код, для которого задача декодирования не имеет эффективного решения.

**4.2 Генерация ключа**

В системе Мак-Элиса параметрами системы, общими для всех абонентов, являются числа k, n, t. Для получения открытого и соответствующего секретного ключа каждому из абонентов системы следует осуществить следующие действия:

1 Выбрать порождающую матрицу G = Gkn двоичного (n,k)-линейного кода, исправляющего t ошибок, для которого известен эффективный алгоритм декодирования.

2 Случайно выбрать двоичную невырожденную матрицу S = Sk.

3 Случайно выбрать подстановочную матрицу P = Pn.

4 Вычислить произведение матриц G1 = S · G · P.

Открытым ключом является пара (G1, t), секретным – тройка (S, G, P).

**4.3 Шифрование**

Для того чтобы зашифровать сообщение M, предназначенное для абонента A, абоненту B следует выполнить следующие действия:

Представить M в виде двоичного вектора длины k.

Выбрать случайный бинарный вектор ошибок Z длиной n, содержащий не более t единиц.

Вычислить бинарный вектор C = M · GA + Z и направить его абоненту A.

# 4.4 Расшифрование

Получив сообщение C, абонент A вычисляет вектор C1 = C · P-1, с помощью которого, используя алгоритм декодирования кода с порождающей матрицей G, получает далее векторы M1 и M = M1 · S-1.

В качестве кода, исправляющего ошибки в системе Мак-Элиса, можно использовать код Гоппы. Известно, что для любого неприводимого полинома g(x) степени t над полем GF(2m) существует бинарный код Гоппы длины n = 2m и размерности k ≥ n – mt, исправляющий до t ошибок включительно, для которого имеется эффективный алгоритм декодирования.

Рекомендуемые параметры этой системы - n = 1024, t = 38, k > 644 – приводят к тому, что открытый ключ имеет размер около 219 бит, а длина сообщения увеличивается при шифровании примерно в 1,6 раза, в связи с чем данная система не получила широкого распространения.

**ЗАДАНИЕ:**

1. Изучить теоретические сведения
2. Реализовать криптостойкое программное средство шифрования и дешифрования текстовых файлов при помощи Криптосистемы Мак-Элиса.

**Лабораторная работа № 5. Хеш-функции**

**ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

# 5.1 Общие сведения

Хеш-функции получили широкое распространение в разнообразных алгоритмах быстрого поиска информации.

Однако с появлением криптографии у них появилась вторая, ничуть не меньшая, область применения.

Хеш-функцией (англ, hash — мелко измельчать и перемешивать) называется необратимое преобразование данных, обладающее следующими свойствами:

на вход алгоритма преобразования может поступать двоичный блок данных произвольной длины;

на выходе алгоритма получается двоичный блок данных фиксированной длины;

значения на выходе алгоритма распределяются по равномерному закону по всему диапазону возможных результатов;

при изменении хотя бы одного бита на входе алгоритма его выход значительно меняется: в идеальном случае инвертируется произвольная половина бит.

Основное, но не единственное, предназначение хеш-функций в криптографии — вычисление "неподделываемых" контрольных сумм документов.

Действительно, если для алгоритма хеширования выполняются требования невозможности подобрать иной документ с той же хеш-суммой и невозможности подобрать два документа с произвольной одинаковой хеш-суммой, то хеш-сумма становится уникальной характеристикой документа:

# 5.2 Алгоритм вычисления хеш-функции ГОСТ 34.11

В алгоритме ГОСТ 34.11 используются следующие преобразования:

*X-преобразование.* На вход функции X подаются две последовательности длиной 512 бит каждая, выходом функции является XOR этих последовательностей.

*S-преобразование*. Функция S является обычной функцией подстановки. Каждый байт из 512-битной входной последовательности заменяется соответствующим байтом из таблицы подстановок π.

*P-преобразование.* Функция перестановки. Для каждой пары байт из входной последовательности происходит замена одного байта другим.

*L-преобразование.* Представляет собой умножение 64-битного входного вектора на бинарную матрицу A размерами 64x64.

# Функция сжатия.

Основным элементом любой хеш-функции является функция сжатия.

Опишем используемую в новом стандарте функцию сжатия g n в виде алгоритма. Пусть h, N и m — 512-битные последовательности. Для вычисления функции g(N, m, h) необходимо проделать следующие шаги:

1. Вычислить значение K = h ⊕ N
2. Присвоить значение K = S(K)
3. Присвоить значение K = P(K)
4. Присвоить значение K = L(K)
5. Вычислить t = E(K, m)
6. Присвоить значение t = h ⊕ t
7. Вычислить значение G = t ⊕ m
8. Вернуть G в качестве результата вычисления функции g(N, m, h)

Функция E(K, m) выполняет нижеприведенные действия:

1. Вычислить значение state = K ⊕ m
2. Для i=0 по 11 выполнить:

– присвоить значение state = S(state);

– присвоить значение state = P(state);

– присвоить значение state = L(state);

– вычислить K=KeySchedule(K, i);

– присвоить значение state = state ⊕ K.

1. Вернуть state в качестве результата.

Функция KeySchedule(K, i) отвечает за формирование временного ключа

K на каждом раунде функции E(K, m) и производит следующие вычисления:

1. Присвоить значение K = K ⊕ C[i].
2. Присвоить значение K = S(K).
3. Присвоить значение K = P(K).
4. Присвоить значение K = L(K).
5. Вернуть K в качестве результата функции.

# Вычисление хеш-функции.

Для любого входного сообщения M:

1. Присвоить начальные значения внутренних переменных:

Для хеш-функции с длиной выхода 512 бит: h=iv=0x

Для хеш-функции с длиной выхода 256 бит: h=iv=0x

N =

Σ =

1. Проверить следующее условие:

длина сообщения M<512. Если условие выполняется перейти к пункту 3). В противном случае выполнить последовательность вычислений:

m — последние 512 бит сообщения M

h = g(N, m, h)

N = (N + 512) mod

Σ = (Σ + m) mod

Обрезать M, убрав последние 512 бит.

Перейти к шагу 2 .

1. Произвести дополнение сообщения M до длины в 512 бит по следующему правилу:

m = ||1||M, где |M| — длина сообщения M в битах.

Вычислить h = g(N, m, h).

Вычислить N = (N + |M|) mod

Вычислить Σ = (Σ + m) mod

Вычислить h = g(0, h, N)

Вычислить h = g(0, h, Σ)

Для хеш-функции с длиной выхода в 512 бит возвращаем h в качестве результата. Для функции с длиной выхода 256 бит возвращаем MSB 256 (h).

# Алгоритм вычисления хеш-функции SHA-1

Алгоритм SHA-1 (Secure Hash Algorithm) предложен Институтом Стандартизации США NIST как стандарт хеширования в гражданской криптографии. Этот алгоритм был призван дать еще больший запас прочности к криптоатакам.

SHA-1 реализует [хеш-функцию](https://ru.wikipedia.org/wiki/%D0%A5%D0%B5%D1%88%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения.

Выход представляет собой значение всех хеш-блоков до этого момента. Иными словами хеш блока Mi равен hi = f (Mi , hi−1 ). Хеш-значением всего сообщения является выход последнего блока.

Алгоритм получает на входе сообщение максимальной длины 264 бит и создает в качестве выхода *дайджест сообщения* длиной 160 бит.

**Шаг 1: добавление недостающих битов**

Сообщение добавляется таким образом, чтобы его длина была кратна 448 по модулю 512 ( ≡ 448 mod 512).

Добавление осуществляется всегда, даже если сообщение уже имеет нужную длину.

Таким образом, число добавляемых битов находится в диапазоне от 1 до 512.

Добавление состоит из единицы, за которой следует необходимое количество нулей.

**Шаг 2: добавление длины**

К сообщению добавляется блок из 64 битов. Этот блок трактуется как беззнаковое 64-битное целое и содержит длину исходного сообщения до добавления.

Результатом первых двух шагов является сообщение, длина которого кратна 512 битам.

Расширенное сообщение может быть представлено как последовательность 512-битных блоков Y0, Y1, . . . , YL-1, так что общая длина расширенного сообщения есть L \* 512 бит.

Таким образом, результат кратен шестнадцати 32-битным словам.

**Шаг 3: инициализация SHA-1 буфера**

Используется 160-битный буфер для хранения промежуточных и окончательных результатов *хeш-функции*.

Буфер может быть представлен как пять 32-битных регистров A, B, C, D и E.

Эти регистры инициализируются следующими шестнадцатеричными числами:

A = 67452301

B = EFCDAB89

C = 98BADCFE

D = 10325476

E = C3D2E1F0

**Шаг 4: обработка сообщения в 512-битных (16-словных) блоках**

Основой алгоритма является модуль, состоящий из 80 циклических обработок, обозначенный как HSHA.

Все 80 циклических обработок имеют одинаковую структуру.

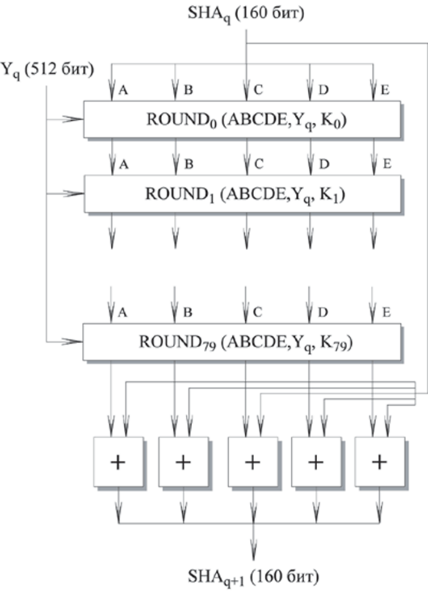


Рисунок 5,1 Обработка очередного 512-битного блока

Каждый цикл получает на входе текущий 512-битный обрабатываемый блок Yq и 160-битное значение буфера *ABCDE*, и изменяет содержимое этого буфера.

В каждом цикле используется дополнительная константа Кt, которая принимает только четыре различных значения:

0<=t<=19 **Kt=5A827999** (целая часть числа [230x21/2])

20<=t<=39 **Kt=6ED9EBA1** (целая часть числа [230x31/2])

40<=t<=59 **Kt=8F1BBCDC** (целая часть числа [230x51/2])

60<=t<=79 **Kt=CA62C1D6** (целая часть числа [230x101/2])

Для получения *SHA*q+1 выход 80-го цикла складывается со значением *SHA*q.

Сложение по модулю 232 выполняется независимо для каждого из пяти слов в буфере с каждым из соответствующих слов в *SHA*q.

**Шаг 5: выход**

После обработки всех 512-битных блоков выходом L-ой стадии является 160-битный *дайджест сообщения*.

Рассмотрим более детально логику в каждом из 80 циклов обработки одного 512-битного блока.

Каждый цикл можно представить в виде:

**A,B,C,D,E(CLS5(A)+ft(B, C, D)+E+Wt+Kt),A,CLS30(B),C,D**

где

|  |
| --- |
| A, B, C, D, E - пять слов из буфера. |
| t - номер цикла, 0 <= t <= 79. |
| ft - элементарная *логическая функция*. |
| CLSs - циклический левый сдвиг 32-битного аргумента на s битов. |
| Wt - 32-битное слово, полученное из текущего входного 512-битного блока. |
| Kt - дополнительная константа. |
| + - сложение по модулю 232. |

**Логика выполнения отдельного цикла SHA-1**

Каждая элементарная функция получает на входе три 32-битных слова и создает на выходе одно 32-битное слово.

Элементарная функция выполняет набор побитных логических операций, т.е. n-ый бит выхода является функцией от n-ых битов трех входов. Функции следующие:

|  |  |
| --- | --- |
| Номер цикла | *ft (B, C, D)* |
| (0 <= t <= 19) | *(B Λ C) ν (¬ B Λ D)* |
| (20 <= t <= 39) | *B ⊕ C ⊕ D* |
| (40 <= t <= 59) | *(B Λ C) ν (B Λ D) ν (C Λ D)* |
| (60 <= t <= 79) | *B ⊕ C ⊕ D* |

На самом деле используются только три различные функции.

Для 0 <= t <= 19 функция является условной:

if B then C else D.

Для 20 <= t <= 39 и 60 <= t <= 79 функция создает бит *четности*.

Для 40 <= t <= 59 функция является истинной, если два или три аргумента истинны.

Получение входных значений каждого цикла из очередного блока/

Двухбитные слова Wt получаются из очередного 512-битного блока сообщения следующим образом.

Первые 16 значений Wt берутся непосредственно из 16 слов текущего блока. Оставшиеся значения определяются следующим образом:

Wt= Wt-16 ⊕Wt-14 ⊕Wt-8 ⊕Wt-3

В первых 16 циклах вход состоит из 32-битного слова данного блока. Для оставшихся 64 циклов вход состоит из *XOR* нескольких слов из блока сообщения.

Алгоритм *SHA-1* можно суммировать следующим образом:

**SHA0=IV**

**SHAq+1=Σ32(SHAq, ABCDEq),**

Где:

**IV** - начальное значение буфера *ABCDE*;

***ABCDE*q** - результат обработки q-того блока сообщения;

**L**- число блоков в сообщении, включая поля добавления и длины;

**Σ32** - сумма по модулю 232, выполняемая отдельно для каждого слова буфера;

***SHA*** - значение *дайджеста сообщения*.

**ЗАДАНИЕ:**

1. Изучить теоретические сведения

2. Реализовать самостоятельно (без использования готовых библиотек и функций) программное средство контроля целостности сообщений с помощью вычисления хеш-функции и алгоритма ГОСТ 34.11 и SHA 1.