**УЧЕБНО-НАУЧНО-ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС**
**«МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ КЫРГЫЗСТАНА»**

«УТВЕРЖДЕНО»
Ректор НОУ УНПК «МУК»
к.т.н., доцент Савченков Е.Ю.
_____
«16» 10 2018 г.

**БАКАЛАВРИАТ**

**Кафедра «Компьютерные информационные системы и управление»**

**Учебно-методический комплекс дисциплины**

# Защита информации

Направление: **710100 «Информатика и вычислительная техника»**
Профиль: **Компьютерные информационные системы в бизнесе**
Академическая степень - **бакалавр**
Форма обучения **(очная)**

**График проведения модулей 6-семестр**

| **неделя** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **лекц. зан.** | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| **прак./лаб. зан.** | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

**«РАССМОТРЕНО»**
Протокол заседания кафедры
«КИСиУ»
№ 2 от 16.10.2018
Зав. кафедрой д.т.н., проф. Миркин Е.Л.
_____

**«СОГЛАСОВАНО»**
Проректор по академ. вопросам
проф. Мадалиев М.М.
_____

Составитель _____ к.ф.-м..н., и.о.доцент Красниченко Л.С.

Директор Научной библиотеки _____ Асанова Ж.Ш.

БИШКЕК 2018

ОГЛАВЛЕНИЕ

**АННОТАЦИЯ 3**

**УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ (МОДУЛЕЙ) 4**

**1. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА 4**
1.1. Миссия и стратегия 4
1.2. Цель и задачи дисциплины 4
1.3. Формируемые компетенции, а также перечень планируемых результатов обучения по дисциплине 4
1.4. Место дисциплины (модулей) в структуре ООП ВПО 5

**2. СТРУКТУРА ДИСЦИПЛИНЫ 5**

**3. СОДЕРЖАНИЕ ДИСЦИПЛИНЫ 6**

**4. КОНСПЕКТ ЛЕКЦИЙ – ПРИЛОЖЕНИЕ 1 7**

**5. ИНФОРМАЦИОННЫЕ И ОБРАЗОВАТЕЛЬНЫЕ ТЕХНОЛОГИИ 7**

**6. ФОНД ОЦЕНОЧНЫХ СРЕДСТВ ДЛЯ ТЕКУЩЕГО, РУБЕЖНОГО И ИТОГОВОГО КОНТРОЛЕЙ ПО ИТОГАМ ОСВОЕНИЮ ДИСЦИПЛИНЫ (МОДУЛЕЙ) 8**
6.1. Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины 8
**6.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности** 9
**6.1. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания** 11
**6.2. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности.** 12
Контрольные вопросы 12
Тематика рефератов 13
Самостоятельная работа студентов 15

**7. УЧЕБНО-МЕТОДИЧЕСКОЕ И ИНФОРМАЦИОННОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ 16**
7.1. Список источников и литературы 16
7.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей) 17

**8. ПЕРЕЧЕНЬ УЧЕБНО-МЕТОДИЧЕСКОГО ОБЕСПЕЧЕНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ ОБУЧАЮЩИХСЯ 17**

**9. МАТЕРИАЛЬНО-ТЕХНИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ДИСЦИПЛИНЫ 20**

**10. ГЛОССАРИЙ 20**

**ПРИЛОЖЕНИЯ 21**

# АННОТАЦИЯ

Дисциплина «Защита информации» представляет собой специализированный курс, который является одним из важнейших при подготовке специалистов в области информационных технологий. Данный курс дает полное представление специалистам об областях применения алгоритмов шифрования данных, о существующих стандартах цифровой подписи и других способах применения симметричных и ассиметричных алгоритмов шифрования. Данный курс состоит из трех модулей, итоговый контроль в виде экзамена.

**УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ (МОДУЛЕЙ)**

# 1. Пояснительная записка

## 1.1. Миссия и стратегия

Миссия НОУ УНПК "МУК" – подготовка международно - признанных, свободно мыслящих специалистов, открытых для перемен и способных трансформировать знания в ценности на благо развития общества.

Видение НОУ УНПК «МУК»- создание динамичного и креативного университета с инновационными научно-образовательными программами и с современной инфраструктурой, способствующие достижению академических и профессиональных целей.

Стратегии развития - модернизация образовательной деятельности университета – совершенствование образовательного процесса в соответствии с требованиями Болонского процесса.

## 1.2. Цель и задачи дисциплины

*Цель дисциплины:*

Целью преподавания дисциплины является ознакомление с организационными, техническими, алгоритмическими и другими методами и средствами защиты компьютерной информации, с законодательством и стандартами в этой области, с современными криптосистемами.

*Задачи дисциплины:*

Изучение курса должно обеспечить:

1. Сформировать взгляд на криптографию и защиту информации как на систематическую научно-практическую деятельность, носящую прикладной характер.
2. Сформировать базовые теоретические понятия (возможно, на элементарном уровне), лежащие в основе процесса защиты информации.
3. Дать представление о роли компьютера, как о центральном месте в области криптографии, взявшем на себя большинство функций традиционной компьютерной
деятельности, включающей реализацию криптографических алгоритмов, проверку их
качества, генерацию и распределение ключей, автоматизацию работы по анализу перехвата и раскрытию шифров.
4. Научить использованию криптографических алгоритмов в широко распространенных
программных продуктах.

## 1.3. Формируемые компетенции, а также перечень планируемых результатов обучения по дисциплине

Дисциплина направлена на формирование следующих компетенций:

- способен понимать и применять традиционные и инновационные идеи, находить подходы к их реализации и участвовать в работе над проектами, используя базовые методы исследовательской деятельности (ОК-4);
- способен готовить конспекты и проводить занятия по обучению сотрудников применению программно-методических комплексов, используемых на предприятии (ПК-8).
- способен сопрягать аппаратные и программные средства в составе информационных и автоматизированных систем (ПК-10);

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

1. *Знать:*
   математические основы криптографии, организационные, технические и программные методы защиты информации в современных компьютерных системах и сетях, стандарты, модели и методы шифрования, методы идентификации пользователей, методы защиты программ от вирусов, основы инфраструктуры систем, построенных с с использованием публичных и секретных ключей;
2. *Уметь:*
   применять известные методы и средства поддержки информационной безопасности в компьютерных системах, проводить сравнительный анализ, выбирать методы и средства, оценивать уровень защиты информационных ресурсов в прикладных системах;
3. *Владеть:*
   Навыками применения методов средств защиты информации и управления правами использования информационных ресурсов при передаче конфиденциальной информации по каналам связи, установлении подлинности передаваемых сообщений, хранении информации (документов, баз данных), встраивании скрытой служебной информации.

### 1.4. Место дисциплины (модулей) в структуре ООП ВПО

Дисциплина «Защита информации» является частью профессионального цикла (блока) дисциплин учебного плана по направлению подготовки 710100 «Информатика и вычислительная техника» подготовки бакалавров (специализации Компьютерные информационные системы).

Для освоения дисциплины (модулей) необходимы компетенции, сформированные в ходе изучения следующих дисциплин и прохождения практик: основные разделы математики, программирования.

В результате освоения дисциплины (модулей) формируются компетенции, необходимые для изучения следующих дисциплин и прохождения практик: написание выпускной квалифицированной работы.

## 2. Структура дисциплины

Общая трудоемкость дисциплины составляет 3 кредита, 90ч., в том числе аудиторная работа обучающихся с преподавателем 48ч., самостоятельная работа обучающихся 42ч. из них 14ч-СРСП.

| Наименование разделов и тем | *очная форма обучения* | | | | |
|---|---|---|---|---|---|
| | Количество часов | | | | |
| | Лек | Прак | СРС | СРСП | Всего часов по теме |
| **Раздел 1 Криптографические методы.** | **4** | **8** | **8** | **4** | **24** |
| ***Тема 1.*** Основные понятия и определения информацией | 2 | 4 | 5 | 2 | 13 |
| ***Тема 2.*** Типы криптографических систем. | 2 | 4 | 3 | 2 | 11 |
| **Раздел 2 Симметричные криптографические системы.** | **6** | **12** | **12** | **5** | **35** |
| ***Тема 3.*** Стандарт шифрования DES. | 2 | 4 | 4 | 2 | 12 |
| ***Тема 4.*** Стандарт шифрования данных ГОСТ 28147-89. | 2 | 4 | 4 | **1** | 11 |
| ***Тема 5.*** Блочные алгоритмы. | 2 | 4 | 4 | 2 | 12 |
| **Раздел 3. Асимметричные криптографические системы.** | **2** | **4** | **3** | **2** | **11** |
| **Раздел 4. Задача обмена ключами** | **2** | **4** | **3** | **1** | **10** |
| **Раздел 5. Цифровая электронная подпись** | **2** | **4** | **3** | **2** | **10** |
| | **16** | **32** | **28** | **14** | **90** |

## 3. Содержание дисциплины

### Раздел 1. Криптографические методы.

***Тема 1.*** Основные понятия и определения.

Понятие криптографического протокола. Основные типы протоколов. Классы преобразований: подстановки, перестановки, гаммирование, блочные шифры. Датчики ПСЧ.

***Тема 2.*** Типы криптографических систем.

Симметричная криптография. Асимметричная криптография. Цифровой дайджест и хэш-функция. Подстановочные и перестановочные шифры. Шифры Цезаря, Вижинера, Вернома. Исследования Шеннона в области криптографии. Нераскрываемость шифра Вернома.

### Раздел 2. Симметричные криптографические системы.

***Тема 3.*** Стандарт шифрования DES.

Алгоритм, скорость работы на различных платформах, режимы пользования, основные результаты по анализу стойкости.

***Тема 4.*** Стандарт шифрования данных ГОСТ 28147-89.

Алгоритм, скорость работы на различных платформах, режимы пользования.

***Тема 5.*** Блочные алгоритмы.

Алгоритм Blowfish. Потоковые алгоритмы. Алгоритм PKZIP. Теоретическая и практическая стойкость.

### Раздел 3. Асимметричные криптографические системы.

Системы с открытым ключом. Алгоритм шифрования RSA. Вычислительные аспекты реализации алгоритма RSA. Вопросы стойкости. Криптосистема Эль-Гамаля. Криптосистемы на основе эллиптических уравнений.

### Раздел 4. Алгоритм обмена ключами

Алгоритм Диффи-Хеллмана. Протоколы обмена ключами на основе алгоритма Диффи-Хеллмана: двусторонний и многосторонний протокол.

### Раздел 5. Цифровая электронная подпись

Проблема аутентификации данных и электронная цифровая подпись. Однонаправленные хэш-функции. Алгоритм безопасного хэширования SHA. Однонаправленные хэш-функции на основе симметричных блочных алгоритмов. Электронная подпись на основе алгоритма RSA. Алгоритм цифровой подписи Эль-Гамаля (EGSA). Алгоритм цифровой подписи DSA.

## 4. Конспект лекций – ПРИЛОЖЕНИЕ 1

## 5. Информационные и образовательные технологии

| № п/п | Наименование раздела | Виды учебной работы | Формируемые компетенции (указывается код компетенции) | Информационные и образовательные технологии |
|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 |
| 1 | **Криптографические методы.** | Лекция<br><br>Лабораторная работа.<br><br>Самостоятельная работа | (ПК-8)<br><br>(ПК-10)<br><br>(ОК-4) | **Лекция-визуализация** с применением проектора **Лабораторная работа** зашифровать открытый текст произвольной длины рассмотренными методами шифрования. Оценить крипто стойкость и разобрать эффективный алгоритм для его раскрытия **Подготовка к занятию** с использованием электронного курса лекций |
| 2 | **Симметричные криптографические системы.** | Лекция<br><br>Лабораторная работа<br><br>Самостоятельная работа | ПК-8)<br><br>(ПК-10)<br><br>(ОК-4) | **Лекция-визуализация** с применением проектора **Лабораторная работа** в Разработать DES - подобную структуру. Оценить скорость работы алгоритма **Подготовка к занятию** с использованием |

| | | | | |
|---|---|---|---|---|
| | | | | электронного курса лекций |
| 3 | **Асимметричные криптографические системы.** | Лекция<br><br>Лабораторная работа<br><br>Самостоятельная работа | ПК-8)<br><br>(ПК-10)<br><br>(ОК-4) | **Лекция-визуализация** с применением проектора **Лабораторная работа** в Разработка эффективного алгоритма генерации больших чисел **Подготовка к занятию** с использованием электронного курса лекций |
| 5 | **Алгоритм обмена ключами** | Лекция<br><br>Лабораторная работа<br><br>Самостоятельная работа | ПК-8)<br><br>(ПК-10)<br><br>(ОК-4) | **Лекция-визуализация** с применением проектора **Лабораторная работа** в Разработка алгоритма и программная реализация атаки на симметричную криптографическую систему **Подготовка к занятию** с использованием электронного курса лекций |
| 6 | **Цифровая электронная подпись** | Лекция<br><br>Лабораторная работа<br><br>Самостоятельная работа | (ПК-8)<br><br>(ПК-10)<br><br>(ОК-4) | **Лекция-визуализация** с применением проектора **Лабораторная работа** в Разработка алгоритмов реализации цифровой подписи для известных криптосистем **Подготовка к занятию** с использованием электронного курса лекций |

## 6. Фонд оценочных средств для текущего, рубежного и итогового контролей по итогам освоению дисциплины (модулей)

### 6.1.Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины

| *№ п/п* | *Контролируемые разделы дисциплины (модулей)* | *Код контролируемой компетенции (компетенций)* | *Наименование оценочного средства* |
|---|---|---|---|

| | | | |
|---|---|---|---|
| *1* | **Криптографические методы.** | (ПК-8)<br>(ПК-10)<br>(ОК-4) | Практическое задание (лабораторная)<br>Коллоквиум (Вопросы по разделам дисциплины) |
| *2* | **Симметричные криптографические системы.** | (ПК-8)<br>(ПК-10)<br>(ОК-4) | Практическое задание (лабораторная)<br>Коллоквиум (Вопросы по разделам дисциплины)<br>Темы рефератов |
| *3* | **Асимметричные криптографические системы.** | (ПК-8)<br>(ПК-10)<br>(ОК-4) | Практическое задание (лабораторная)<br>Коллоквиум (Вопросы по разделам дисциплины)<br>Тест |
| *4* | **Алгоритм обмена ключами** | (ПК-8)<br>(ПК-10)<br>(ОК-4) | Практическое задание (лабораторная)<br>Коллоквиум (Вопросы по разделам дисциплины) |
| *5* | **Цифровая электронная подпись** | (ПК-8)<br>(ПК-10)<br>(ОК-4) | Практическое задание (лабораторная)<br>Коллоквиум (Вопросы по разделам дисциплины) |

### 6.2.Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Методические материалы составляют систему текущего, рубежного и итогового (экзамена) контролей освоения дисциплины (модулей), закрепляют виды и формы текущего, рубежного и итогового контролей знаний, сроки проведения, а также его сроки и формы проведения (устный экзамен, письменный экзамен и т.п.). В системе контроля указывается процедура оценивания результатов обучения, при использовании балльно-рейтинговой системы приводится таблица с баллами и требованиями к пороговым значениям достижений по видам деятельности обучающихся; показывается механизм получения оценки (из чего складывается оценка по дисциплине (модулю).

**Текущий контроль** осуществляется в виде опроса, участие в дискуссии на семинаре, выполнение самостоятельной - оценивается до **80 баллов**.

**Рубежный контроль** (сдача модулей) проводится преподавателем и представляет собой письменный контроль, либо компьютерное тестирование знаний по теоретическому и практическому материалу. Контрольные вопросы рубежного контроля включают полный объём материала части дисциплины (модулей), позволяющий оценить знания, обучающихся по изученному материалу и соответствовать УМК дисциплины, которое оценивается до **20 баллов**.

**Итоговый контроль** (экзамен) знаний принимается по экзаменационным билетам, включающий теоретические вопросы и практическое задание, и оценивается до **20 баллов**.

| Форма контроля | Срок отчетности | Макс. количество баллов | |
|---|---|---|---|
| | | За одну работу | Всего |
| Текущий контроль:<br>- Прием лабораторных работ<br>-опрос<br>- посещаемость | <br>1,2,3,4,5,6,7,8 недели<br>1, ,2,3,4,5,6,7,8 недели<br>1, ,2,3,4,5,6,7,8 неделя | <br>8 баллов<br>6 баллов<br>2 балла | <br>До 40 баллов<br>До 30 баллов<br>10 баллов |
| Рубежный контроль:<br>(сдача модуля) | 8 неделя | 100%×0,2=20 баллов | |
| Итого за I модуль | | | До 100 баллов |

| Форма контроля | Срок отчетности | Макс. количество баллов | |
|---|---|---|---|
| | | За одну работу | Всего |
| Текущий контроль:<br>- Прием лабораторных работ<br>-опрос<br>- посещаемость | <br>9,10,11, недели<br>9,10,11, недели<br>9,10,11, недели | <br>10 баллов<br>6 баллов<br>2 балла | <br>До 40 баллов<br>До 30 баллов<br>10 баллов |
| Рубежный контроль:<br>(сдача модуля) | **11 неделя** | 100%×0,2=20 баллов | |
| Итого за II модуль | | | До 100 баллов |

| Форма контроля | Срок отчетности | Макс. количество баллов | |
|---|---|---|---|
| | | За одну работу | Всего |
| Текущий контроль:<br>- Прием лабораторных работ<br>-опрос<br>- посещаемость | <br>12, 13, 14 недели<br>12, 13, 14 недели<br>12, 13, 14 неделя | <br>8 баллов<br>6 баллов<br>2 балла | <br>До 40 баллов<br>До 30 баллов<br>10 баллов |
| Рубежный контроль:<br>(сдача модуля) | **14 неделя** | 100%×0,2=20 баллов | |
| Итого за III модуль | | | До 100 баллов |

Экзаменатор выставляет по результатам балльной системы в семестре экзаменационную оценку без сдачи экзамена, набравшим суммарное количество баллов, достаточное для выставления оценки от 55 и выше баллов – автоматически (при согласии обучающегося).

Полученный совокупный результат (максимум 100 баллов) конвертируется в традиционную шкалу:

| Рейтинговая оценка (баллов) | Оценка экзамена |
|---|---|
| От 0 - до 54 | неудовлетворительно |
| от 55 - до 69 включительно | удовлетворительно |
| от 70 – до 84 включительно | хорошо |
| от 85 – до 100 | отлично |

### 6.3.Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

***Текущий контроль (0 - 80 баллов)***

При оценивании посещаемости, опроса и приема лабораторных работ из расчета на одну неделю учитываются:

- посещаемость (2 балла одно занятие (10 баллов за модуль)
- степень раскрытия содержания материала (2.8 балла одно занятие (14 баллов за модуль);
- изложение материала (грамотность речи, точность использования терминологии и символики, логическая последовательность изложения материала (2.8 балла одно занятие (14 баллов за модуль);
- знание теории изученных вопросов (2.8 балла одно занятие (14 баллов за модуль);
- сформированность и устойчивость используемых при ответе умений и навыков (2.8 балла одно занятие (14 баллов за модуль);
-точность решения задачи (2.8 балла одно занятие (14 баллов за модуль).

***Рубежный контроль (0 – 20 баллов)***

**При оценивании контрольной работы учитывается:**

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – 8 баллов;
- обоснованность содержания и выводов работы (задание выполнено полностью, но обоснование содержания и выводов недостаточны, но рассуждения верны) – 14 баллов;
- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок, возможна одна неточность - 17 баллов.
- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок - 20 баллов.

**При оценивании теста учитывается:**

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – до 20 баллов;

***Итоговый контроль (экзаменационная сессия)*** - ИК = Бср × 0,8+Бэкз × 0,2

При проведении итогового контроля обучающийся должен ответить на 3 вопроса (два вопроса теоретического характера и один вопрос практического характера).

При оценивании ответа на вопрос теоретического характера учитывается:

- теоретическое содержание не освоено, знание материала носит фрагментарный характер, наличие грубых ошибок в ответе (2 балла);
- теоретическое содержание освоено частично, допущено не более двух-трех недочетов (5 баллов);
- теоретическое содержание освоено почти полностью, допущено не более одного-двух недочетов, но обучающийся смог бы их исправить самостоятельно (8 баллов);
- теоретическое содержание освоено полностью, ответ построен по собственному плану (10 баллов).

При оценивании ответа на вопрос практического характера учитывается:

- ответ содержит менее 20% правильного решения (3 балла);
- ответ содержит 21-89 % правильного решения (7 баллов);
- ответ содержит 90% и более правильного решения (10 баллов).

### 6.4.Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности.

Раздел УМК включает образцы оценочных средств, примерные перечни вопросов и заданий в соответствии со структурой дисциплины и системой контроля.

**Контрольные вопросы**

1. Криптография. Основные разделы современной криптографии.
2. Схемы шифрования и дешифрования. Ключ. Типы криптосистем.
3. Требования к криптосистемам.
4. Криптоанализ. Уровни атак. Криптостойкость.
5. Классы симметричных криптосистем.
6. Блочные шрифты. Перестановки.
7. Системы подстановок.
8. Подстановка Цезаря.
9. Многоалфавитные подстановки.
10. Гаммирование.
11. Структура итерации сети Фейстеля.
12. Схема алгоритма DES.
13. DES. Функция сжатия.
14. DES. Функция расширения.
15. DES. Генерация ключа.
16. Схема шифрования алгоритма ГОСТ 28147-89.
17. ГОСТ 28147-89.Режим шифрования ключевой информации.
18. ГОСТ 28147-89. Режим гаммирования.
19. ГОСТ 28147-89. Режим гаммирования с обратной связью.
20. Стандарт AES. Алгоритм Rijndal. Схема.
21. AES. Преобразование ByteSub.
22. AES. Преобразование ShiftRow.
23. AES. Преобразование MixColumn.
24. AES. Преобразование AddRoundKey. Последняя итерация.
25. AES. Алгоритм обработки ключа.
26. Ассиметричные криптосистемы.

27. Криптосистема Эль-Гамаля.
28. Криптосистема RSA.
29. Цифровые подписи.
30. Функция хеширования MD5.

**Тематика рефератов**

***3-Way***

Простой и быстрый криптоалгоритм. Автор - Joan Daemen. Длинна ключа 96 бит. Длинна обрабатываемых блоков данных также 96 бит. David Wagner, John Kelsey, и Bruce Schneier открыли так называемуя related key атаку, которая требует всего 1 связанного запроса и 222 известных символа. Алгоритм не запатентован.

***Blowfish***

Создатель - автор наиболее популярной книги по криптографии "Applied Cryptography", Bruse Schneier. Длинна ключа 32-448 бит. Длинна обрабатываемых блоков - 64 бит. Число раундов - 16. Считается одним из наиболее сильных криптоалгоритмов. Для выработки ключей используется 521 цикл шифрования, что значительно затрудняет атаку приямым перебором. Эта же особенность делает затруднительной и возможность применения алгоритма в системах, где ключи меняются очень часто.

***CAST***

Авторы - Carlisle Adams и Stafford Taveres. Алгоритм очень похож на Blowfish. Длинна ключа 40...64 бит. Количество проходов 8. Известна related key атака на 64-битный CAST, открытая David Wagner, John Kelsey, и Bruce Schneier. Других атак кроме прямого перебора нет. Вариант алгоритма CAST-256 был выставлен на конкурс AES, но до финала не дошел. Запатенотован, но (!) открыт для свободного использования.

***CMEA***

*Алгоритм, который используется в сотовых телефонах для шифрования служебных каналов. Был взломан* by David Wagner, John Kelsey, and Bruce Schneier of Counterpane Systems.

***CRYPTON***

***DEAL***

*Интересный алгоритм, который был предложен на конкурс AES. Оперирует 128-битовыми блоками данных и ключами длинной 128, 192 и 256 бит. Использует DES во внутренних итерациях. Для обеспечения сносной криптостойкости требуется не менее 8 итераций. Существуют атаки, которые успешно вскрывают шифр, если количество итераций меньше указанного значения.*

***FEAL***

*Крайне ненадежный криптоалгоритм. Взломан! Разработан* by the Nippon Telephone & Telegraph более быстрая, но менее надежная альтернатива DES. Известно множество атак, каждая из которых требует минимума информации.

***IDEA***

International Decryption-Encryption Algorithm - разработанный в Щвейцарии (Цюрих) считается очень стойким криптоалгоритмом, хотя полной проверки еще не прошел. Авторы: Xuejia Lai, James Massey.Размер блока данных 64 бит. Длинна ключа 128 бит. Число раундов - 8. Оперирует 16-битовыми подюлоками. Оптимизирован для 16-битных процессоров. Запатентован в США, Европе и Японии. Свободен для некоммерческого применения. В остальных случаях необходима лицензия компании Аском (владелец патента)

***MARS***

*Новый симметричный блочный алгоритм от создателей DES. Один из $5^{mu}$ финалистов AES. Разработан исследователями фирмы IBM. Оперирует 128-битовыми блоками. Длинна ключа переменная. Скорость работы - 65 Мбит/с на Pentium Pro 200 и 85 Мбит на 200MHz Power PC. Есть аппаратные реализации данного алгоритма.*

***RC2***

*Блочный шифр разработанный Роном Ривестом для RSA Data Security. Криптостойкость считается очень высокой. Размер блока 64 бит. Длинна ключа переменная. Скорость работы примерно вдвое быстрее чем DES. Является собственностью RSA Data Security.*

***RC5***

*Мощный и быстрый блочный симметричный криптоалгоритм разработанный Ривестом для RSA Data Security. Оперирует блоками данных длинной 32, 64 или 128 бит. Длинна ключа переменна. Количество проходов 0...255. Длинна ключа 0...2048 бит. Скорость алгоритма обусловлена применением только трех простых и быстрых операций - сложения, XOR и чередования (rotation). Был объявлен конкурс на взлом этого алгоритма. За три года были взломаны только слабо зашифрованные тексты (с малыми длинной ключа и количеством проходов). Собственность RSA Data Security.*

***Serpent***

128 битный алгоритм designed by Ross Anderson, Eli Biham and Lars Knudsen. Скорость в 5 раз быстрее DES.

***Skipjack***

*Разработка АНБ США. До недавнего времени был секретным. Достаточно стоек. Размер блока 64 бит. Длинна ключа 80 бит. 32 нелинейных комплексных раунда. Создан чип, который шифроует данные по этому алгоритму.*

***Twofish***

*Великолепный алгоритм, пришедший на смену Blowfish. Автор - все тот же Брюс Шайнер. Скорость работы может меняться в зависимости от глубины проведенной прекомпиляции ключа. Цитата с сайта Counterpane Internet Security Inc:*

***Tripple DES***

*Это ответ на вопрос "что будет если несколько раз зашифровать/расшифровать данные с разными кллючами?" Этот алгоритм позволяет используя DES увеличить его криптостойкость путем дифрования/дешифрования/шифрования по алгоритму DES с разными ключами. Объединение этих ключей в один дает длинну 3*56=168 бит. Но DES он и в Африке DES...*

***SHA1***

*Secure Hash Algorithm. Создает 160-битовое значение хэш-функции из исходных данных переменного размера. Предложена NIST и принята правительством США как стандарт. Используется в стандарте DSS.*

***ГОСТ Р34.11-94***

***DSA***

Digital Signature Authorization. Алгоритм, использующий открытые ключи для создания электронной подписи. Секретное создание хэш-значения и публичная проверка ее - только один человек может создать хэш-значение сообщения, но любой может проверить ее корректность. Основан на вычислительной сложности взятия логарифмов в конечных полях.

***RSA***

Отправитель создает хэш-функцию сообщения, а затем шифрует ее с использованием своего секретного ключа. Получатель использует открытый ключ отправителя для расшифровки хэша, сам рассчитывает хэш для сообщения, и сравнивает эти два хэша. Запатентовано RSA Data Security.

***MAC*** (Код аутентификации сообшения)

Электронная подпись, использующая схемы хэширования, аналогичные MD или SHA, но хэш-значение вычисляется с использованием как данных сообщения, так и секретного ключа.

***DTS*** (служба электронных временных меток)

Выдает пользователям временные метки, связанные с данными документа,

криптографически стойким образом.

**Самостоятельная работа студентов**

Самостоятельная работа включает:

- ✓ **Реферат:** Продукт самостоятельной работы студента, представляющий собой краткое изложение в письменном виде полученных результатов теоретического анализа определенной научной (учебно-исследовательской) темы, где автор раскрывает суть исследуемой проблемы, приводит различные точки зрения, а также собственные взгляды на нее.
- ✓ **Доклад:** Продукт самостоятельной работы студента, представляющий собой публичное выступление по представлению полученных результатов решения определенной учебно-практической, учебно-исследовательской или научной темы

**Подготовка доклада к занятию.**

Основные этапы подготовки доклада:

- ✓ выбор темы;
- ✓ консультация преподавателя;
- ✓ подготовка плана доклада;
- ✓ работа с источниками и литературой, сбор материала;
- ✓ написание текста доклада;
- ✓ оформление рукописи и предоставление ее преподавателю до начала доклада, что определяет готовность студента к выступлению;
- ✓ выступление с докладом, ответы на вопросы.
- ✓ Тематика доклада предлагается преподавателем в ФОС.

**Рекомендации по написанию реферата.**

✓ Тема реферата выбирается в соответствии с Вашими интересами и не обязательно должна соответствовать приведенному ниже примерному перечню. Важно, чтобы в реферате: во-первых, были освещены как естественнонаучные, так и социальные стороны проблемы; а во-вторых, представлены как общетеоретические положения, так и конкретные примеры. Особенно приветствуется использование собственных примеров из окружающей Вас жизни.

✓ Реферат должен основываться на проработке нескольких дополнительных к основной литературе источников. Как правило, это специальные монографии или статьи. Во многих регионах регулярно издаются Доклады о состоянии окружающей среды. Рекомендуется использовать также в качестве дополнительной литературы научно-популярные журналы: "Природа", "Наука и жизнь", "Химия и жизнь", "Энергия" и др, а также газеты специализирующиеся на природоохранной тематике.

✓ План реферата должен быть авторским. В нем проявляется подход автора, его мнение, анализ проблемы.

✓ Все приводимые в реферате факты и заимствованные соображения должны сопровождаться ссылками на источник информации. *Например: ... Нас заинтересовало снижение рождаемости, зарегистрированное в последнее время в России (Население России, 2008)...* или *... Установлено, что в крупных городах, таких как Москва, уровень загрязнения воздуха в некоторые часы может превышать предельно допустимые концентрации в 10 и более раз (Лихачева, Смирнова, 2006) ...*

- ✓ Недопустимо просто скомпоновать реферат из кусков заимствованного текста. Все цитаты должны быть представлены в кавычках с указанием в скобках источника и страницы, например: *"Проанализировав историю человечества за 2400 лет, А.Л.Чижевский установил связь между циклами исторических событий и солнечной активностью, причем равны они в среднем, 11 годам." (Лупачев, 1995, с.39).* Отсутствие кавычек и ссылок означает плагиат и, в соответствии с установившейся научной этикой, считается грубым нарушением авторских прав.
- ✓ Реферат оформляется в виде текста на листах стандартного формата (А- 4). Начинается с титульного листа, в котором указывается название вуза, учебной дисциплины, тема реферата, фамилия и инициалы студента, номер академической группы или название кафедры, год и географическое место местонахождения вуза. Затем следует оглавление с указанием страниц разделов. Сам текст реферата желательно подразделить на разделы: главы, подглавы и озаглавить их. Приветствуется использование в реферате количественных данных и иллюстраций (графики, таблицы, диаграммы, рисунки).
- ✓ Завершают реферат разделы "Заключение" и "Список использованной литературы". В заключении представлены основные выводы, ясно сформулированные в тезисной форме и, обычно, пронумерованные.
- ✓ Список литературы должен быть составлен в полном соответствие с действующим стандартом (правилами), включая особую расстановку знаков препинания. Для этого достаточно использовать в качестве примера любую книгу изданную крупными научными издательствами: "Наука", "Прогресс", "Мир", "Издательство МГУ" и др. Или приведенный выше список литературы. В общем случае наиболее часто используемый в нашей стране порядок библиографических ссылок следующий:
- ✓ *Автор И.О. Название книги. Место издания: Издательство, Год издания. Общее число страниц в книге.*
- ✓ *Автор И.О. Название статьи // Название журнала. Год издания. Том__. № __. Страницы от __ до ___.*
- ✓ *Автор И.О. Название статьи / Название сборника. Место издания: Издательство, Год издания. Страницы от __ до ___.*

## 7. Учебно-методическое и информационное обеспечение дисциплины

### 7.1.Список источников и литературы

*Основные учебники*


1. Гультяева Т. А. Основы теории информации и криптографии// Методические указания к лабораторным работам по курсу " Основы теории информации и криптографии".
2. К. Шеннон, Теория связи в секретных системах.
3. Ю. С. Харин, Математические и компьютерные основы криптологии, Минск, ООО Новое знание, 2003
4. Б. Шнайер, Прикладная криптография. Протоколы, алгоритмы и исходные тексты на языке С, 2-ое издание
5. В. М. Фомичев, Дискретная математика и криптология, Москва, Диалог Мифи, 2003
6. А.В.Черемушкин, Лекции по теоретико-числовым алгоритмам в криптографии, Москва, 2002
7. А.А. Терехов, Криптографическая защита информации, СПб, 1999

8. Введение в криптографию (под ред. В.В. Ященко), СПб, Питер, 2001
9. О. Н. Василенко, Теоретико-числовые алгоритмы в криптографии, Москва, 2003
10. В.Н. Потапов, Теория информации. Кодирование дискретных вероятностных источников, Новосибирск, 1999
11. В.И. Шульгин, Основы теории передачи информации (Часть 1: Экономное кодирование), Харьков, 2003
12. А.А.Ожиганов, М.В.Тарасюк, Передача данных по дискретным каналам (учебно-методическое пособие по дисциплине "Теория информации"), Санкт-Петербург. 1999

*Дополнительная литература*

1. Богуш Р.П., Элементы теории информации (конспект лекций), 2003
2. Д. В. Ломакин, Прикладная теория информации (конспект лекций), Н. Новгород ,1998
3. В.В. Лидовский, Теория информации, 2002
4. Б. А. Погорелов, А.В. Черемушкин, С. И. Чечета, Об определении основных криптографических понятий
5. Е. Б. Цой, И. В. Самочернов, Моделирование и управление в экономике (часть **I),** Новосибирск, 2003
6. С. В. Сизый, Лекции по теории чисел, Екатеринбург, 1999
7. С. Коутинхо, Введение в теорию чисел. Алгоритм RSA. Москва, 2001
8. А. В. Асосков, М. А. Иванов, А. А. Мирский, Поточные шифры, Москва, 2003

### 7.2.Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей)

1. https://www.intuit.ru/search
2. https://www.twirpx.com/ Библиотека все для студента
3. https://uk.sagepub.com/en-gb/asi/home
4. https://uk.sagepub.com/en-gb/asi/sage-premier
5. https://www.nejm.org/
6. https://uk.sagepub.com/en-gb/asi/imeche
7. http://global.oup.com/?cc=kg
8. https://www.cambridge.org
9. https://www.intellectbooks.co.uk/journals/index/
10. http://iopscience.iop.org/journalList
11. https://royalsociety.org/journals/
12. https://www.elibrary.imf.org/?redirect=true
13. https://www.elgaronline.com/page/70/journals
14. http://www.dukejournals.org/
15. http://www.iprbookshop.ru/
16. http://kyrlibnet.kg/ru/
17. http://biblioteka.kg/

## 8. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся

### 8.1.Планы практических (семинарских) и лабораторных занятий.

| №№ и названия разделов и тем (в соответствии с п . 3.3) | Цель и содержание лабораторной работы | Результаты лабораторной работы |
|---|---|---|
| Лабораторная работа № **1: Простые шифры** | | |
| **Раздел 1** ***Тема 1,.*** | Простые шифры. Разработка и анализ простых криптографических алгоритмов на основе методов перестановок и подстановок. | Зашифровать открытый текст произвольной длины рассмотренными методами шифрования. Оценить криптостойкость и разобрать эффективный алгоритм для его раскрытия |
| Лабораторная работа № 2: **Метод гаммирования.** | | |
| **Раздел 1** ***Тема 2.*** | Генерация псевдослучайных последовательностей чисел в системах защиты информации. Оценка статистических характеристик датчика псевдослучайных чисел с заданным законом распределения | Разработать алгоритм атаки на криптосистему, построенную методом гаммирования |
| Лабораторная работа № 3: **Симметричная криптография.** | | |
| **Раздел 2.** ***Тема 3. Тема 4.*** | Разработка и реализация варианта симметричного<br><br>криптографического алгоритма | Разработать DES - подобную структуру. Оценить скорость работы алгоритма |
| Лабораторная работа № 4: **Криптоанализ** | | |
| **Раздел 2 .** ***Тема 3-5.*** | Проведение сравнительного анализа систем ГОСТ 28147-89 и DES | Разработка алгоритма и программная реализация атаки на симметричную криптографическую систему |
| Лабораторная работа № 5 **Асимметричные криптографические системы.** | | |
| **Раздел 3** | Программная реализация алгоритма RSA | Разработка эффективного алгоритма генерации больших чисел |
| Лабораторная работа № 6 **Алгоритм обмена ключами** | | |
| **Раздел** 4 | Разработка и программная реализация протокола обмена симметричными ключами на основе<br><br>алгоритма Diffyie-Hellman | Модифицировать криптосистему Diffyie-Hellman на случай, если число участников больше или<br><br>равно трем. |
| Лабораторная работа № 7 **Цифровая электронная подпись** | | |
| **Раздел 5** | Программная реализация алгоритмов цифровой подписи | Разработка алгоритмов реализации цифровой подписи для известных криптосистем |

### 8.2. Методические указания по организации и проведению лабораторных занятий -ПРИЛОЖЕНИЕ II

### 8.3. Методические указания для обучающихся по освоению дисциплины (модулей)

Методические указания предназначены для рационального распределения времени студента по видам **самостоятельной работы** и разделам дисциплины. Они составляются на основе сведений о трудоемкости дисциплины, ее содержании и видах работы по ее изучению, а также учебно-методического и информационного обеспечения. В раздел включаются: рекомендации по изучению дисциплины (модулей) или отдельных тематических разделов, вопросы и задания для самостоятельной работы, материалы, необходимые, для подготовки к занятиям (разделы книг, статьи и т.д.). Раздел может быть представлен в табличной форме.

### 8.4.Методические рекомендации по подготовке отчетов по лабораторным работам

Требования при оформлении лабораторных работ:

1. Требования
    - Первая страница Титульный лист
    - Условия задачи, цели, этапы выполнения
    - Программный код
    - Графики
    - Результаты
    - Выводы

Правила оформления лабораторных работ:

- текст печатается на странице формата А4;
- шрифт – Times New Roman;
- размеры полей: левое – 3 см, верхнее – 2 см, правое – 2 см и нижнее – 2 см;
- выравнивание по ширине.
- размер шрифта основного текста – 12;
- интервал межстрочный (полуторный) – 1,5;
- название работы печатается полужирным, размер шрифта – 14;
- заголовки печатаются жирным шрифтом 14-ым размером, перед ними следует оставить пустую строку, выравниваются по центу;
- подзаголовки печатаются жирным шрифтом 12-ым размером выравниваются по центу;
- нумерация страниц – внизу по центру.
- Нумерация рисунков, графиков и т.п. Например: (рис.1 Название рисунка) рисунки нумеруются снизу и по центру, таблица (Таблица 1. Название таблицы) таблицы нумеруются сверху выравнивание к правому краю.
    - Библиографические ссылки при цитировании приводятся в конце статьи и нумеруются согласно порядку цитирования в тексте. Указываются автор (сначала фамилия, потом инициалы), название, место и год издания, страница. Порядковые номера ссылок должны быть написаны внутри квадратных скобок (например: [1], [2]). Источники приводятся с указанием в алфавитном порядке фамилий и инициалов всех авторов, сначала отечественных, затем иностранных, полного названия статьи, названия источника, где напечатана статья, том, номер, страницы (от и до) или полное название книги, место и год издания. Фамилии иностранных авторов,

название и выходные данные их работ даются в оригинальной транскрипции. Каждый источник приводится с новой строки.

## 9. Материально-техническое обеспечение дисциплины

Минимальные требования к материально-техническому обеспечению дисциплины:

- Компьютерный класс
- проектор, экран
- колонки
- программное обеспечение NNT Matlab.

## 10. Глоссарий

***Криптология*** - наука о создании и анализе систем безопасности связи.

***Криптография*** - это наука о методах обеспечения секретности и подлинности (идентичности) данных при их передаче по линиям связи или хранении.

***Криптоанализ*** - это наука о методах раскрытия или подделки данных.

***Открытым текстом*** называют исходные сообщение, которое должен защищать криптограф.

***Шифр*** - это множество обратимых преобразований формы открытого текста, проводимых с целью его защиты.

Процесс применения обратимого преобразования шифра к открытому тексту называется ***зашифрованием,***

Совокупность данных, определяющих конкретное преобразование из множества преобразований шифра, называют ***ключом.***

***Дешифрование*** - атака на шифр, раскрытие шифра со стороны взломщика. ***Стойкость*** - это способность противостоять попыткам хорошо вооруженного современной техникой и знаниями криптоаналитика дешифровать перехваченный шифротекст, раскрыть ключи шифра или нарушить целостность и подлинность информации.

**Приложения**

ПРИЛОЖЕНИЕ 1

- ***Лекция 1***
- ***Теоретические сведения о DES***

Американский стандарт криптографического закрытия данных DES (Data Encryption Standard), принятый в 1978 г., представляет собой блочный шифр, который шифрует данные 64-битовыми блоками. С одного конца алгоритма вводится 64-битовый блок открытого текста, а с другого конца выходит 64-битовый шифротекст. DES является симметричным алгоритмом: для шифрования и дешифрования используются одинаковые алгоритм и ключ.

Длина ключа равна 56 битам. Ключ обычно представляется 64-битовым ключом, но каждый восьмой бит используется для проверки четности и игнорируется. Биты четности являются наименьшими значащими битами байтов ключа.

Фундаментальным строительным блоком DES является применение к тексту единичной комбинации методов подстановки и перестановки, зависящей от ключа. Такой блок называется этапом. Алгоритм DES состоит из 16 этапов, одинаковая комбинация методов применяется к открытому тексту 16 раз.

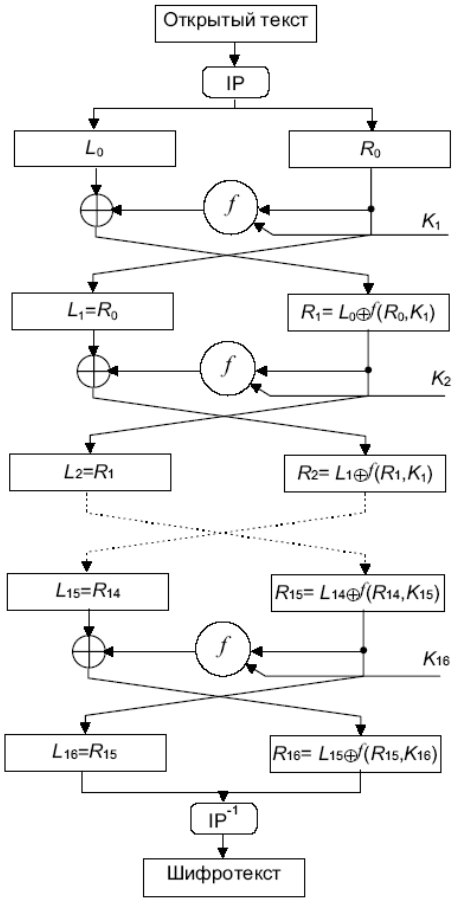



**Рисунок 1 Основная схема.**

- ***Схема алгоритма***

Процесс шифрования данных поясняется рисунком 1.

Шифр DES представляет собой результат 33 отображений:

$$DES = IP^{-1} \times \pi_{T_{16}} \times \theta \times \ldots \times \theta \times \pi_{T_1} \times IP,$$

где IP-исходная перестановка представляет собой коммутацию с инверсией $IP^{-1}$, т.е. сначала 64 бита входной последовательности перестанавливаются в соответствии с таблицей 1. Таким образом, бит 58 входной последовательности становится битом 1, бит 50 – 2 и т.д.

**Таблица 1. Начальная перестановка**

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 58 | 50 | 42 | 34 | 26 | 18 | 10 | 2 | $L_0$ |
| 60 | 52 | 44 | 36 | 28 | 20 | 12 | 4 | |
| 62 | 54 | 46 | 38 | 30 | 22 | 14 | 6 | |
| 64 | 56 | 48 | 40 | 32 | 24 | 16 | 8 | |
| 57 | 49 | 41 | 33 | 25 | 17 | 9 | 1 | $R_0$ |
| 59 | 51 | 43 | 35 | 27 | 19 | 11 | 3 | |
| 61 | 53 | 45 | 37 | 29 | 21 | 13 | 5 | |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 | 7 | |

Композиция $\theta \times \pi_{T_i}$, *где* $\theta$ -перестановка местами правой и левой половин блока данных, представляет собой одну итерацию Фейстеля (1 этап).

Подстановки $\pi_{T_i}, 1 \le i \le 16$, описываются следующим образом:

Полученная последовательность бит разделяется на две последовательности:

$L_{(0)}$ (биты 58, 50, 42, ..., 8) и $R_{(0)}$ (биты 57, 49, 41, ..., 7), каждая из которых содержит 32 бита. Затем выполняется итеративный процесс шифрования, который описывается следующими формулами:

$L_{(i)}=R_{(i-1)}$, i=1,2,...,16.

$R_{(i)}=L_{(i-1)} + F(R_{(i-1)},K_{(i)})$, i=1,2,...,16.

*Функция F* называется *функцией шифрования*. Ее аргументами являются последовательность R, полученная на предыдущем шаге, и 48-битовый ключ $K_{(i)}$, который является результатом функции преобразования 64-битового ключа шифра. Подробно функция шифрования и алгоритм получения ключей $K_{(i)}$ описаны ниже.

На последнем шаге итерации будут получены последовательности $L_{(16)}$ и $R_{(16)}$, которые конкатенируются в 64-х битовую последовательность $R_{(16)}L_{(16)}$. Видно, что в полученной последовательности 64 бита, перестанавливаются в соответствии с таблицей 2. Как легко видеть данная перестановка является обратной по отношению к начальной (см. таблицу 1).

**Таблица 2. Конечная перестановка**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 40 | 8 | 48 | 16 | 56 | 24 | 64 | 32 |
| 39 | 7 | 47 | 15 | 55 | 23 | 63 | 31 |
| 38 | 6 | 46 | 14 | 54 | 22 | 62 | 30 |
| 37 | 5 | 45 | 13 | 53 | 21 | 61 | 29 |
| 36 | 4 | 44 | 12 | 52 | 20 | 60 | 28 |
| 35 | 3 | 43 | 11 | 51 | 19 | 59 | 27 |
| 34 | 2 | 42 | 10 | 50 | 18 | 58 | 26 |
| 33 | 1 | 41 | 9 | 49 | 17 | 57 | 25 |

Полученная последовательность из 64 бит и будет являться зашифрованной последовательностью.

- ***Процесс расшифрования***

Процесс расшифрования данных является инверсным по отношению к процессу шифрования. Все действия должны быть выполнены в обратном порядке. Это означает, что расшифровываемые данные сначала переставляются в соответствии с таблицей 1, а затем над последовательностью бит $R_{(16)}L_{(16)}$ выполняется те же действия, что и в процессе зашифрования, но в обратном порядке. Итеративный процесс расшифрования описан следующими формулами:

$R_{(i-1)}=L_{(i)}$, i =16, 15, ..., 1

$L_{(i-1)}=R_{(i)}+F(L_{(i)},K_{(i)})$, i=16, 15, ..., 1.

На последнем шаге итерации будут получены последовательности $L_{(0)}$ и $R_{(0)}$, которые конкатенируются в 64 битовую последовательность $L_{(0)}R_{(0)}$. В полученной последовательности 64 бита перестанавливаются в соответствии с таблицей 2. Результат преобразования - исходная последовательность бит (расшифрованное 64-битовое значение).

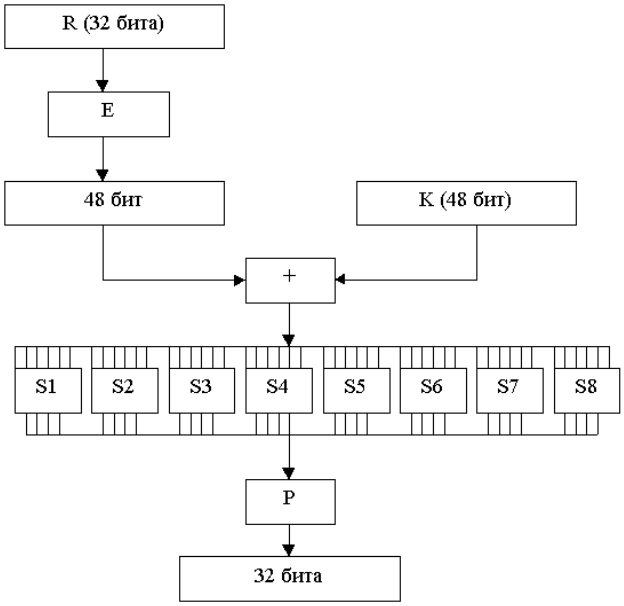



**Рисунок 2. Схема шифрования.**

- ***Функция шифрования***

Функция шифрования F(R,K) схематически показана на рисунке 2. Для вычисления значения функции F используется

- функция E (расширение 32 бит до 48),
- функции S(1), S(2),...,S(8) преобразование 6-битового числа в 4-битовое)
- функция P (перестановка бит в 32-битовой последовательности).

Приведем определения этих функций.

Аргументами функции шифрования являются R (32 бита) и K (48 бит). Результат функции E(R) есть 48-битовое число, которое складывается по модулю 2 с числом K. Таким образом, получается 48-битовая последовательность, которая рассматривается, как конкатенация 8 строк длиной по 6 бит (т.е. $B_{(1)}B_{(2)}B_{(3)}B_{(4)}B_{(5)}B_{(6)}B_{(7)}B_{(8)}$).

Результат функции $S_{(i)}B_{(i)}$ - 4 битовая последовательность, которую будем обозначать $L_{(i)}$. В результате конкатенации всех 8 полученных последовательностей L(i) имеем 32-битовую последовательность $L=L_{(1)}L_{(2)}L_{(3)}L_{(4)}L_{(5)}L_{(6)}L_{(7)}L_{(8)}$. Наконец, для получения результат функции шифрования надо переставить биты последовательности L. Для этого применяется функция перестановки $P_{(L)}$.

Функция расширения E, выполняющая расширение 32 бит до 48, определяется таблицей 3. В соответствии с этой таблицей первые три бита E(R) - это биты 32,1 и 2, а последние - 31,32,1.

**Таблица 3. Функция расширения E**

| | | | | | |
|---|---|---|---|---|---|
| 32 | 1 | 2 | 3 | 4 | 5 |
| 4 | 5 | 6 | 7 | 8 | 9 |
| 8 | 9 | 10 | 11 | 12 | 13 |
| 12 | 13 | 14 | 15 | 16 | 17 |
| 16 | 17 | 18 | 19 | 20 | 21 |
| 20 | 21 | 22 | 23 | 24 | 25 |
| 24 | 25 | 26 | 27 | 28 | 29 |
| 28 | 29 | 30 | 31 | 32 | 1 |

Функция $S_{(i)}$, которая преобразует 6-битовые числа в 4-битовые, определяется в таблицей 4.

**Таблица 4. Функции преобразования S(i)**

| | ***0*** | ***1*** | ***2*** | ***3*** | ***4*** | ***5*** | ***6*** | ***7*** | ***8*** | ***9*** | ***10*** | ***11*** | ***12*** | ***13*** | ***14*** | ***15*** | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***0*** | **14** | **4** | **13** | **1** | **2** | **15** | **11** | **8** | **3** | **10** | **6** | **12** | **5** | **9** | **0** | **7** | **$S_1$** |
| ***1*** | **0** | **15** | **7** | **4** | **14** | **2** | **13** | **1** | **10** | **6** | **12** | **11** | **9** | **5** | **3** | **8** | |
| ***2*** | **4** | **1** | **14** | **8** | **13** | **6** | **2** | **11** | **15** | **12** | **9** | **7** | **3** | **10** | **5** | **0** | |
| ***3*** | **15** | **12** | **8** | **2** | **4** | **9** | **1** | **7** | **5** | **11** | **3** | **14** | **10** | **0** | **6** | **13** | |
| | | | | | | | | | | | | | | | | | |
| ***0*** | **15** | **1** | **8** | **14** | **6** | **11** | **3** | **4** | **9** | **7** | **2** | **13** | **12** | **0** | **5** | **10** | **$S_2$** |
| ***1*** | **3** | **13** | **4** | **7** | **15** | **2** | **8** | **14** | **12** | **0** | **1** | **10** | **6** | **9** | **11** | **5** | |
| ***2*** | **0** | **14** | **7** | **11** | **10** | **4** | **13** | **1** | **5** | **8** | **12** | **6** | **9** | **3** | **2** | **15** | |
| ***3*** | **13** | **8** | **10** | **1** | **3** | **15** | **4** | **2** | **11** | **6** | **7** | **12** | **0** | **5** | **14** | **9** | |
| | | | | | | | | | | | | | | | | | |
| ***0*** | **10** | **0** | **9** | **14** | **6** | **3** | **15** | **5** | **1** | **13** | **12** | **7** | **11** | **4** | **2** | **8** | **$S_3$** |
| ***1*** | **13** | **7** | **0** | **9** | **3** | **4** | **6** | **10** | **2** | **8** | **5** | **14** | **12** | **11** | **15** | **1** | |
| ***2*** | **13** | **6** | **4** | **9** | **8** | **15** | **3** | **0** | **11** | **1** | **2** | **12** | **5** | **10** | **14** | **7** | |
| ***3*** | **1** | **10** | **13** | **0** | **6** | **9** | **8** | **7** | **4** | **15** | **14** | **3** | **11** | **5** | **2** | **12** | |
| | | | | | | | | | | | | | | | | | |
| ***0*** | **7** | **13** | **14** | **3** | **0** | **6** | **9** | **10** | **1** | **2** | **8** | **5** | **11** | **12** | **4** | **15** | **$S_4$** |
| ***1*** | **13** | **8** | **11** | **5** | **6** | **15** | **0** | **3** | **4** | **7** | **2** | **12** | **1** | **10** | **14** | **9** | |
| ***2*** | **10** | **6** | **9** | **0** | **12** | **11** | **7** | **13** | **15** | **1** | **3** | **14** | **5** | **2** | **8** | **4** | |
| ***3*** | **3** | **15** | **0** | **6** | **10** | **1** | **13** | **8** | **9** | **4** | **5** | **11** | **12** | **7** | **2** | **14** | |
| | | | | | | | | | | | | | | | | | |
| ***0*** | **2** | **12** | **4** | **1** | **7** | **10** | **11** | **6** | **8** | **5** | **3** | **15** | **13** | **0** | **14** | **9** | **$S_5$** |
| ***1*** | **14** | **11** | **2** | **12** | **4** | **7** | **13** | **1** | **5** | **0** | **15** | **10** | **3** | **9** | **8** | **6** | |
| ***2*** | **4** | **2** | **1** | **11** | **10** | **13** | **7** | **8** | **15** | **9** | **12** | **5** | **6** | **3** | **0** | **14** | |
| ***3*** | **11** | **8** | **12** | **7** | **1** | **14** | **2** | **13** | **6** | **15** | **0** | **9** | **10** | **4** | **5** | **3** | |
| | | | | | | | | | | | | | | | | | |
| ***0*** | **12** | **1** | **10** | **15** | **9** | **2** | **6** | **8** | **0** | **13** | **3** | **4** | **14** | **7** | **5** | **11** | **$S_6$** |
| ***1*** | **10** | **15** | **4** | **2** | **7** | **12** | **9** | **5** | **6** | **1** | **13** | **14** | **0** | **11** | **3** | **8** | |
| ***2*** | **9** | **14** | **15** | **5** | **2** | **8** | **12** | **3** | **7** | **0** | **4** | **10** | **1** | **13** | **11** | **6** | |
| ***3*** | **4** | **3** | **2** | **12** | **9** | **5** | **15** | **10** | **11** | **14** | **1** | **7** | **6** | **0** | **8** | **13** | |
| | | | | | | | | | | | | | | | | | |
| ***0*** | **4** | **11** | **2** | **14** | **15** | **0** | **8** | **13** | **3** | **12** | **9** | **7** | **5** | **10** | **6** | **1** | **$S_7$** |
| ***1*** | **13** | **0** | **11** | **7** | **4** | **9** | **1** | **10** | **14** | **3** | **5** | **12** | **2** | **15** | **8** | **6** | |
| ***2*** | **1** | **4** | **11** | **13** | **12** | **3** | **7** | **14** | **10** | **15** | **6** | **8** | **0** | **5** | **9** | **2** | |

| | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ***3*** | **6** | **11** | **13** | **8** | **1** | **4** | **10** | **7** | **9** | **5** | **0** | **15** | **14** | **2** | **3** | **12** | |
| | | | | | | | | | | | | | | | | | |
| ***0*** | **13** | **2** | **8** | **4** | **6** | **15** | **11** | **1** | **10** | **9** | **3** | **14** | **5** | **0** | **12** | **7** | **$S_8$** |
| ***1*** | **1** | **15** | **13** | **8** | **10** | **3** | **7** | **4** | **12** | **5** | **6** | **11** | **0** | **14** | **9** | **2** | |
| ***2*** | **7** | **11** | **4** | **1** | **9** | **12** | **14** | **2** | **0** | **6** | **10** | **13** | **15** | **3** | **5** | **8** | |
| ***3*** | **2** | **1** | **14** | **7** | **4** | **10** | **8** | **13** | **15** | **12** | **9** | **0** | **3** | **5** | **6** | **11** | |

К таблице 4 требуются дополнительные пояснения. Каждая из функций $S_{(i)}B_{(i)}$ преобразовывает 6-битовый код в 4-битовый выход по следующему алгоритму:

- первый и последний биты входной последовательности B, определяют номер строки k.
- второй, третий, четвертый и пятый биты последовательности B задают номер колонки l
- результат преобразования выбирается из строки k и колонки l.

Предположим, что B=011011. Тогда $S_{(1)}(B)=0101$. Действительно, k=1, l=13. В колонке 13 строки 1 задано значение 5, которое и является значением функции $S_{(1)}(011011)$.

Функция перестановки бит P(L), также используемая для определения функции шифрования, задается значениями, приведенными в таблице 5. В последовательности $L_{(32)}$ перестанавливается так, чтобы бит 16 стал первым битом, бит 7 - вторым и т.д.

**Таблица 5. Функция перестановки P**

| | | | |
|---|---|---|---|
| 16 | 7 | 20 | 21 |
| 29 | 12 | 28 | 17 |
| 1 | 15 | 23 | 26 |
| 5 | 18 | 31 | 10 |
| 2 | 8 | 24 | 14 |
| 32 | 27 | 3 | 9 |
| 19 | 13 | 30 | 6 |
| 22 | 11 | 4 | 25 |

Чтобы завершить описание алгоритма шифрования данных, осталось привести алгоритм получение ключей K(i), i=1,2,...,16, размерностью в 48 бит. Ключи K(i) определяются по 64-битовому ключу шифра как это показано на рисунке 3.

В начале над ключом шифра выполняется операция B, которая сводится к выбору определенных бит и их перестановке, как это показано в таблицей 6.

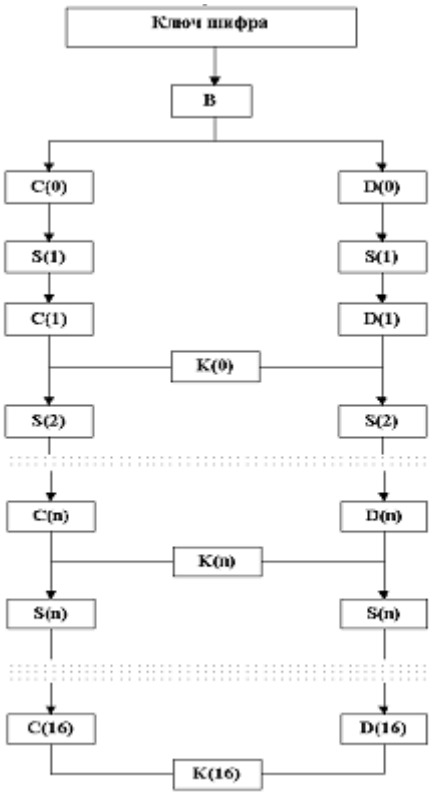



**Рисунок 2. Генерация ключей.**

**Таблица 6. Функция перестановки и выбора последовательности B**

| | | | | | | |
|---|---|---|---|---|---|---|
| 57 | 49 | 41 | 33 | 25 | 17 | 9 |
| 1 | 58 | 50 | 42 | 34 | 26 | 18 |
| 10 | 2 | 59 | 51 | 43 | 35 | 27 |
| 19 | 11 | 3 | 60 | 52 | 44 | 36 |
| 63 | 55 | 47 | 39 | 31 | 23 | 15 |
| 7 | 62 | 54 | 46 | 38 | 30 | 22 |
| 14 | 6 | 61 | 53 | 45 | 37 | 29 |
| 21 | 13 | 5 | 28 | 20 | 12 | 4 |

Причем, первые четыре строки определяют, как выбираются биты последовательности $C_{(0)}$, (первым битом $C_{(0)}$ будет 57 бит ключа шифра, затем 49 бит и т.д., а последними битами - 44 и 36 ключа шифра), а следующие четыре строки - как выбираются биты последовательности $D_{(0)}$ (т.е. последовательность $D_{(0)}$ будем стоять из битов 63,55,...,12, 4 ключа шифра).

Как видно из таблицы 5, для генерации последовательностей $C_{(0)}$ и $D_{(0)}$ не используются биты 8,16,25,32,40,48,56 и 64 ключа шифра. Эти биты не влияют на шифрование и могут служить для других целей (например, для контроля по четности). Таким образом, в действительности ключ шифра является 56-битовым. После определения $C_{(0)}$ и $D_{(0)}$ рекурсивно определяются $C_{(i)}$ и $D_{(i)}$, i=1,2,...,16. Для этого применяются операции сдвига влево на один или два бита в зависимости от номера шага итерации, как это показано в таблицей 6. Операции сдвига выполняются для последовательностей $C_{(i)}$ и $D_{(i)}$ независимо. Например, последовательность $C_{(3)}$ получается, посредством сдвига влево на две позиции последовательности $C_{(2)}$, а последовательность $D_{(3)}$ - посредством сдвига влево на две позиции последовательности D(2). Следует иметь в виду, что выполняется циклический сдвиг влево. Например,

единичный сдвиг влево последовательности $C_{(i)}$ приведет к тому, что первый бит $C_{(i)}$ станет последним, и последовательность бит будет следующая: 2,3,..., 28,1.

**Таблица 7. Число битов сдвига ключа в зависимости от этапа.**

| Этап | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Число | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 2 | 2 | 2 | 2 | 1 | 2 | 1 |

Ключ $K_{(i)}$, определяемый на каждом шаге итерации, есть результат выбора определенных бит из 56-битовой последовательности $C_{(i)}D_{(i)}$ и их перестановки. Другими словами, $K_{(i)} = K(C_{(i)}D_{(i)})$, где функция K определяется данными, приведенными в таблицей 8.

**Таблица 8. Функция перестановки и выбора K**

| | | | | | |
|---|---|---|---|---|---|
| 14 | 17 | 11 | 24 | 1 | 5 |
| 3 | 28 | 15 | 6 | 21 | 10 |
| 23 | 19 | 12 | 4 | 26 | 8 |
| 16 | 7 | 27 | 20 | 13 | 2 |
| 41 | 52 | 31 | 37 | 47 | 55 |
| 30 | 40 | 51 | 45 | 33 | 48 |
| 44 | 49 | 39 | 56 | 34 | 53 |
| 46 | 42 | 50 | 36 | 29 | 32 |

Как следует из таблицы 8 первый бит $K_{(i)}$ - это бит 14 последовательности $C_{(i)}D_{(i)}$, второй - бит 17, последний - бит 32.

При расшифрование–раундовые ключи теже, что и при зашифровании, но они используются в обратном порядке

КРИПТОАНАЛИЗ DES

С самого начала использования алгоритма криптоаналитики всего мира прилагали множество усилий для взлома DES. Фактически DES дал невиданный доселе толчок развитию криптоанализа. Вышли сотни трудов, посвященных различным методам криптоанализа именно в приложении к алгоритму DES, а также деталям самого алгоритма и их влиянию на криптостойкость. Можно утверждать, что именно благодаря DES появились целые направления криптоанализа, такие как:

-линейный криптоанализ–анализ зависимостей между открытым текстом и шифр о текстом;

-дифференциальный криптоанализ–анализ зависимостей между соотношениями двух или более открытых текстов и соответствующих им шифротекстов;

-криптоанализ на связанных ключах –поиск и анализ зависимостей между шифротекстами, полученными на искомом ключе и ключах, связанных предполагаемым соотношением с искомым ключом;

однако DES оказался неуязвим к данному виду атак, так как по ключовому расписанию циклический сдвиг битов ключа выполняется на различное число позиций в разных раундах.

DES стойко выдержал 20 лет массового всемирного криптоанализа –десятилетия криптоанализа не привели к обнаружению серьезных уязвимостей в алгоритме. Основными результатами усилий по взлому DES можно считать следующие:

1. Японский специалист Мицуру Мацуи (Mitsuru Matsui), изобретатель линейного криптоанализа, в 1993 году показал, что вычислить ключ шифр а можно методом линейного криптоанализа при наличии у атакующего $2^{47}$ пар «открытый текст–шифротекст».

2. Криптологи из Израиля, изобретатели дифференциального криптоанализа Эли Бихам (Eli Biham) и Ади Шамир (Adi Shamir) в 1991 году представили атаку, в которой ключ шифрования вычислялся методом дифференциального криптоанализа при условии, что атакующий имеет $2^{47}$ специально выбранных пар «открытый текст – шифротекст».

В дальнейшем эти атаки были несколько усилены (например, атака линейным криптоанализом при наличии $2^{43}$ пар вместо $2^{47}$), появлялись также новые виды атак на DES (например, атака, позволяющая вычислить ключ высокоточным облучением аппаратного шифратора и последующим анализом ошибок шифрования).

Однако, все эти атаки требуют наличия огромного количества пар «открытый текст –шифротекст», получение которых на практике является настолько трудоемкой операцией, что наиболее простой атакой на DES все еще можно считать полный перебор ключей.

Лекция 2-3

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

### 1. Описание алгоритма.

#### 1. Термины и обозначения.

Описание стандарта шифрования Российской Федерации содержится в очень интересном документе, озаглавленном «Алгоритм криптографического преобразования данных ГОСТ 28147-89». То, что в его названии вместо термина «шифрование» фигурирует более общее понятие «***криптографическое преобразование***», вовсе не случайно. Помимо нескольких тесно связанных между собой процедур шифрования, в документе описан дополнительный алгоритм выработки ***имитовставки,*** которая является криптографической контрольной комбинацией, то есть кодом, вырабатываемым из исходных данных с использованием секретного ключа с целью ***имитозащиты***, или защиты данных от внесения в них несанкционированных изменений.

Используемые обозначения:

- Элементы данных в данной статье обозначаются заглавными латинскими буквами с наклонным начертанием (например, $\boldsymbol{X}$).
- Через $|\boldsymbol{X}|$ обозначается размер элемента данных $\boldsymbol{X}$ в битах. Таким образом, если интерпретировать элемент данных $\boldsymbol{X}$ как целое неотрицательное число, можно записать следующее неравенство: $0 \le \boldsymbol{X} < 2^{|\boldsymbol{X}|}$.
- Если элемент данных состоит из нескольких элементов меньшего размера, то этот факт обозначается следующим образом:

$$X = (X_0, X_1, \ ..., X_{n-1}) = X_0 \,||\, X_1 \,||\ldots||\, X_{n-1}.$$

Процедура объединения нескольких элементов данных в один называется ***конкатенацией*** данных и обозначается символом

- В алгоритме элемент данных может интерпретироваться как массив отдельных битов, в этом случае биты обозначаем той же самой буквой, что и массив, но в строчном варианте: $X = (x_0, x_1, ..., x_{n-1}) = x_0 + 2^1 \cdot x_1 + ... + 2^{n-1} \cdot x_{n-1}$.
- Если над элементами данных выполняется некоторая операция, имеющая логический смысл, то предполагается, что данная операция выполняется над соответствующими битами элементов. Иными словами

$A \bullet B = (a_0 \bullet b_0, a_1 \bullet b_1, ..., a_{n-1} \bullet b_{n-1})$,

где $n=|A|=|B|$, а символом "•" обозначается произвольная бинарная логическая операция; как правило, имеется ввиду операция ***исключающего или***, она же – операция суммирования по модулю 2: $a \oplus b = (a+b) \bmod 2$.

## 2. Структура ключевой информации ГОСТа.

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

1. ***Ключ*** является массивом из восьми 32-битных элементов кода, далее в настоящей работе он обозначается символом $K$: $K = \{K_i\}_{0 \le i \le 7}$. В ГОСТе элементы ключа используются как 32-разрядные целые числа без знака: $0 \le K_i < 2^{32}$. Таким образом, размер ключа составляет 32·8=256 бит или 32 байта.

2. ***Таблица замен*** является матрицей 8×16, содержащей 4-битовые элементы, которые можно представить в виде целых чисел от 0 до 15. Строки ***таблицы замен*** называются ***узлами замен***, они должны содержать различные значения, то есть каждый ***узел замен*** должен содержать 16 различных чисел от 0 до 15 в произвольном порядке. В настоящей статье таблица замен обозначается символом $H$: $H = \{H_{i,j}\}_{0 \le j \le 15}^{0 \le i \le 7}$, $0 \le H_{i,j} \le 15$. Таким образом, общий объем таблицы замен равен: 8 узлов × 16 элементов/узел × 4 бита/элемент = 512 бит или 64 байта.

## 3. Основной шаг криптопреобразования.

Основной шаг криптопреобразования по своей сути является оператором, определяющим преобразование 64-битового блока данных. Дополнительным параметром этого оператора является 32-битовый блок, в качестве которого используется элемент ключа. Схема алгоритма основного шага приведена на рисунке 1.

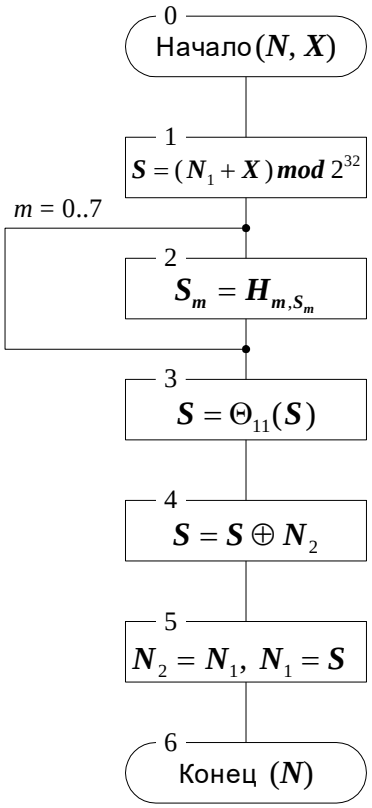



**Рисунок 3. Схема основного шага криптопреобразования алгоритма ГОСТ 28147-89.**

Ниже даны пояснения к алгоритму основного шага:

0. Определяет исходные данные для основного шага криптопреобразования:
   0.1. $\boldsymbol{N}$ – преобразуемый 64-битовый блок данных, в ходе выполнения шага его младшая ($\boldsymbol{N}_1$) и старшая ($\boldsymbol{N}_2$) части обрабатываются как отдельные 32-битовые целые числа без знака. Таким образом, можно записать $\boldsymbol{N}=(\boldsymbol{N}_1,\boldsymbol{N}_2)$.
   0.2. $\boldsymbol{X}$ – 32-битовый элемент ключа;
1. Сложение с ключом. Младшая половина преобразуемого блока складывается по модулю $2^{32}$ с используемым на шаге элементом ключа, результат передается на следующий шаг;
2. Поблочная замена. 32-битовое значение, полученное на предыдущем шаге, интерпретируется как массив из восьми 4-битовых блоков кода: $\boldsymbol{S}=(\boldsymbol{S}_0,\boldsymbol{S}_1,\boldsymbol{S}_2,\boldsymbol{S}_3,\boldsymbol{S}_4,\boldsymbol{S}_5,\boldsymbol{S}_6,\boldsymbol{S}_7)$.
   Далее значение каждого из восьми блоков заменяется на новое, которое выбирается по таблице замен следующим образом: значение блока $\boldsymbol{S_i}$ заменяется на $\boldsymbol{S_i}$-тый по порядку элемент (нумерация с нуля) $\boldsymbol{i}$-того узла замен (т.е. $\boldsymbol{i}$-той строки таблицы замен, нумерация также с нуля). Другими словами, в качестве замены для значения блока выбирается элемент из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа. Теперь становится понятным размер таблицы замен: число строк в ней равно числу 4-битных элементов в 32-битном блоке данных, то есть восьми, а число столбцов равно числу различных значений 4-битного блока данных, равному как известно $2^4$, шестнадцати.

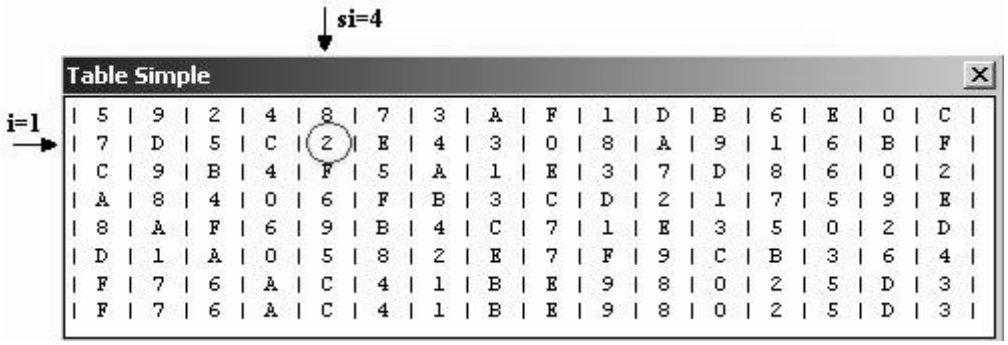



**Рисунок 4 Пример таблицы замен.**

3. Циклический сдвиг на 11 бит влево. Результат предыдущего шага сдвигается циклически на 11 бит в сторону старших разрядов и передается на следующий шаг. На схеме алгоритма символом $\Theta_{11}$ обозначена функция циклического сдвига своего аргумента на 11 бит в сторону старших разрядов.
4. Побитовое сложение: значение, полученное на шаге 3, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.
5. Сдвиг по цепочке: младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего шага.
6. Полученное значение преобразуемого блока возвращается как результат выполнения алгоритма основного шага криптопреобразования.

## 4. Базовые циклы криптографических преобразований.

ГОСТ относится к классу блочных шифров, то есть единицей обработки информации в нем является блок данных. Следовательно, в нем определены алгоритмы для криптографических преобразований, то есть для зашифрования, расшифрования и «учета» в контрольной комбинации одного блока данных. Именно эти алгоритмы и называются ***базовыми циклами*** ГОСТа, что подчеркивает их фундаментальное значение для построения этого шифра. Каждый из циклов имеет собственное буквенно-цифровое обозначение, соответствующее шаблону «*n-X*», где первый элемент обозначения (*n*),

задает число повторений основного шага в цикле, а второй элемент обозначения (*X*), буква, задает порядок зашифрования («З») или расшифрования («Р») в использовании ключевых элементов.

- **цикл зашифрования (32-З);**
- **цикл расшифрования (32-Р);**
- **цикл выработки имитовставки (16-З).**

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

Все сделанные выше предположения, опирающиеся просто на здравый смысл, оказались верными. Базовые циклы заключаются в многократном выполнении *основного шага* с использованием разных элементов ключа и отличаются друг от друга только числом повторения шага и порядком использования ключевых элементов. Ниже приведен этот порядок для различных циклов.

1. Цикл зашифрования 32-З:

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

2. Цикл расшифрования 32-Р:

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

3. Цикл выработки имитовставки 16-З:

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

### 4.1. Основные режимы шифрования.

ГОСТ 28147-89 предусматривает три следующих режима шифрования данных:

- простая замена,
- гаммирование,
- гаммирование с обратной связью,

и один дополнительный режим выработки имитовставки.

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

Прежде чем перейти к рассмотрению конкретных алгоритмов криптографических преобразований, необходимо пояснить обозначения, используемые на схемах в следующих разделах:

$T_{\text{О}},T_{\text{Ш}}$ – массивы соответственно открытых и зашифрованных данных;

$T_i^{\circ}, T_i^{\text{Ш}}$ – *i*-тые по порядку 64-битные блоки соответственно открытых и зашифрованных данных: $T_{\text{О}} = (T_1^{\circ},T_2^{\circ},...,T_n^{\circ})$, $T_{\text{Ш}} = (T_1^{\text{Ш}},T_2^{\text{Ш}},...,T_n^{\text{Ш}})$, $1 \le i \le n$, последний блок может быть неполным: $|T_i^{\circ}|=|T_i^{\text{Ш}}|= 64$ при $1 \le i < n$, $1 \le |T_n^{\circ}|=|T_n^{\text{Ш}}| \le 64$;

***n*** – число 64-битных блоков в массиве данных;

***Ц*x** – функция преобразования 64-битного блока данных по алгоритму базового цикла «X»;

Теперь опишем основные режимы шифрования:

### 4.2. Простая замена.

Зашифрование в данном режиме заключается в применении цикла 32-З к блокам открытых данных, расшифрование – цикла 32-Р к блокам зашифрованных данных. Это наиболее простой из режимов, 64-битовые блоки данных обрабатываются в нем независимо друг от друга. Схемы алгоритмов зашифрования и расшифрования в режиме простой замены приведены на рисунках 3 и 4 соответственно.

Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам: $|\boldsymbol{T}_{\text{о}}|=|\boldsymbol{T}_{\text{ш}}|=64\cdot\boldsymbol{n}$, после выполнения операции размер полученного массива данных не изменяется.

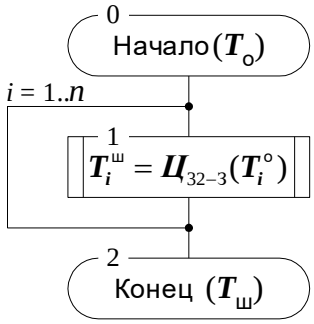



**Рисунок 5 Алгоритм зашифрования данных в режиме простой замены.**

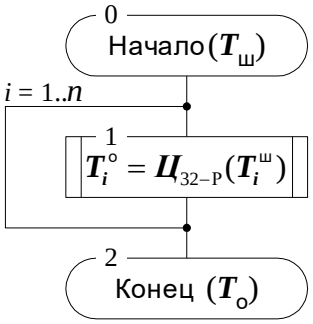



**Рисунок 6 Алгоритм расшифрования данных в режиме простой замены.**

Режим шифрования простой заменой имеет следующие особенности:

1. Так как блоки данных шифруются независимо друг от друга и от их позиции в массиве, при зашифровании двух одинаковых блоков открытого текста получаются одинаковые блоки шифротекста и наоборот. Отмеченное свойство позволит криптоаналитику сделать заключение о тождественности блоков исходных данных, если в массиве зашифрованных данных ему встретились идентичные блоки, что является недопустимым для серьезного шифра.
2. Если длина шифруемого массива данных не кратна 8 байтам или 64 битам, возникает проблема, чем и как дополнять последний неполный блок данных массива до полных 64 бит. Очевидные решения типа «дополнить неполный блок нулевыми битами» или «дополнить неполный блок фиксированной комбинацией нулевых и единичных битов» могут при определенных условиях дать в руки криптоаналитика возможность методами перебора определить содержимое этого самого неполного блока, и этот факт означает снижение стойкости шифра. Кроме того, длина шифротекста при этом изменится, увеличившись до ближайшего целого, кратного 64 битам, что часто бывает нежелательным.

Кажется, что для любых реальных данных гарантировать выполнение указанных условий невозможно. Это почти так, но есть одно очень важное исключение: размер ключа составляет 32 байта, а размер таблицы замен – 64 байта. Кроме того, наличие повторяющихся 8-байтовых блоков в ключе или таблице замен будет говорить об их весьма плохом качестве, поэтому в реальных ключевых элементах такого повторения быть не может. Таким образом, режим простой замены подходит для шифрования ключевой информации, тем более, что прочие режимы для этой цели менее удобны, поскольку требуют наличия дополнительного синхронизирующего элемента данных –

синхропосылки (см. следующий раздел). Вывод: ГОСТ предписывает использовать режим простой замены исключительно для шифрования ключевых данных.

### 4.3 . Гаммирование.

Как же можно избавиться от недостатков режима простой замены? Для этого необходимо сделать возможным шифрование блоков с размером менее 64 бит и обеспечить зависимость блока шифротекста от его номера, иными словами, ***рандомизировать*** процесс шифрования. В ГОСТе это достигается двумя различными способами в двух режимах шифрования, предусматривающих ***гаммирование***.

***Гаммирование*** – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, то есть последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. В ГОСТе для этой цели используется операция побитного сложения по модулю 2, поскольку она является обратной самой себе и к тому же наиболее просто реализуется. Гаммирование решает обе упомянутые проблемы;

во первых, все элементы гаммы различны для реальных шифруемых массивов и, следовательно, результат зашифрования даже двух одинаковых блоков в одном массиве данных будет различным;

во вторых, хотя элементы гаммы и вырабатываются одинаковыми порциями в 64 бита, использоваться может и часть такого блока с размером, равным размеру шифруемого блока.

Гамма для этого режима получается следующим образом: с помощью некоторого алгоритмического рекуррентного генератора последовательности чисел (РГПЧ) вырабатываются 64-битные блоки данных, которые далее подвергаются преобразованию по циклу 32-З, то есть зашифрованию в режиме простой замены, в результате получаются блоки гаммы. Благодаря тому, что наложение и снятие гаммы осуществляется при помощи одной и той же операции побитового исключающего или, алгоритмы зашифрования и расшифрования в режиме гаммирования идентичны, их общая схема приведена на рисунке 5.

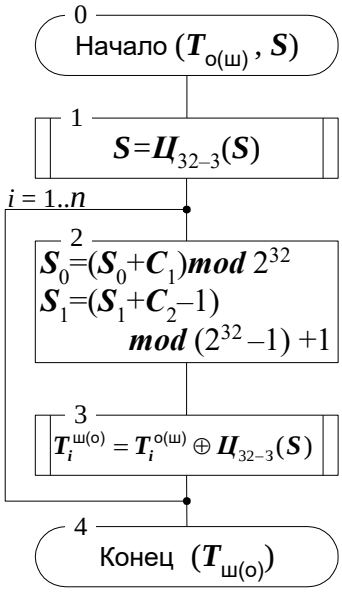



**Рисунок 7. Алгоритм зашифрования (расшифрования) данных в режиме гаммирования.**

РГПЧ, используемый для выработки гаммы, является рекуррентной функцией: $\Omega_{i+1}=f(\Omega_i)$, где $\Omega_i$ – элементы рекуррентной последовательности, $f$ – функция преобразования. Следовательно, неизбежно возникает вопрос о его инициализации, то есть об элементе $\Omega_0$. В действительности, этот элемент данных является параметром алгоритма для режимов гаммирования, на схемах он обозначен как $S$, и называется в

криптографии ***синхропосылкой***, а в нашем ГОСТе – ***начальным заполнением*** одного из регистров шифрователя. По определенным соображениям разработчики ГОСТа решили использовать для инициализации РГПЧ не непосредственно синхропосылку, а результат ее преобразования по циклу 32-З: $\Omega_0=\boldsymbol{Ц}_{32\text{-}З}(\boldsymbol{S})$. Последовательность элементов, вырабатываемых РГПЧ, целиком зависит от его начального заполнения, то есть элементы этой последовательности являются функцией своего номера и начального заполнения РГПЧ: $\Omega_i=f_i(\Omega_0)$, где $f_i(X)=f(f_{i-1}(X))$, $f_0(X)=X$. С учетом преобразования по алгоритму простой замены добавляется еще и зависимость от ключа: $\Gamma_i=\boldsymbol{Ц}_{32\text{-}З}(\Omega_i)=\boldsymbol{Ц}_{32\text{-}З}(f_i(\Omega_0))=\boldsymbol{Ц}_{32\text{-}З}(f_i(\boldsymbol{Ц}_{32\text{-}З}(\boldsymbol{S})))=\varphi_i(\boldsymbol{S},\boldsymbol{K})$, где $\Gamma_i$ – ***i***-тый элемент гаммы, $\boldsymbol{K}$ – ключ.

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

Теперь подробно рассмотрим РГПЧ, используемый в ГОСТе для генерации элементов гаммы. Прежде всего надо отметить, что к нему не предъявляются требования обеспечения каких-либо статистических характеристик вырабатываемой последовательности чисел. РГПЧ спроектирован разработчиками ГОСТа исходя из необходимости выполнения следующих условий:

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

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

- в 64-битовом блоке старшая и младшая части обрабатываются независимо друг от друга: $\Omega_i=(\Omega_i^0,\Omega_i^1)$, $|\Omega_i^0|=|\Omega_i^1|=32$, $\Omega_{i+1}^0=\hat{f}(\Omega_i^0)$, $\Omega_{i+1}^1=\tilde{f}(\Omega_i^1)$, фактически, существуют два независимых РГПЧ для старшей и младшей частей блока.
- рекуррентные соотношения для старшей и младшей частей следующие:
  $\Omega_{i+1}^0=(\Omega_i^0+\boldsymbol{C}_1)\,\boldsymbol{mod}\,2^{32}$, где $\boldsymbol{C}_1=1010101_{16}$;
  $\Omega_{i+1}^1=(\Omega_i^1+\boldsymbol{C}_2-1)\,\boldsymbol{mod}\,(2^{32}-1)+1$, где $\boldsymbol{C}_2=1010104_{16}$;
  Нижний индекс в записи числа означает его систему счисления, таким образом, константы, используемые на данном шаге, записаны в 16-ричной системе счисления. Второе выражение нуждается в комментариях, так как в тексте ГОСТа приведено нечто другое: $\Omega_{i+1}^1=(\Omega_i^1+\boldsymbol{C}_2)\,\boldsymbol{mod}\,(2^{32}-1)$, с тем же значением константы $\boldsymbol{C}_2$. Но далее в тексте стандарта дается комментарий, что, оказывается, под операцией взятия остатка по модулю $2^{32}$–1 *там* понимается не то же самое, что и в математике. Отличие заключается в том, что согласно ГОСТу $(2^{32}-1)\boldsymbol{mod}(2^{32}-1)=(2^{32}-1)$, а не 0. На самом

деле, это упрощает реализацию формулы, а математически корректное выражение для нее приведено выше.

- период повторения последовательности для младшей части составляет $2^{32}$, для старшей части $2^{32}–1$, для всей последовательности период составляет $2^{32}\cdot(2^{32}–1)$. Первая формула из двух реализуется за одну команду, вторая, несмотря на ее кажущуюся громоздкость, за две команды на всех современных 32-разрядных процессорах.

Схема алгоритма шифрования в режиме гаммирования приведена на рисунке 5, ниже изложены пояснения к схеме:

**1.** Определяет исходные данные для основного шага криптопреобразования:

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

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

**3.** Один шаг работы РГПЧ, реализующий его рекуррентный алгоритм. В ходе данного шага старшая ($\boldsymbol{S}_1$) и младшая ($\boldsymbol{S}_0$) части последовательности данных вырабатываются независимо друг от друга;

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

**5.** Результат работы алгоритма – зашифрованный (расшифрованный) массив данных.

Ниже перечислены особенности гаммирования как режима шифрования.

1. Одинаковые блоки в открытом массиве данных дадут при зашифровании различные блоки шифротекста, что позволит скрыть факт их идентичности.
2. Поскольку наложение гаммы выполняется побитно, шифрование неполного блока данных легко выполнимо как шифрование битов этого неполного блока, для чего используется соответствующие биты блока гаммы . Так, для зашифрования неполного блока в 1 бит можно использовать любой бит из блока гаммы.
3. Синхропосылка, использованная при зашифровании, каким-то образом должна быть передана для использования при расшифровании. Это может быть достигнуто следующими путями:
   - хранить или передавать синхропосылку вместе с зашифрованным массивом данных, что приведет к увеличению размера массива данных при зашифровании на размер синхропосылки, то есть на 8 байт;
   - использовать предопределенное значение синхропосылки или вырабатывать ее синхронно источником и приемником по определенному закону, в этом случае изменение размера передаваемого или хранимого массива данных отсутствует;

   Оба способа дополняют друг друга, и в тех редких случаях, где не работает первый, наиболее употребительный из них, может быть использован второй, более экзотический. Второй способ имеет гораздо меньшее применение, поскольку сделать синхропосылку предопределенной можно только в том случае, если на данном комплекте ключевой информации шифруется заведомо не более одного массива данных, что бывает в редких случаях. Генерировать синхропосылку синхронно у источника и получателя массива данных также не всегда представляется возможным, поскольку требует жесткой привязки к чему-либо в системе. Так, здравая на первый взгляд идея использовать в качестве синхропосылки в системе передачи зашифрованных сообщений номер передаваемого сообщения не подходит, поскольку сообщение может потеряться и не

дойти до адресата, в этом случае произойдет десинхронизация систем шифрования источника и приемника. Поэтому в рассмотренном случае нет альтернативы передаче синхропосылки вместе с зашифрованным сообщением.

С другой стороны, можно привести и обратный пример. Допустим, шифрование данных используется для защиты информации на диске, и реализовано оно на низком уровне, для обеспечения независимого доступа данные шифруются по секторам. В этом случае невозможно хранить синхропосылку вместе с зашифрованными данными, поскольку размер сектора нельзя изменить, однако ее можно вычислять как некоторую функцию от номера считывающей головки диска, номера дорожки (цилиндра) и номера сектора на дорожке. В этом случае синхропосылка привязывается к положению сектора на диске, которое вряд ли может измениться без переформатирования диска, то есть без уничтожения данных на нем.

Режим гаммирования имеет еще одну интересную особенность. В этом режиме биты массива данных шифруются независимо друг от друга. Таким образом, каждый бит шифротекста зависит от соответствующего бита открытого текста и, естественно, порядкового номера бита в массиве: $t_i^{\text{ш}} = t_i^{\text{о}} \oplus \gamma_i = f(t_i^{\text{о}}, i)$. Из этого вытекает, что изменение бита шифротекста на противоположное значение приведет к аналогичному изменению бита открытого текста на противоположный:

$$\bar{t}_i^{\text{ш}} = t_i^{\text{ш}} \oplus 1 = (t_i^{\text{о}} \oplus \gamma_i) \oplus 1 = (t_i^{\text{о}} \oplus 1) \oplus \gamma_i = \bar{t}_i^{\text{о}} \oplus \gamma_i,$$

где $\bar{t}$ обозначает инвертированное по отношению к *t* значение бита ($\bar{0} = 1$, $\bar{1} = 0$).

Данное свойство дает злоумышленнику возможность воздействуя на биты шифротекста вносить предсказуемые и даже целенаправленные изменения в соответствующий открытый текст, получаемый после его расшифрования, не обладая при этом секретным ключом. Это иллюстрирует хорошо известный в криптологии факт, что ***«секретность и аутентичность суть различные свойства шифров»***. Иными словами, свойства шифров обеспечивать защиту от несанкционированного ознакомления с содержимым сообщения и от несанкционированного внесения изменений в сообщение являются независимыми и лишь в отдельных случаях могут пересекаться. Сказанное означает, что существуют криптографические алгоритмы, обеспечивающие определенную секретность зашифрованных данных и при этом никак не защищающие от внесения изменений и наоборот, обеспечивающие аутентичность данных и никак не ограничивающие возможность ознакомления с ними. По этой причине рассматриваемое свойство режима гаммирования не должно рассматриваться как его недостаток.

### 4.4 Гаммирование с обратной связью.

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

Схема алгоритмов за- и расшифрования в режиме гаммирования с обратной связью приведена на рисунке 6 и ввиду своей простоты в комментариях не нуждается.

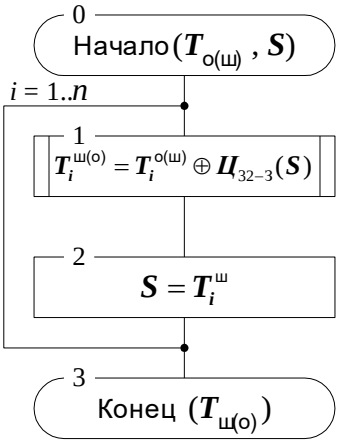



**Рисунок 8. Алгоритм зашифрования (расшифрования) данных в режиме гаммирования с обратной связью.**

Шифрование в режиме гаммирования с обратной связью обладает теми же особенностями, что и шифрование в режиме обычного гаммирования, за исключением влияния искажений шифротекста на соответствующий открытый текст. Для сравнения запишем функции расшифрования блока для обоих упомянутых режимов:

$T_i^{\circ} = T_i^{ш} \oplus \Gamma_i$, гаммирование;

$T_i^{\circ} = T_i^{ш} \oplus U_{32-3}(T_{i-1}^{ш})$, гаммирование с обратной связью;

Если в режиме обычного гаммирования изменения в определенных битах шифротекста влияют только на соответствующие биты открытого текста, то в режиме гаммирования с обратной связью картина несколько сложнее. Как видно из соответствующего уравнения, при расшифровании блока данных в режиме гаммирования с обратной связью, блок открытых данных зависит от соответствующего и предыдущего блоков зашифрованных данных. Поэтому, если внести искажения в зашифрованный блок, то после расшифрования искаженными окажутся два блока открытых данных – соответствующий и следующий за ним, причем искажения в первом случае будут носить тот же характер, что и в режиме гаммирования, а во втором случае – как в режиме простой замены. Другими словами, в соответствующем блоке открытых данных искаженными окажутся те же самые биты, что и в блоке шифрованных данных, а в следующем блоке открытых данных все биты независимо друг от друга с вероятностью 1/2 изменят свои значения.

## 5 Выработка имитовставки к массиву данных.

В предыдущих разделах мы обсудили влияние искажения шифрованных данных на соответствующие открытые данные. Мы установили, что при расшифровании в режиме простой замены соответствующий блок открытых данных оказывается искаженным непредсказуемым образом, а при расшифровании блока в режиме гаммирования изменения предсказуемы. В режиме гаммирования с обратной связью искаженными оказываются два блока, один предсказуемым, а другой непредсказуемым образом. Значит ли это, что с точки зрения защиты от навязывания ложных данных режим гаммирования является плохим, а режимы простой замены и гаммирования с обратной связью хорошими? Ни в коем случае. При анализе данной ситуации необходимо учесть то, что непредсказуемые изменения в расшифрованном блоке данных могут быть обнаружены только в случае избыточности этих самых данных, причем чем больше степень избыточности, тем вероятнее обнаружение искажения. Очень большая избыточность имеет место, например, для текстов на естественных и искусственных языках, в этом случае факт искажения обнаруживается практически неизбежно. Однако в других случаях, например, при искажении сжатых звуковых образов, мы получим просто другой образ, который сможет воспринять наше ухо. Искажение в этом случае останется необнаруженным, если, конечно, нет априорной информации о характере звука. Вывод

здесь такой: поскольку способность некоторых режимов шифрования обнаруживать искажения, внесенные в шифрованные данные, существенным образом опирается на наличие и степень избыточности шифруемых данных, эта способность не является имманентным свойством соответствующих режимов и не может рассматриваться как их достоинство.

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

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

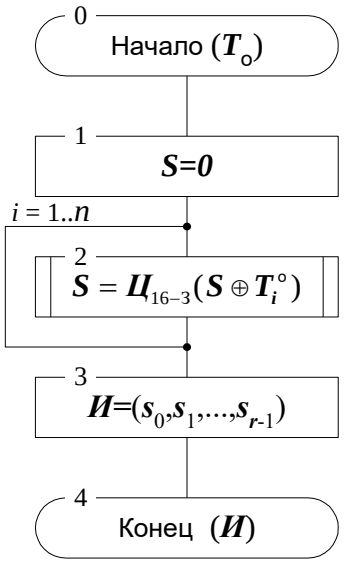



**Рисунок 9. Алгоритм выработки имитовставки для массива данных.**

Схема алгоритма выработки имитовставки приведена на рисунке 7. 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 $2^{-32} \approx 0.23 \cdot 10^{-9}$.

## Лекция 4-5

Алгоритм DES (Data Encryption Standard), разработанный корпорацией IBM и являвшийся с 1977 федеральным стандартом шифрования данных США, использовался для шифрования не только правительством США, но и получил широкое распространение по всему миру среди частных пользователей.

С ростом вычислительной мощности компьютеров стали возникать вопросы о криптостойкости DES’а перед вскрытием методом "грубой силы", но стандарт успешно проходил повторные сертификации, проводившиеся в 1983, 1988 и 1993. Хотя к середине 90х годов стало очевидным несоответствие общепринятого стандарта шифрования DES современным требованиям. В первую очередь из-за недостаточной длины ключа всего в 56 бит. По данным Брюса Шнайера уже в 1993 году существовали устройства способные вскрыть DES за приемлемое время.

Процесс разработки нового федерального информационного стандарта (FIPS) для шифрования данных Advanced Encryption Standard (AES) был инициирован Национальным Институтом стандартов и технологий (NIST).

В начале января 1997 года NIST объявил о начале разработки AES, выпустив документ “Announcing development of a federal information processing standard for advanced encryption standard”, содержащий первичные требования к алгоритму.

1. Алгоритм шифрования AES должен быть открыто опубликован.
2. Алгоритм должен быть симметричным блочным шифром.
3. AES должен предусматривать возможность увеличения длины ключа.
4. AES должен быть легко реализуем и аппаратной и программно.
5. AES должен распространяться бесплатно или быть общедоступным в рамках патентов ANSI.

В сентябре 1997 года вышел уточняющий документ “Call for AES Candidate Algorithms”, объявлявший о проведении конкурса на AES и содержащий официальные требования к кандидатам. В частности алгоритм должен поддерживать следующие комбинации длин блока и ключа 128-128, 128-192 и 128-256 битов.

К 15 июня 1998 году был заявлен 21 криптографический алгоритм, но только 15 из которых удовлетворяли первоначальным требованиям

| Алгоритмы | Авторы или организация, страна | Основные результаты анализа |
|---|---|---|
| CAST-256 | Entrust Technologies, Inc. Канада | В алгоритме не обнаружено уязвимостей. Однако, скорость шифрования у данного алгоритма невысока; кроме того, у него высокие требования к оперативной и энергонезависимой памяти. |
| Crypton | Future Systems, Inc. Корея | Алгоритм без явных недостатков. Однако, Crypton уступает по всем характеристикам похожему на него алгоритму Rijndael. Эксперты конкурса сочли, что в финале конкурса Crypton однозначно проиграет Rijndael, поэтому не выбрали его во второй раунд |
| DEAL | Richard Outerbridge, Lars Knudsen Канада | Множество недостатков: наличие подмножеств слабых ключей, подверженность дифференциальному криптоанализу, отсутствие усиления при использовании 192-битных ключей по сравнению с 128-битными. |
| DFC | CNRS — Centre National pour la Recherche Scientifique — Ecole Normale Superieure Франция | Достоинство алгоритма: очень высокая скорость шифрования на 64-битных платформах. При этом алгоритм весьма медленно шифрует на остальных платформах, а также имеет классы слабых ключей. |

| | | |
|---|---|---|
| E2 | NTT — Nippon Telegraph and Telephone Corporation Япония | Аналогично алгоритму CAST-256, в E2 не найдено уязвимостей, но требования к энергонезависимой памяти слишком высоки. |
| FROG | TecApro Internacional S.A. Коста Рика | Алгоритм с наиболее оригинальной структурой и с наибольшим количеством недостатков: наличие уязвимостей, низкая скорость шифрования и высокие требования к ресурсам. |
| HPC | Richard Schroeppel США | Аналогично алгоритму DFC, HPC очень быстро шифрует на 64-битных платформах, но весьма медленно на остальных. Кроме того, сложная и медленная процедура расширения ключа ограничивает возможные применения алгоритма, а наиболее сложная из всех представленных на конкурс алгоритмов структура раунда усложняет анализ алгоритма и делает недоказуемым отсутствие |
| LOKI97 | Lawrie Brown, Josef Pieprzyk, Jennifer Seberry Австралия | Низкая скорость шифрования, высокие требования к ресурсам, наличие уязвимостей. |
| Magenta | Deutsche Telekom AG Германия | Алгоритм подвержен атакам методами линейного[2] и дифференциального криптоанализа; медленная скорость шифрования. |
| MARS | IBM США | У алгоритма отсутствуют серьезные недостатки, за исключением низкой скорости шифрования на ряде платформ, не имеющих аппаратной поддержки требуемых операций и некоторых других незначительных недостатков. Алгоритм был незначительно модифицирован в течение первого раунда; измененный вариант вышел в финал конкурса. |
| RC6 | RSA Laboratories США | RC6 имеет весьма похожие на MARS проблемы с реализацией на некоторых платформах. Эксперты посчитали это незначительным недостатком — алгоритм вышел в финал конкурса. |
| Rijndael | Joan Daemen, Vincent Rijmen Бельгия | Каких-либо недостатков у алгоритма не обнаружено; алгоритм вышел в финал конкурса. |

| | | |
|---|---|---|
| SAFER+ | Cylink Corporation США | Алгоритм подвержен ряду атак и имеет низкую скорость выполнения. |
| Serpent | Ross Anderson, Eli Biham, Lars Knudsen. Англия, Норвегия, Израиль | Выявлены некоторые незначительные недостатки, алгоритм вышел в финал конкурса. |
| Twofish | Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson США | Из недостатков эксперты отметили сложность алгоритма, затрудняющую его анализ. Алгоритм вышел в финал конкурса. |

В конце августа 1998 года в Калифорнии состоялась конференция, посвященная отбору кандидатов на AES, получившая название AES1. На конференции давались краткие описания алгоритмов шифрования, и были даны ответы на накопленные вопросы.

В качестве основных критериев оценки алгоритмов были :

- **Криптостойкость**. Проводилось сравнение алгоритмов на степень независимости выходного блока от случайной перестановки битов входного блока, подверженность известным криптоатакам. Каждый кандидат должен был представить оценочную криптостойкость алгоритма.
- **Стоимость.** Кандидат предоставлял информацию о лицензионных соглашениях и патентах на алгоритм. Также учитывалась производительность алгоритма (скорость шифрования), необходимый для шифрования размер памяти.
- **Особенности алгоритма и его реализации.** Гибкость, аппаратное и программное удобство реализации.

В апреле 1999 года в Риме состоялась конференция AES2, в рамках которой проводились сравнение производительности программных реализаций алгоритмов с оптимизациями под языки C и Java. Наибольшую скорость шифрования/дешифрования показал алгоритм Crypton (40 Мбайт/с), наименьшую Magenta и HPC (2Мбайт/с). Хотя при проведении тестов на разных платформах и с различными компиляторами, полученные результаты довольно сильно различались.

Все блочные алгоритмы можно разбить на три основные группы:

1. использующие сети Фейстеля (CAST-256, DEAL, DFC, E2, LOKI97, MAGENTA, MARS, RC6, TWOFISH)
2. сети перестановок-подстановок (SP-сети) основанные на последовательных перестановках и подстановках, зависящих от ключа (CRYPTON, Rijndael, SAFER+ и SERPENT).
3. алгоритмы FROG и HPC не подпадали ни под одну из категорий, но входе обсуждения кандидатов не выявлено каких-либо выдающихся качеств данных алгоритмов.

В результате первичного обсуждения была выявлена слабая криптостойкость алгоритма MAGENTA, вскоре появились данные по криптоанализу алгоритмов FROG,

LOKI, показывающие слабости алгоритмов относительно других алгоритмов. Также часть алгоритмов имели низкие шансы на успех из-за крайне малой скорости шифрования/дешифрования.

Отбор финалистов по названным критериям продолжался до начала августа 1999г.

9 августа 1999г NIST выпустила пресс-релиз “Announces AES Finalists”, в котором объявлялось о пяти финалистах:

***MARS, RC6, Rijndael, Serpent, TwoFish***.

**Serpent**

Создателями алгоритма являются трое известных криптоаналитиков:

- Эли Бихам (один из создателей дифференциального криптоанализа),
- Росс Андерс,
- Ларс Кнудсен (также принимал участие в создании алгоритма DEAL).

Профессионалам криптоанализа действительно удалось создать алгоритм криптостойкий ко всем известным атакам, и до сих пор слабых сторон данного алгоритма не обнаружено.

Алгоритм представляет собой 32 раундовую сеть перестановок-подстановок (SP-сеть). В алгоритме используются комбинации S-box’ов и линейных преобразований, похожие на преобразования в DES, что упрощает анализ криптостойкости алгоритма, а для увеличения стойкости алгоритма количество раундов удвоено.

Согласно данным, представленным создателями протокола, 16 раундовый Серпент по криптостойкости аналогичен 3DES, а по быстродействию оптимизированный алгоритм в два раза быстрее DES. Хотя относительно других кандидатов скорость шифрования явно ниже. Согласно тестам, проводившимся на Intel-Pentium 200 МГц, алгоритмы Rijndael, RC6, E2, Mars показали скорость около 10 Мбайт/с, а Serpent только 14.7 Мбит/с.

**Mars**

Коропорация IBM, создавшая DES, представила алгоритм MARS, обладающий как хорошей криптостойкостью так и высокой скоростью шифрования.

Непосредственное шифрование представляет собой сеть Фейстеля с 4 ветвями.

MARS поддерживает переменную длину ключа от 128 до 448 битов, используя процедуру расширения входного ключа до 40 32-битовых слов, которые используются при шифровании и дешифровании.

Одним из недостатков алгоритма является сложность его криптоанализа из-за использования двойного перемешивания. Алгоритм показал хорошую скорость шифрования. Скорость шифрования на Intel-Pentium 200 МГц достигала 65 Мбит/с, скорость выполнения блока прямого и обратного шифрования достигала 100 Мбит/с.

**RC6**

Алгоритм, разработанный RSA Laboratories, является видоизменным алгоритмом RC5 для соответствия требованиям AES. Алгоритм построен на сети Фейстеля с четырьмя ветвями и имеет 20 раундов.

Характерной особенностью алгоритма является отказ от привычных Sbox’ов и введение операция квадратичной трансформации. Простота и высокая скорость шифрования являются его основными достоинствами, к тому же RC6 имеет наибольшую скорость среди финалистов (12.6 Мбайт/с)

**TwoFish**

Алгоритм, разработанный Шнайером, является модификацией алгоритма BlowFish, созданного в 1993г. Алгоритм TwoFish основан на 16 раундовой сети Фейстеля.

В алгоритме используются преобразование PHT (Pseudo-Hadamard Transforms), MDS матрицы, зависимые от ключа Sbox’ы, по сравнению с другими алгоритмами TwoFish имеет довольно сложную структуру.

В середине апреля 2000 года в Нью-Йорке состоялась конфренция AES3, в которой приняли участие более 250 криптографов. На конференции подводились итоги по криптоанализу финалистов. Одним из интересных полученных результатов является максимальное количество раундов, при котором наилучшая из известных криптографических атак на данный алгоритм более эффективна, чем взлом методом грубой силы 256 битного ключа.

2 октября 2000 NIST объявил, что алгоритм **Rijndael** выбран в качестве Advanced Encryption Standard.

Лекция 7-8

## 1. ОПИСАНИЕ АЛГОРИТМА AES

### 1.2. Обозначения, используемые в описании алгоритма

В табл. 1.2 приведены основные обозначения функций и параметров ариптопреобразования AES.

Таблица 1.2

Параметры алгоритма, символы и функции

| Обозначение | Смысл обозначения |
|---|---|
| *AddRoundKey*() | Преобразование в Шифровании и Дешифровании, в котором Раундовый ключ добавляется к *state*, используя операцию XOR. |
| *InvMixColumns*() | Преобразование в Дешифровании, которое является инверсией *MixColumns*. |
| *InvShiftRows*() | Преобразование в Дешифровании, которое является инверсией *ShiftRows*. |
| *InvSubBytes*() | Преобразование в Дешифровании, которое является инверсией *SubBytes*. |
| *K* | Ключ шифрования — массив из 128 бит или 16 байт. |
| *MixColumns*() | Преобразование в процессе Шифрования, которое берет все столбцы *state* и смешивает их данные (независимо друг от друга), чтобы получить новые столбцы. |
| *Rcon*() | Массив постоянных раундовых Слов. |
| *RotWord*() | Функция используемая в операции Расширения ключа, она берет четырехбайтовое слово и выполняет циклическую перестановку. |

| | |
|---|---|
| *ShiftRows*() | Преобразование в процессе Шифрования, которое выполняется над *state*, циклически сдвигая последние 3 строки *state* на различные значения. |
| *SubBytes*() | Преобразование в процессе Шифрования, которое выполняется над *state* и состоит в замене каждого байта, используя таблицу замены (*S-box*). |
| $N_k$ | Длина ключа в словах, для AES 4 слова. |
| $N_b$ | Длина блока в словах, для AES 4 слова. |
| $N_r$ | Число раундов при шифровании – 10. |
| XOR | Операция «исключающее ИЛИ» над битами. |
| ⊗ | Операция «исключающее ИЛИ» над битами. |
| • | Умножение в конечном поле. |

## МАТЕМАТИЧЕСКИЕ ОСНОВЫ AES

Алгоритм AES оперирует байтами, т.е. числами от 0 до 255 или 8-и разрядными двоичными числами. Все байты в AES интерпретируются как элементы конечного поля $F(2^8)$. Конечное поле $F$ — это множество произвольных элементов (членов множества), содержащее конечное число членов. Для элементов поля определены две бинарные операции: сложение двух элементов поля и умножение двух элементов поля, результатом которых для любых двух элементов поля является элемент поля. Среди элементов поля есть элемент «ноль», сложение с которым любого элемента дает в результате этот же элемент и умножение на который любого элемента дает элемент «ноль». Среди элементов поля есть элемент «единица», умножение на который любого элемента дает в результате этот же элемент. Для каждого элемента поля существует «мультипликативно обратный» элемент, произведение которых дает в результате элемент «единица».

Элементы конечного поля могут складываться и умножатся, но эти операции несколько отличаются от операций используемых для простых чисел.

### 1.4. Сложение

Сложение выполняется с помощью операции побитового сложения по модулю 2. Операция обозначается XOR или ⊕ и выполняется над двоичным представление числа поразрядно так, что $1 \oplus 1 = 0$, $1 \oplus 0 = 0 \oplus 1 = 1$, $0 \oplus 0 = 0$.

Для двух байт $A = \{a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0\}$, где $a_i$ - $i$-й разряд двоичного представления числа $A \in [0..255]$ и $B = \{b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0\}$ сумма будет равна байту $C = \{c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0\}$, где $c_i = a_i \oplus b_i$.

### 1.6. Умножение

Для умножения используется полиномиальное представление двоичного числа, так байт $A = \{a_7, a_6, a_5, a_4, a_3, a_2, a_1, a_0\}$ может быть представлен в виде полинома

$$a(x) = a_7 \cdot x^7 + a_6 \cdot x^6 + a_5 \cdot x^5 + a_4 \cdot x^4 + a_3 \cdot x^3 + a_2 \cdot x^2 + a_1 \cdot x^1 + a_0 \cdot x^0, \quad (1.2)$$

где $a_i$ - принимает значения 0 или 1.

В полиномиальном представлении, умножение (обозначенное •) в конечном поле $F(2^8)$ (здесь число в скобках $2^8$ указывает на число элементов конечного поля) выполняется между полиномами по модулю неприводимого в поле $F$ полинома степени 8. Полином неприводим[1)] в поле $F$, если он делится только на себя и на единицу поля. Для алгоритма AES неприводимый полином это:

$$m(x)=x^8+x^4+x^3+x+1. \quad (1.4)$$

Результат умножения двух байтов $A$ и $B$ есть остаток от деления произведения полиномиального представления (1.2) для $A$ на аналогичное полиномиального представление для $B$, выполненного по обычным правилам произведения полиномов на неприводимый полином $m(x)$ (1.4).

Например, вычислим произведение

$$57 \bullet 83.$$

Полиномиальное представление 57 есть

$$x^6+x^4+x^2+x+1,$$

полиномиальное представление 83 есть

$$x^7+x+1.$$

Результат произведения полиномов

$$(x^6+x^4+x^2+x+1)(x^7+x+1)=x^{13}+x^{11}+x^9+x^8+x^7+x^7+x^5+x^3+x^2+x+x^6+x^4+x^2+x+1.$$

Суммирование коэффициентов при одинаковых степенях производится по правилам сложения в поле, т.е. по модулю 2, например

$$1 \cdot x^7+1 \cdot x^7 = (1 \oplus 1) \cdot x^7 = 0 \cdot x^7.$$

Получим

$$(x^6+x^4+x^2+x+1)(x^7+x+1)=x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1. \quad (1.6)$$

Вычисляя остаток от деления (1.6) на (1.4), получим

$$(x^{13}+x^{11}+x^9+x^8+x^6+x^5+x^4+x^3+1) \bmod (x^8+x^4+x^3+x+1)=x^7+x^6+1,$$

что является полиномиальным представлением числа 193. Таким образом

$$57 \bullet 83=193.$$

Взятие по модулю $m(x)$ гарантирует, что результат будет двоичным полиномом степени меньше 8, и таким образом может быть представлен байтом. В отличии от операции сложения, нет другой простой операции на уровне битов, чтобы выполнить умножение.

# 2. АЛГОРИТМ AES

## 2.2. Шифрование

Для шифрования необходим ключ. Размер ключа в алгоритме AES 128 бит, обычно ключ представляют как матрицу 4×4 байта.

В начале процесса шифрования, входные данные разбиваются на блоки размером 16 байт или 128 бит. Если полный размер данных не кратен 16 байтам — данные дополняются до размера кратного 16 байтам. Блок данных в алгоритме AES называется *state* и обычно представляется в виде матрицы 4×4 байта, см. рис. 2.2. Операция шифрования каждого блока данных проводится независимо от содержимого других блоков. По окончанию шифрования блока — матрица заполняется следующей порцией данных и процесс повторяется. В силу независимости шифрования одного блока от другого процесс шифрования хорошо поддается распараллеливанию.

Представление блока данных AES — state

| | | | |
|---|---|---|---|
| $S_{00}$ | $S_{01}$ | $S_{02}$ | $S_{03}$ |
| $S_{10}$ | $S_{11}$ | $S_{12}$ | $S_{13}$ |
| $S_{20}$ | $S_{21}$ | $S_{22}$ | $S_{23}$ |
| $S_{30}$ | $S_{31}$ | $S_{32}$ | $S_{33}$ |

Рис. 2.2

Каждый блок шифруется в несколько этапов — раундов. Схема криптопреобразования может быть записана так.

1.Расширение ключа *KeyExpansion*.

2.Начальная операция — *AddRoundKey* — суммирование с основным ключом.

3.9 раундов из четырех шагов каждый.

3.1.*SubBytes* — замена байтов *state* по таблице замен.

3.2.*ShiftRows* — циклический сдвиг строк *state*.

3.3.*MixColumns* — перестановка столбцов *state*.

3.4.*AddRoundKey* — суммирование с раундовым ключом.

4.Заключительный 10-й раунд

4.1.*SubBytes* — замена байтов *state* по таблице замен.

4.2.*ShiftRows* — циклический сдвиг строк *state*.

4.3.*AddRoundKey* — суммирование с раундовым ключом.

Далее подробно описано каждое из преобразований.

### 2.2.2. Преобразование *SubBytes*

Преобразование *SubBytes* – это нелинейная замена байт, проводящаяся над каждым байтом *state*, используя таблицу замены *S-box*, см. табл.2.2. Цель применения таблицы замен — затруднить линейный и дифференциальный криптоанализ. Таблица замен в алгоритме

AES фиксированная. В табл.2.2 числа представлены в шестнадцатеричной системе счисления, в этой системе счисления любое значение байта представимо не более чем двумя шестнадцатеричными разрядами.

Таблица 2.2

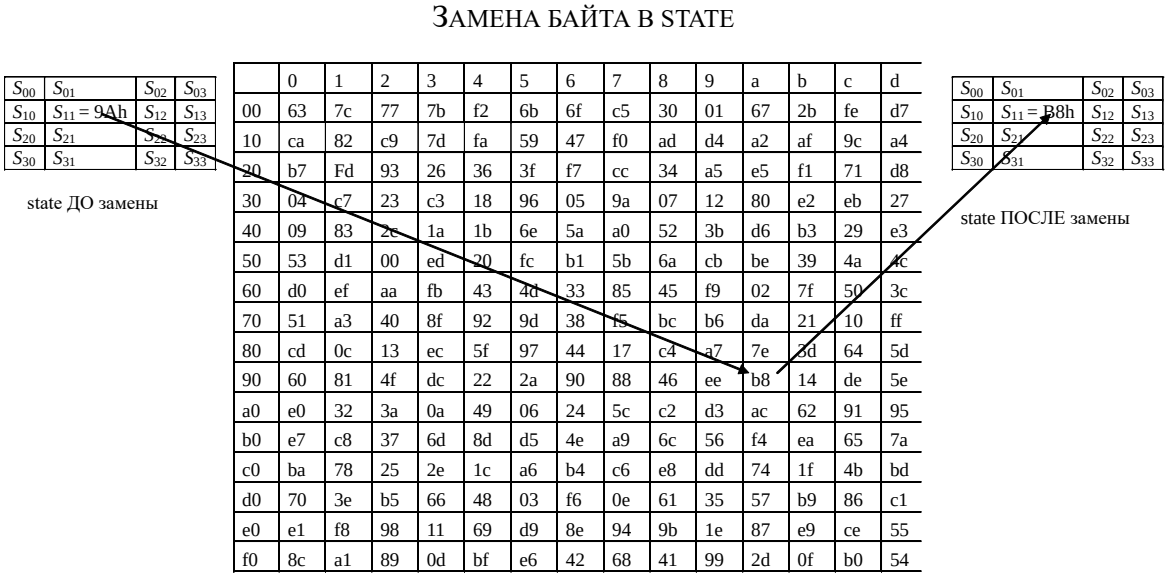
ТАБЛИЦА ЗАМЕНЫ БАЙТ *S-BOX* АЛГОРИТМА AES

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 63 | 7c | 77 | 7b | f2 | 6b | 6f | c5 | 30 | 01 | 67 | 2b | fe | d7 | ab | 76 |
| 10 | ca | 82 | c9 | 7d | fa | 59 | 47 | f0 | ad | d4 | a2 | af | 9c | a4 | 72 | c0 |
| 20 | b7 | Fd | 93 | 26 | 36 | 3f | f7 | cc | 34 | a5 | e5 | f1 | 71 | d8 | 31 | 15 |
| 30 | 04 | c7 | 23 | c3 | 18 | 96 | 05 | 9a | 07 | 12 | 80 | e2 | eb | 27 | b2 | 75 |
| 40 | 09 | 83 | 2c | 1a | 1b | 6e | 5a | a0 | 52 | 3b | d6 | b3 | 29 | e3 | 2f | 84 |
| 50 | 53 | d1 | 00 | ed | 20 | fc | b1 | 5b | 6a | cb | be | 39 | 4a | 4c | 58 | cf |
| 60 | d0 | ef | aa | fb | 43 | 4d | 33 | 85 | 45 | f9 | 02 | 7f | 50 | 3c | 9f | a8 |
| 70 | 51 | a3 | 40 | 8f | 92 | 9d | 38 | f5 | bc | b6 | da | 21 | 10 | ff | f3 | d2 |
| 80 | cd | 0c | 13 | ec | 5f | 97 | 44 | 17 | c4 | a7 | 7e | 3d | 64 | 5d | 19 | 73 |
| 90 | 60 | 81 | 4f | dc | 22 | 2a | 90 | 88 | 46 | ee | b8 | 14 | de | 5e | 0b | db |
| a0 | e0 | 32 | 3a | 0a | 49 | 06 | 24 | 5c | c2 | d3 | ac | 62 | 91 | 95 | e4 | 79 |
| b0 | e7 | c8 | 37 | 6d | 8d | d5 | 4e | a9 | 6c | 56 | f4 | ea | 65 | 7a | ae | 08 |
| c0 | ba | 78 | 25 | 2e | 1c | a6 | b4 | c6 | e8 | dd | 74 | 1f | 4b | bd | 8b | 8a |
| d0 | 70 | 3e | b5 | 66 | 48 | 03 | f6 | 0e | 61 | 35 | 57 | b9 | 86 | c1 | 1d | 9e |
| e0 | e1 | f8 | 98 | 11 | 69 | d9 | 8e | 94 | 9b | 1e | 87 | e9 | ce | 55 | 28 | df |
| f0 | 8c | a1 | 89 | 0d | bf | e6 | 42 | 68 | 41 | 99 | 2d | 0f | b0 | 54 | bb | 16 |

Замена байта по таблице *S-box* производится так:

1. Байт *Z* преобразуется в шестнадцатеричную систему счисления, например *XY*h, *X* — старший разряд, *Y* — младший разряд. Если старшего разряда нет — он заменяется нулем.
2. В *S-box* выбирается строка *X* и столбец *Y*.
3. Значение *Z'* на пересечении строки *X* и столбца *Y* таблицы *S-box* используется как замена *Z*.

Например, для значения байта *Z*=9Ah, заменой согласно таблицы *S-box* будет *Z'*=B8h. Полный процесс *SubBytes* состоит в замене всех 16 байт матрицы *state.*




Пример замены одного байта с помощью *S-box*.

Рис. 2.4

ПРИМЕЧАНИЕ. Нет никакой необходимости использовать таблицу *S-box* в том виде, в котором она приведена в описании AES. Проще представить ее в виде одномерного массива байт, где индексом служит значение байта, а содержимым — содержимое соответствующей ячейки *S-box*. Например, *S-arr*={63h, 7Ch, 77h, …}. Процесс замены по такой таблице сводится к *Z'*=*S-arr*[*Z*], что заметно быстрее. Размер памяти, требующийся для размещения таблицы замен не изменится.

### 2.2.4. Преобразование *ShiftRows*

В преобразовании *ShiftRows* байты в последних трех строках *state* циклически смещаются влево на различное число байт. Строка 1 (нумерация строк с нуля, см. рис.2.2) смещается на один байт, строка 2 – на два байта, строка 3 – на три байта. На рис.2.6 проиллюстрировано применение преобразования *ShiftRows* к *state*.

ПРЕОБРАЗОВАНИЕ СДВИГА AES — *SHIFTROWS*

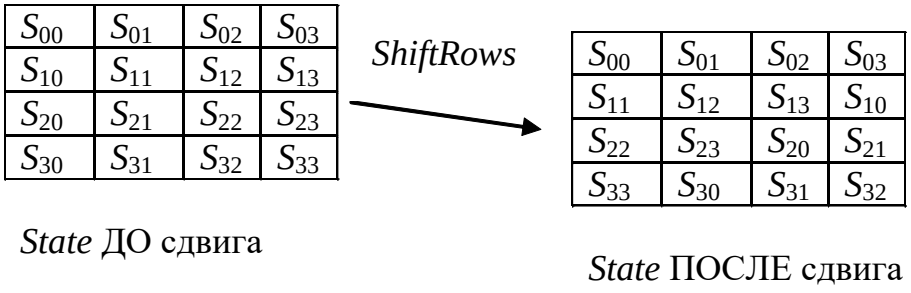



Рис. 2.6

### 2.2.6. Преобразование *MixColumns*

В преобразовании *MixColumns* — перемешивание столбца — столбцы состояния (*state*) рассматриваются как полиномы над полем $F(2^8)$ и умножаются по модулю $x^4+1$ на постоянный полином:

$$a(x)=3x^3+1x^2+1x^2+2 \tag{2.2}$$

Процесс умножения полиномов эквивалентен матричному умножению

$$\begin{bmatrix} S'_{0c} \\ S'_{1c} \\ S'_{2c} \\ S'_{3c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \bullet \begin{bmatrix} S_{0c} \\ S_{1c} \\ S_{2c} \\ S_{3c} \end{bmatrix},$$

где $c$ – номер столбца массива *state* и $0 \le c \le 3$. Операция сложения и умножения чисел выполняется по правилам, описанным в пунктах 1.4 и 1.6.

В результате такого умножения, байты столбца $c$ $\{S_{0c},S_{1c},S_{2c},S_{3c}\}$ заменяются, соответственно, на байты

$$\begin{aligned} S'_{0c}&=(2\bullet S_{0c})\oplus(3\bullet S_{1c})\oplus S_{2c}\oplus S_{3c};\\ S'_{1c}&=S_{0c}\oplus(2\bullet S_{1c})\oplus(3\bullet S_{2c})\oplus S_{3c};\\ S'_{2c}&=S_{0c}\oplus S_{1c}\oplus(2\bullet S_{2c})\oplus(3\bullet S_{3c});\\ S'_{3c}&=(3\bullet S_{0c})\oplus S_{1c}\oplus S_{2c}\oplus(2\bullet S_{3c}). \end{aligned} \tag{2.4}$$

Преобразование (2.4) применяется к каждому из четырех столбцов *state*.

### 2.2.8. Преобразование *AddRoundKey*

В преобразовании *AddRoundKey*, раундовый ключ *RK* добавляется к *state* посредством поразрядного XOR. Каждый раундовый ключ состоит из 16 байт расширенного ключа. Байты раундового ключа записываются в матрицу 4×4, подобную *state*. Каждый байт раундового ключа суммируется с соответствующим байтом из *state*, как показано на рис.2.8.

СУММИРОВАНИЕ С РАУНДОВЫМ КЛЮЧОМ — *ADDROUNDKEY*

Рис. 2.8

### 2.2.10. Расширение ключа *KeyExpansion*

Алгоритм AES берет ключ шифрования $K$ и выполняет операцию расширения ключа, чтобы создать набор данных для раундовых ключей. Расширенный ключ $W$ содержит $4\times(10+1)$ слов — начальный ключ в 4 слова и по 4 слова расширенного ключа на каждый из 10 раундов. Расширенный ключ $W$ состоит из слов (четыре байта на слово), обозначаемых ниже как $w_i$, где $i$ находится в диапазоне [0..44]. Полная длина расширенного ключа 1408 бит, по 128 бит на каждый раунд.

В процессе расширения ключа используется массив констант *Rcon*. Элементы массива *Rcon* пронумерованы от 1 до 256+3. Значения элементов массива определены так

$Rcon_1=1$;

$Rcon_k=2\bullet Rcon_{k-1}=2^{k-1}$, для $k=2, 3,\ldots255$;

$Rcon_k=0$, для $k=256,257,258$;

Расширение ключа можно описать следующей последовательностью операций.

1) Четыре слова ключа шифрования $K$ копируются в первые четыре слова расширенного ключа $W$: $w_i=k_i$ для $i=0,1,2,3$.

2) Остальные слова расширенного ключа $W$ для $i=4,5,\ldots44$ генерируются так:

- если $i$ кратно 4, то $w_i=SubBytes(RotByte(w_{i-1}))\oplus Rcon_{(i/4)}$;
- если $i$ не кратно 4, то $w_i=w_{i-4}\oplus w_{i-1}$.

Функция *RotByte* переставляет четыре байта исходного слова $\{a_0,a_1,a_2,a_3\}$ с помощью циклической перестановки, превращая в слово $\{a_3,a_1,a_2,a_0\}$. Функция *SubBytes* применяет к каждому из четырех байтов слова замену по таблице *S-box*, как это описано в пункте 2.2.2.

#### Выбор раундового ключа

Раундовый ключ $RK$ для раунда $k$ выбирается из расширенного ключа $W$ как слова с $w_{4k}$ по $w_{4(k+1)}$.

## 2.4. ДЕШИФРОВАНИЕ

Все преобразования шифрования однозначны и, следовательно, имеют обратное преобразование, т.е. могут быть инвертированы и выполнены в обратном порядке, чтобы выполнить дешифрование для алгоритма AES.

Схема криптопреобразования может быть записана так.

1.Расширение ключа *KeyExpansion*.

3.9 раундов из четырех шагов каждый.

3.1.*AddRoundKey* — суммирование с раундовым ключом.

3.2.*InvMixColumns* — обратная перестановка столбцов *state*.

3.3.*InvShiftRows* — обратный циклический сдвиг строк *state*.

3.4.*InvSubBytes* — обратная замена байтов *state* по таблице замен.

4.Заключительный 10-й раунд

4.1.*AddRoundKey* — суммирование с раундовым ключом.

4.2.*InvShiftRows* — обратный циклический сдвиг строк *state*.

4.3.*InvSubBytes* — обратная замена байтов *state* по таблице замен.

Далее подробно описано каждое из преобразований.

### 2.4.2. Преобразование *InvMixColumns*

Преобразование *InvMixColumns* является обратным для преобразования *MixColumns.* В преобразовании *InvMixColumns*, столбцы состояния (*state*) рассматриваются как полиномы над полем $F(2^8)$ и умножаются по модулю $x^4+1$ с постоянным полиномом $d(x)=a^{-1}(x)$, в поле $F(2^8)$:

$$d(x)=0\text{Bh}x^3+0\text{Dh}x^2+09\text{h}x+0\text{Eh}. \quad (2.6)$$

Подробнее о процессе умножения см. пункт 2.2.6.

### 2.4.4. Преобразование *InvShiftRows*

Преобразование *InvShiftRows* обратно преобразованию *ShiftRows.* Байты последних трех рядов массива *state* циклически сдвигаются вправо. Строка 1 (нумерация с нуля) смещается на 1 байт, строка 2 – на 2 байта, строка 3 – на 3 байта.

ПРЕОБРАЗОВАНИЕ ОБРАТНОГО СДВИГА AES — *INVSHIFTROWS*

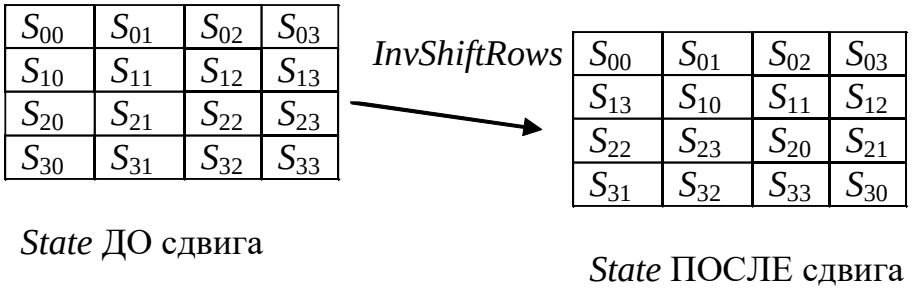



Рис. 2.10

### 2.4.6. Преобразование *InvSubBytes*

Преобразование *InvSubBytes* выполняет обратную замену байт с помощью обратной таблицы замен, назовем ее *InvS-box*. Обратная таблица замен приведена:

Таблица 2.4

ТАБЛИЦА *INVS-BOX* ДЛЯ ОБРАТНОЙ ЗАМЕНЫ БАЙТ

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 52 | 09 | 6a | d5 | 30 | 36 | a5 | 38 | bf | 40 | a3 | 9e | 81 | f3 | d7 | fb |
| 10 | 7c | e3 | 39 | 82 | 9b | 2f | ff | 87 | 34 | 8e | 43 | 44 | c4 | de | e9 | cb |
| 20 | 54 | 7b | 94 | 32 | a6 | c2 | 23 | 3d | ee | 4c | 95 | 0b | 42 | fa | c3 | 4e |
| 30 | 08 | 2e | a1 | 66 | 28 | d9 | 24 | b2 | 76 | 5b | a2 | 49 | 6d | 8b | d1 | 25 |
| 40 | 72 | f8 | f6 | 64 | 86 | 68 | 98 | 16 | d4 | a4 | 5c | cc | 5d | 65 | b6 | 92 |
| 50 | 6c | 70 | 48 | 50 | fd | ed | b9 | da | 5e | 15 | 46 | 57 | a7 | 8d | 9d | 84 |
| 60 | 90 | d8 | ab | 00 | 8c | bc | d3 | 0a | f7 | e4 | 58 | 05 | b8 | b3 | 45 | 06 |
| 70 | d0 | 2c | 1e | 8f | ca | 3f | 0f | 02 | c1 | af | bd | 03 | 01 | 13 | 8a | 6b |
| 80 | 3a | 91 | 11 | 41 | 4f | 67 | dc | ea | 97 | f2 | cf | ce | f0 | b4 | e6 | 73 |
| 90 | 96 | ac | 74 | 22 | e7 | ad | 35 | 85 | e2 | f9 | 37 | e8 | 1c | 75 | df | 6e |
| a0 | 47 | f1 | 1a | 71 | 1d | 29 | c5 | 89 | 6f | b7 | 62 | 0e | aa | 18 | be | 1b |
| b0 | fc | 56 | 3e | 4b | c6 | d2 | 79 | 20 | 9a | db | c0 | fe | 78 | cd | 5a | f4 |
| c0 | 1f | dd | a8 | 33 | 88 | 07 | c7 | 31 | b1 | 12 | 10 | 59 | 27 | 80 | ec | 5f |
| d0 | 60 | 51 | 7f | a9 | 19 | b5 | 4a | 0d | 2d | e5 | 7a | 9f | 93 | c9 | 9c | ef |

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| e0 | a0 | e0 | 3b | 4d | ae | 2a | f5 | b0 | c8 | eb | bb | 3c | 83 | 53 | 99 | 61 |
| f0 | 17 | 2b | 04 | 7e | ba | 77 | d6 | 26 | e1 | 69 | 14 | 63 | 55 | 21 | 0c | 7d |

Процесс замены байта по таблице подробно описан в пункте 2.2.2.

### 2.4.8. Преобразование обратное *AddRoundKey*

Преобразование *AddRoundKey*, описанное в пункте 2.2.8, является обратным для самого себя, так как использует операцию *XOR.*

**Лекция 9-10**

## ОТКРЫТОЕ РАСПРОСТРАНЕНИЕ КЛЮЧЕЙ КРИПТОГРАФИЧЕСКОГО ПРЕОБРАЗОВАНИЯ

На практике для защиты передаваемых по каналам связи сообщений от несанкционированного доступа к их содержимому широко используется передача этих сообщений в преобразованном виде. При этом преобразование сообщений осуществляется с помощью систем, реализующих различные криптографические преобразования. В таких системах выбор конкретного преобразования осуществляется с помощью ключа криптографического преобразования, который должен храниться в тайне как на передающей, так и на приемной сторонах.

В этой связи необходимо заметить, что если даже для преобразования сообщений используются самые лучшие из известных криптографических преобразований, их применение окажется бесполезным в случае попадания используемых ключей в руки злоумышленников. Поэтому распространение ключей криптографического преобразования является одной из основных проблем при обеспечении защиты сообщений от не санкционированного доступа.

Рассмотрим способ, являющийся наиболее простыми и эффективными при распространении ключей криптографического преобразования. Заметим, что основой для разработки таких способов послужило замечание К.Шеннона, сделанное в статье «Теория связи в секретных системах» о том, что *«проблема создания хорошего шифра является, по существу, проблемой нахождения наиболее сложных задач, удовлетворяющих определенным условиям... Можно составить наш шифр таким образом, чтобы раскрытие его было эквивалентно ... решению некоторой проблемы, про которую известно, что для ее решения требуется большой объем работ»*. Это замечание нашло отклик в плодотворных работах американских ученых Диффи и Хеллмана. В своей статье «Новые направления в криптологии» они привели результаты исследований, из которых следовал ошеломляющий по тем временам вывод о возможности создания стойких систем, вообще не требующих передачи ключа криптографического преобразования.

Эти системы получили название **систем с открытыми ключами**.

Диффи и Хеллманом было показано, что в основе построения таких систем должны лежать односторонние функции вида $y=\Psi(x)$, которые обладают следующими свойствами:

- значение величины y является легко вычислимым по значению величины x;
- значение величины $x=\Psi^{-1}(y)$ является трудно вычислимым по значению величины x.

При этом «трудновычислимость» понимается в смысле отсутствия другого алгоритма определения значения величины $x=\Psi^{-1}(y)$ по значению величины y за исключением алгоритма тотального перебора возможных значений величины x или определение значения величины $x=\Psi^{-1}(y)$ вычислительно не осуществимо в случае наличия самого экономного из алгоритмов.

В качестве простейших односторонних функций были предложены функции вида

**$y = \gamma^x(modp)$,** где

- x - целое число от 1 до (p-1) включительно;
- γ - целое число, степени которого $\gamma^1, \gamma^2,..., \gamma^{p-1}$ при вычислении по modp в некотором порядке равны 1,2,...,p-1;
- p . большое простое число.

В алгебре целое число γ называется примитивным элементом и известно, что такое число всегда существует.

Например, если p=7, то γ =3. Действительно, $3^1(mod7)=3$, $3^2(mod7)=2$, $3^3(mod7)=6$, $3^4(mod7)=4$, $3^5(mod7)=5$, а $3^6(mod7)=l$.

Поскольку $y = \gamma^x(modp)$, то $x=\log_\gamma y(modp)$. Доказано, что даже при очень больших значениях величины p , например, при $p\approx2^{1000}$, можно достаточно просто вычислить значение величины $y = \gamma^x(modp)$ путем возведения в квадрат, умножения и приведения по modp. В то же время вычисление значения величины $x=\log_\gamma y(modp)$ вызывает существенные трудности.

Это обусловлено тем, что под знаком логарифма находится число, полученное в результате выполнения операции приведения по modq.

Известно, что вычисление значения величины $y = \gamma^x(modp)$ требует не более $2\log_2 p$ простых операций умножения и приведения по modp, а вычисление значения величины $x=\log_\gamma y(modp)$ по самому экономному из известных алгоритмов . порядка $p^{0,5}$ таких операций.

Поэтому, если $p\approx2^{1000}$, то для вычисления величины $x=\log_\gamma y(modp)$ потребуется выполнить порядка 2500 операций, что практически не осуществимо.

Рассмотрим теперь некоторые алгоритмы обмена ключами криптографического преобразования, базирующиеся только на открытых сообщениях.

Самым первым, простым и удобным для практического использования алгоритмом обмена ключами является алгоритм, предложенный Диффи и Хеллманом.

**<u>Содержание этого алгоритма заключается в следующем.</u>**

Предположим, что пользователям A и B известны значения величин γ и q. Пользователь A случайным образом выбирает некоторое целое число α, заключенное в пределах от 1 до p-1, и сохраняет его в секрете. Далее пользователь A вычисляет значение величины $Y\alpha = \gamma^\alpha(modp)$, которое опубликовывается путем помещения этого значения в специальный справочник вместе с адресными данными пользователя A.

Аналогичные операции осуществляет пользователь B.

Если у пользователей A и B возникает необходимость обеспечения защищенной с помощью криптографических преобразований связи, то пользователь A возьмет из справочника значение величины $Y_\beta= \gamma^\beta(modq)$ и с помощью значения α, содержащегося в секрете, вычислит значение $Z_{\alpha,\beta}$ по формуле:

**$Z_{\alpha,\beta}=(Y_\beta)^\alpha=[\gamma^\beta(modq)]^\alpha=\gamma^{\beta\alpha}(modq).$**

Таким же образом пользователь B вычислит значение $Z_{\beta,\alpha}$:

**$Z_{\beta,\alpha}=(Y_\alpha)^\beta=[\gamma^\alpha(\mathrm{mod}q)]^\beta=\gamma^{\alpha\beta}(\mathrm{mod}q)$.**

Из полученных зависимостей следует, что $Z_{\alpha,\beta}=Z_{\beta,\alpha}$. Поэтому пользователи с этого момента времени могут использовать значение величины $Z_{\alpha,\beta}(Z_{\beta,\alpha})$ в качестве обычного ключа криптографического преобразования.

*В качестве иллюстрации рассмотрим пример.*

*Пусть $p=7$, а $\gamma=3$.*

*Пользователь А: случайным образом выбирает число $\alpha=3$, вычисляет $Y_\alpha=\gamma^\alpha(\mathrm{mod}p)=3^3(\mathrm{mod}7)=6$*

*Пользователь В случайным образом выбирает число $\beta=4$, вычисляет $Y_\beta=\gamma^\beta(\mathrm{mod}p)=3^4(\mathrm{mod}7)=4$.*

*Для получения ключа криптографического преобразования пользователи А и В осуществляют следующие действия.*

*Пользователь А берет число $Y_\beta$ и возводит в степень, равную 3, т.к. $\alpha=3$, получая в результате этого число $Z_{\alpha,\beta}=(Y_\beta)^\alpha=4^3(\mathrm{mod}7)=1$.*

*Пользователь В берет число $Y_\alpha$ и возводит его в степень, равную 4, т.к. $\beta=4$, получая число $Z_{\beta,\alpha}=(Y_\alpha)^\beta=6^4(\mathrm{mod}7)=1$.*

*Из рассмотренного следует, что в результате выполнения этих действий у пользователей А и В появляется ключ криптографического преобразования, равный 1.*

Из анализа алгоритма Диффи.Хеллмана следует, что в результате его использования у пользователей формируется одинаковый ключ криптографического преобразования, значение которого до момента формирования им было неизвестно. Очевидно, что это свойство алгоритма позволяет обеспечить более высокую защищенность сообщений от несанкционированного доступа, поскольку ключ практически «возникает» в системе только к моменту начала криптографического преобразования сообщений.

**Лекция 12-13**

## ***Схема шифрования Эль Гамаля***

Схема Эль Гамаля, предложенная в 1985 г., может быть использована как для шифрования, так и для цифровых подписей. Безопасность схемы Эль Гамаля обусловлена сложностью вычисления дискретных логарифмов в конечном поле.

Для того чтобы генерировать пару ключей (открытый ключ - секретный ключ), сначала выбирают некоторое большое простое число ***p*** и большое целое число ***q***, причем ***q < p***. Числа ***p*** и ***q*** могут быть распространены среди группы пользователей.

Затем выбирают случайное целое число x, причем ***x<p***. Число ***x*** является секретным ключом и должно храниться в секрете.

Далее вычисляют $y = q^x \mathrm{mod}\ p$. Число ***y*** является открытым ключом.

Для того чтобы зашифровать сообщение ***m***, выбирают случайное целое число ***k***, ***1<k<p -1***, такое, что числа ***k*** и ***(p-1)*** являются взаимно простыми.

Затем вычисляют числа

$$a = q^k \mathrm{mod}\ p$$

$$b = m * y^k \mathrm{mod}\ p.$$

Пара чисел ***(a,b)*** является шифртекстом. Заметим, что длина шифртекста вдвое больше длины исходного открытого текста ***m***.

Для того чтобы расшифровать шифртекст ***(a,b),*** вычисляют

$$m = \frac{b}{a^x}(mod\ p).\ (*)$$

Поскольку

$$\frac{b}{a^x}(mod\ p) = \frac{m * y^k}{a^x}(mod\ p) = \frac{m * q^{xk}}{q^{xk}}(mod\ p) = m\,(mod\ p)$$

то соотношение (*) справедливо.

**Пример**. Выберем ***p*** = 11, ***q*** = 2, секретный ключ ***x*** = 8. Вычисляем

$$y = q^x\,(mod\ p) = 2^8\,(mod\,11) = 256\ (mod\,11) = 3.$$

Итак, открытый ключ ***y*** = 3.

Пусть сообщение ***m*** = 5. Выберем некоторое случайное число ***k*** = 9. Убедимся, что $НОД\ (k, p-1) = 1$. Действительно, $НОД\ (9,10) = 1$. Вычисляем пару чисел ***(a,b)***:

$$a = q^x\,(mod\ p) = 2^9\,(mod\,11) = 512\ (mod\,11) = 6.$$

$$b = y^x m\,(mod\ p) = 3^9\,(mod\,11) = 19683 * 5\ (mod\,11) = 9.$$

Получим шифртекст ***(a,b)*** = (6, 9).

Выполним расшифрование этого шифртекста. Вычисляем сообщение ***m***, используя секретный ключ ***x***:

$$m = \frac{b}{a^x}(mod\ p) = \frac{9}{6^8}(mod\ 11)$$

Выражение $m = \frac{9}{6^8}(mod\ 11)$ можно представить в виде

$$m * 6^8 = 9\,(mod\ 11) \text{ или } \quad m * 1679616 = 9\,(mod\ 11)$$

Решая данное сравнение, находим ***m***= 5.

В реальных схемах шифрования необходимо использовать в качестве модуля ***p*** большое целое простое число, имеющее в двоичном представлении длину 512... 1024 бит.

**Лекция 15-16**

ГОСТ Р 34.11 – 94. Функция хэширования.

Обозначения:

$\{01\}^*$ множество двоичных строк произвольной конечной длины

$\{01\}^n$ множество двоичных строк конечной длины n

Краткий анализ.

Введение

Криптографические хэш функции играют фундаментальную роль в современной криптографии. Говоря в общем- хэш функция h отображает двоичные строки произвольной конечной длины в выходы небольшой (например, 64, 128, 160,192, 224, 256, 384, 512) фиксированной длины, называемые *хэш величинами* за полиномиальное время:

$h:\{01\}^{2n}\to\{01\}^{n}$, где $\{01\}^{*}\in\bigcup_{i\in N}\{01\}^{i}$ (в ГОСТ Р 34.11 – 94 n= 256). Основная идея криптографических функций состоит в том, чтобы хэш величины служили как компактный репрезентативный образ входной двоичной строки, и их можно использовать как если бы они однозначно отождествлялись с этой строкой, хотя для области определения D и области значений $|D|\gg|R|$ (имеются ввиду мощности множеств), функция типа “множество – один” подразумевает, что существование столкновений (пары входов с одинаковым выходом) неизбежно. Область применения хэш функции четко не оговорена: используется “для реализации процедур электронной цифровой подписи (ЭЦП), при передаче, обработке и хранении информации в автоматизированных системах”.

Сообщения с произвольной длины можно сжать используя хэш функцию с фиксированным размером входа при помощи двух методов:

- последовательного (итерационного);
- параллельного.

Создатели ГОСТ Р 34.11 – 94 пошли по первому пути и использовали метод последовательного хэширования использующий хэш функцию с фиксированным размером входа $h:\{01\}^{2n}\to\{01\}^{n}$ (см. Рис. 1), то есть функцию сжатия с коэффициентом 2.

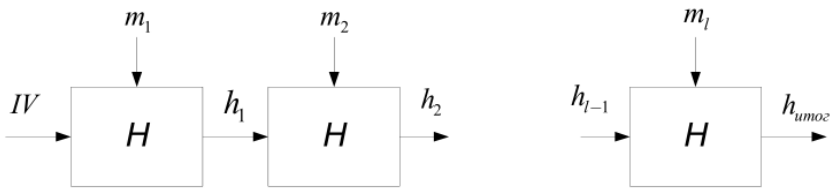



**Рис. 1 Метод последовательного хэширования**

Если необходимо хэшировать сообщение $m=(m_1,m_2,\ldots,m_l)$, , то хэширование выполняется следующим образом:

$h_0\leftarrow IV,$
$h_i\leftarrow H(m_i,h_{i-1}),\quad для\ i=1,2,\ldots,l$
$h_{итог}\leftarrow h_i.$

Здесь $H_i$ -функция сжатия, $h_i$ -переменная сцепления.

Если последний блок меньше чем n бит, то он набивается одним из существующих методов до достижения длины кратной n . В отличие от стандартных предпосылок, что сообщение разбито на блоки и произведена набивка последнего блока (форматирование входа априори), если необходимо, до начала хэширования, то в ГОСТ Р 34.11 – 94 процедура хэширования ожидает конца сообщения (форматирование входного сообщения постериори). Набивка производится следующим образом: последний блок сдвигается вправо, а затем набивается нулями до достижения длины в 256 бит. Алгоритм хэширования по ГОСТ Р 34.11 – 94 можно классифицировать как устойчивый к столкновениям (n= 256, следовательно атака по парадоксу дней рождения потребует приблизительно $2^{256/2}$ операций хэширования) код выявляющий модификации ( Collision Resistant Hash Function, CRHF ). Конструкторы предусмотрели дополнительные меры защиты: параллельно рассчитываются контрольная сумма представляющая собой сумму всех блоков сообщения (последний суммируется уже набитым) по правилу $(A+B)\bmod 2^k$, где $k=|A|=|B|$, а $|A|$ и $|B|$ битовые длины слов A и B (далее на рисунках и в тексте эту операцию будем обозначать значком $\otimes$ ), и битовая длина хэшируемого сообщения приводимая по $\bmod 2^{256}$ (MD - усиление), которые в финальной функции сжатия используются для вычисления итогового хэша (см. Рис. 2).

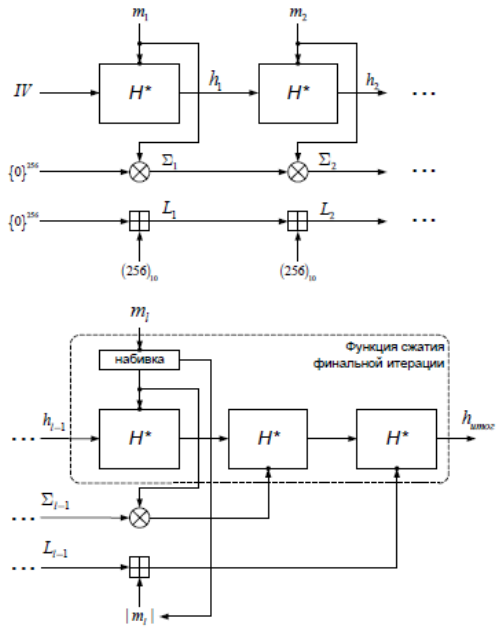



Рис. 2 Общая схема функции хэширования по ГОСТ Р 34.11 – 94

Замечание 1 (набивка): Указывать в передаваемом сообщении сколько было добавлено нулей к последнему блоку не требуется, так как длина сообщения участвует в хэшировании.

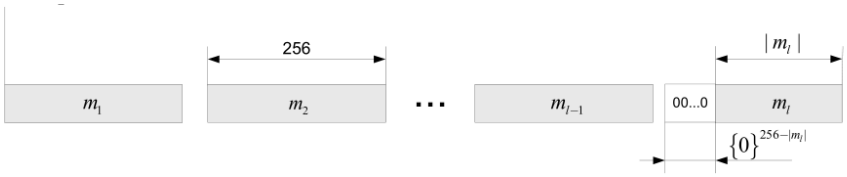



**Рис. 3 Набивка сообщения**

***Замечание 2*** (начальный вектор IV ) Согласно ГОСТ Р 34.11 – 94 IV – произвольное фиксированное слово длиной 256 бит $\left(IV \in \{01\}^{256}\right)$. . В таком случае, если он априорно не известен верифицирующему целостность сообщения, то он должен передаваться вместе с сообщением с гарантией целостности. При небольших сообщениях можно усложнить задачу противнику, если IV выбирается из небольшого множества допустимых величин (но при этом увеличивается вероятность угадывания хэш величины противником). Также он может задаваться в рамках организации, домена как константа.

***Детальное рассмотрение***

1. ***Функция сжатия внутренних итераций (по ГОСТ "шаговая функция хэширования")***

Функция сжатия внутренних итераций $\chi$ отображает два слова длиной 256 бит в одно слово длиной 256 бит:

$$\chi : \{01\}^{226} \times \{01\}^{256} \to \{01\}^{256},$$

и состоит из трех процедур (см. Рис. 4):

- Перемешивающего преобразования (по существу модифицированной схемы Фейстеля),
- Шифрующего преобразования;
- Генерирования ключей.

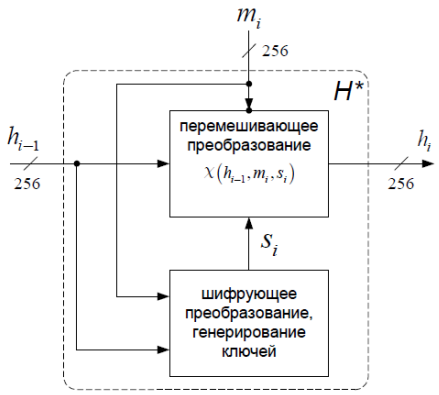



**Рис. 4 Структура функции сжатия в ГОСТ Р 34.11 – 94**

***а) Перемешивающее преобразование***

Здесь использована модифицированная для хэширования перестановка Фейстеля, где n - битный вход разделяется на k блоков по r бит, так чтобы $kl = n$:

$$F_p : \underbrace{\{01\}^r \times \cdots \times \{01\}^r}_{l} \to \underbrace{\{01\}^r \times \cdots \times \{01\}^r}_{l}$$

где p - входное сообщение. Пусть вход $A = (A_1, \ldots, A_l) \in \{01\}^n$ , а выход $B = (B_1, \ldots, B_l) \in \{01\}^n$ , тогда $F_p(A)$ описывается следующим образом:

$$\begin{cases} B_l \leftarrow A_1 \otimes f_p(A_2, \ldots, A_l), \\ B_{l-1} \leftarrow A_l, \ldots, B_1 \leftarrow A_2. \end{cases}$$

Схематически модифицированная функция Фейстеля представлена на Рис. 5

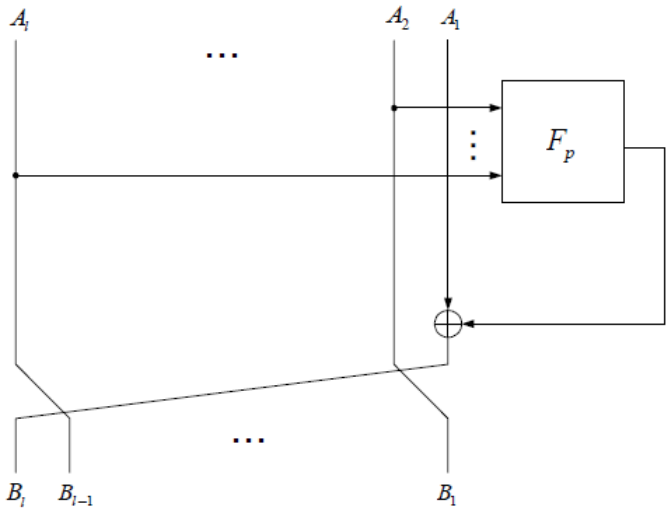



Рис. 5 Модифицированная схема Фейстеля для хэширования

Перемешивающее преобразование имеет вид

$$h_i = \chi(m_i, h_{i-1}, s_i) = \psi^{61}\left(h_{i-1} \oplus \psi\left(m_i \oplus \psi^{12}(s_i)\right)\right),$$

$\psi^j - j$ -я степень преобразования $\psi$ . Схематически данное преобразование представлено на Рис. 6.

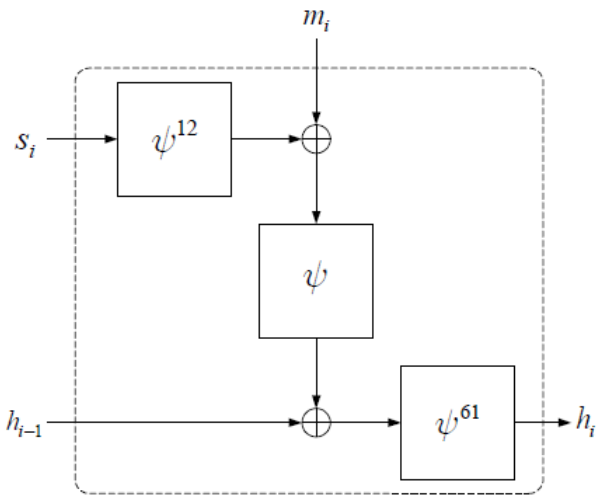



**Рис. 6 Перемешивающее преобразование ГОСТ Р 34.11 – 94**

Заметим (см. Рис. 6), что $s_i$ выводится из $h_{i-1}$ (см. Рис. 4). Функция $\psi : \{01\}^{256} \to \{01\}^{256}$ преобразует слово $\eta_{16} \| \cdots \| \eta_1, \quad \eta_i \in \{01\}^{16}$, $\mathrm{i} = \overline{1,16}$ в слово $\eta_1 \oplus \eta_2 \oplus \eta_3 \oplus \eta_4 \oplus \eta_{13} \oplus \eta_{16} \| \eta_{16} \| \eta_{15} \| \cdots \| \eta_2$ (см. Рис. 7).

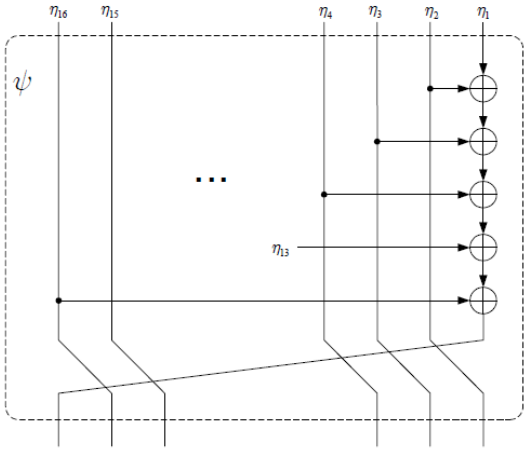



**Рис. 7 Функция $\psi$ перемешивающего преобразования**

***Замечание 3*** Функция $\psi$ перемешивающего преобразования не удовлетворяетследующим локальным критериям:

- 0-1 сбалансированности (гарантирует, что выходы функции будут 0 или 1 с вероятностью 0,5 когда вход функции выбирается случайно и равновероятно над всеми возможными векторами);
- большой нелинейности: Функция $f$ называется *линейной функцией*, если $f$ имеет вид $f(x_n, x_{n-1}, \ldots, x_1) = a_n x_n \oplus a_{n-1} x_{n-1} \oplus \ldots \oplus a_1 x_1 \oplus a,$ где $(2)$ $a_i \in GF(2)$; Если обратить внимание на вид функции $\psi$ , то очевидно, что это линейная функция;
- строгому лавинному критерию (Strict Avalanche Criterion, SAC) говорят, что $f$ удовлетворяет строгому лавинному критерию, если для каждого $1 \le i \le n$, дополнение $x_i$ даст в результате выход $f$ являющийся дополнением для 50% всех возможных входных векторов.

Кроме этого в перемешивающем преобразовании применяется только одна функция $\psi$ , а следовательно оно не обладает следующими общими свойствами:

- линейной неэквивалентности структуры (две функции $f$ и $g$ *линейно эквивалентны по структуре*, если $f$ можно преобразовать в $g$ линейным преобразованием координат и дополнениями функций, т.е имеется неособенная $n \times n$ матрица $A$ над $GF(2)$ а также вектор $B \in V_n$ такой что $f(xA \oplus B) = g(x)$, или $f(xA \oplus B) \oplus 1 = g(x)$, где $\mathrm{x} = (x_n, x_{n-1}, \ldots, x_1)$; этот критерий гарантирует, что функции, используемые криптографическим алгоритмом, не обладают схожестью структуры;
- взаимной некоррелированности выходов (функции $f$ и *взаимно некоррелированные по выходу*, если $f$ , $g$ и $f \oplus g$ - 0-1 сбалансированные нелинейные функции; гарантирует, что последовательности функций не будут взаимно коррелированы или через линейные функции, или через смещение в выходных битах).

***б) Подпрограмма генерирования ключей***

Данная подпрограмма использует следующие функции и константы:

Константы

$C_2, C_4 = \{0\}^{256}$, $C_3 0\times f0ff000ff00f0ff00f0f0f0ff0f0f0f0$.
Функции:

$P:\{01\}^{256} \to \{01\}^{256}$. Пусть $X = \xi_{32} \| \xi_{31} \| \cdots \| \xi_1$, тогда $\mathrm{P}(X) = \xi_{\varphi(32)} \| \xi_{\varphi(31)} \| \cdots \| \xi_{\varphi(1)}$, где $\varphi(i+1+4(k-1)) = 8i+k$, $\mathrm{i} = \overline{0,3}$, $k = \overline{0,8}$.

$\mathrm{A}:\{01\}^{256} \to \{01\}^{256}$. Пусть $X = x_4 \| x_3 \| x_2 \| x_1$, тогда $A(X) = (x_1 \oplus x_2) \| x_4 \| x_3 \| x_2$.

**Алгоритм 1** *Подпрограмма генерирования ключей:*

1. $U \leftarrow m_i, V \leftarrow h_{i-1}, W \leftarrow U \oplus V, K_1 \leftarrow P(W)$
2. Для $j$ от 2 от 4 выполняем следующее:
   $U \leftarrow A(U) \oplus C_j, V \leftarrow A(A(V)), W \leftarrow U \oplus V, K_j \leftarrow P(W)$.
3. Выход: $K_1, K_2, K_3, K_4$,

***в) Шифрующее преобразование***

Основным функциональным предназначением является получение $s_i$ из $h_{i-1}$. Пусть $h_{i-1} = h_{i-1}^4 \| h_{i-1}^3 \| h_{i-1}^2 \| h_{i-1}^1$, где $h_{i-1}^j \in \{01\}^{64}$, $j = \overline{1,4}$, а $s_i = s_i^4 \| s_i^3 \| s_i^2 \| s_i^1$, где $s_i^j \in \{01\}^{64}$, $j = \overline{1,4}$. Тогда $s_i^j \leftarrow E_{K_j}\left(h_{i-1}^j\right)$, где $j = \overline{1,4}$, а $E_{K_j}$ - шифрование по ГОСТ28147 – 89 в режиме простой замены.
Схематически это изображено на Рис. 8.

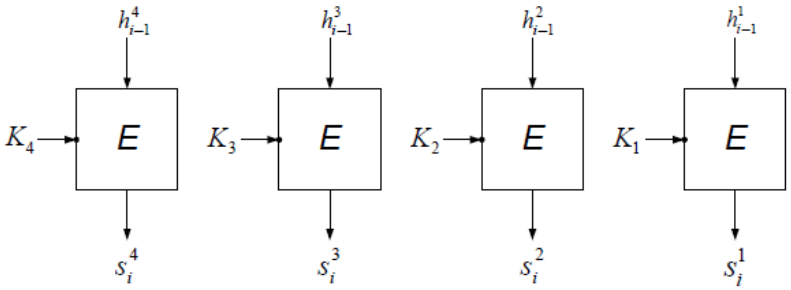



**Рис. 8 Шифрующее преобразование**

***Замечание 4*** Как видно $s_i = s_i^4 \| s_i^3 \| s_i^2 \| s_i^1$ - конкатенация 64 битных слов, полученных в результате шифрования. Следовательно, в свете замечания 3, $s_i$ можно было бы атаковать по частям, но это предотвращается казуальным генерированием ключей (в данном случае $m_i$ и $h_{i-1}$- выступают в роли ключа для выведения ключей).
***Замечание 5*** (S-блоки при шифровании в режиме простой замены) Согласно стандарту, ГОСТ 28147 – 89 S-блоки не определены, но указано, что они должны назначаться вышестоящим органом. Следовательно, они должны учитываться как данные инициализации функции хэширования, и в определенной степени влияют на ее свойства.

*2.* ***Функция сжатия финальной итерации***

Функция сжатия финальной итерации производит итоговую хэш величину, зависящую от результата хэширования последовательным методом, контрольной суммы по mod(2) и длины сообщения, в битах приведенной по mod($2^{256}$) $m_l' \times h_{l-1} \times \Sigma_l \times L_l \to h_{итог}$, где $m_l', h_{l-1}, \Sigma_l . L_l \in \{01\}^{256}$ - последний набитый нулями блок. Сначала вычисляется битовая

длина последнего (не набитого) блока сообщения $|m_l| \le 256$ бит. Если $|m_l| < 256$, то производится его набивка справа нулями до достижения длины 256 бит и получается новый блок $m_l^{'} \leftarrow \{0\}^{256-|m_l|} \| m_l$ . Вычисляются итоговая контрольная сумма $\Sigma_l \leftarrow \Sigma_{l-1} \otimes m_l^{'}$ и длина всего сообщения $L_l \leftarrow L_{l-1} + |m^{'}| \left(\mathrm{mod}\, 2^{256}\right)$. Параллельно выполняется последняя итерация $h_l \leftarrow \chi\left(m^{'}, h_{l-1}\right)$. Затем вычисляются перемешивающие преобразования $h_{l+1} \leftarrow \chi\left(L_l, h_{l-1}\right)$ и $h_{итог} \leftarrow \chi\left(\Sigma_l, h_{l+1}\right)$ , давая в результате итоговую хэш величину $h_{итог}$ . Теперь дадим формальное описание алгоритма:

***Алгоритм 2*** *Вычисление хэш функции по* ГОСТ Р 34.11 – 94
***Вход***: двоичное сообщение $M$ , блоки замен для шифрования в режиме простой замены по ГОСТ 28147-89, начальный вектор $IV \in \{01\}^{256}$
***Выход***: хэш величина $h_{итог}$ для сообщения $M$ .
1. ***Инициализация алгоритма:*** $h \leftarrow IV, \Sigma \leftarrow \{0\}^{256}, L \leftarrow \{0\}^{256}$.
2.***Функция сжатия внутренних итераций:*** Пока $|M| > 256$ выполняем следующее:
2.1. $h \leftarrow \chi\left(m_s, h\right)$ (итерация метода последовательного хэширования),
2.2. $L \leftarrow L + 256\left(\mathrm{mod}\, 2^{256}\right)$ (итерация вычисления длины сообщения),
2.3. $\Sigma \leftarrow \Sigma \otimes m_s$ (итерация вычисления контрольной суммы).
3.Иначе (***функция сжатия финальной итерации***)
3.1. $L \leftarrow L + |m| \left(\mathrm{mod}\, 2^{256}\right)$(вычисление полной длины сообщения),
3.2. $m^{'} \leftarrow \{0\}^{256-|m|} \| m$ (набивка последнего блока),
3.3. $\Sigma \leftarrow \Sigma \otimes m^{'}$ (вычисление контрольной суммы сообщения),
3.4. $h \leftarrow \chi\left(m^{'}, h\right)$,
3.5. $h \leftarrow \chi\left(L, h\right)$ (MD - усиление),
3.6. $h_{итог} \leftarrow \chi\left(\Sigma, h\right)$.
4.***Выход*** ( $h_{итог}$ )

***Заключение*** Из приведенного выше описания хэш функции по ГОСТ Р 34.11 – 94 можно сделать следующие выводы:
 Булева функция перемешивающего преобразования линейна и ее применение неоправданно – это приводит к снижению скорости обработки данных из-за большого числа повторений перемешивающего преобразования для достижения “заданного” уровня безопасности (в первую очередь SAC);
 Шифрующее преобразование, при определенных допущениях, невозможно атаковать по частям, а, следовательно, функцию сжатия можно считать стойкой к столкновениям (CRHF);
 Алгоритм хэширования является методом последовательного хэширования с MD – усилением (коэффициент сжатия 2);
 Используется постериорное форматирование сообщения;
 Стойкость хэш функции в известной мере зависит от выбора блоков замен в шифрующем преобразовании, к тому же они один из параметров инициализации алгоритма хэширования;

 *IV* в стандарте не фиксирован, а это подразумевает, что необходимо выработать правила его использования (см. Замечание 2), к тому же имеется большой класс атак на псевдостолкновения при не фиксированном *IV* [3];
 скорость обработки данных хэш функцией значительно меньше чем у аналогичных по внешним параметрам HAVAL, SHA-256, а тем более остального MD-семейства, из-за попыток ликвидировать очевидные оплошности конструирования усложнением функции сжатия;
 приблизительная скорость реализации 4/5 от скорости реализации лежащего в основе алгоритма шифрования;
 зависимость получаемой хэш величины от длины исходного сообщения исключает возможность проведения атаки на длинные сообщения для второго прообраза;
 нет обоснования выбора конструкции, функций, констант.

Литература:

***ГОСТ Р 34.11 – 94.*** Функция хэширования.Краткий анализ.Шефановский Д.Б.,Май 2001, shefanovski@infosec.ru

1. ГОСТ Р 34.11 – 94. "Информационная технология. Криптографическая защита информации. Функция хэширования".
2. *M. Bellare, R. Canetti, and H. Krawczyk.* **Keying hash functions for message authentication.** *Advances in Cryptology - Crypto 96 Proceedings, Lecture Notes in Computer Science Vol. 1109, N. Koblitz ed,* Springer-Verlag, 1996.
3. *Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone* **Handbook of Applied Cryptography** *CRC Press ISBN: 0-8493-8523-7 October 1997, 816 pages*

ПРИЛОЖЕНИЕ 2

**Лабораторная работа № 1: Простые шифры**

Простые шифры. Разработка и анализ простых криптографических алгоритмов на основе методов перестановок и подстановок.

Рекомендации:

Ещё несколько веков назад само применение письменности можно было рассматривать как способ закрытия информации, т.к. владение письменностью было уделом немногих.

Один из самых древних шифротекстов, написанных клинописью, был обнаружен в Месопотамии и датируется XX веком до н.э. Известны также древнеегипетские религиозные и медицинские шифротексты.

В середине IX века до н.э. использовалось шифрующие устройство - скиталь для получения шифра перестановки. При шифровании слова писались на узкую ленту, намотанную на цилиндр, вдоль образующей этого цилиндра (скиталя). После этого лента разматывалась и на ней оставались переставленные буквы исходного текста. Ключом в этом случае являлся диаметр этого цилиндра.

В начале нашей эры создается шифр замены. Так в 56 году н.э. во время войны с галлами Юлий Цезарь использует шифр простой замены, строящийся следующим образом:

Под алфавитом открытого текста пишется тот же атфавит со сдвигом на три позиции по циклу. При шифровании буквы открытого текста у верхнего алфавита заменялись буквами нижнего алфавита.

А Б В Г Д Е Ж ... Э Ю Я Г Д Е Ж ... Э Ю Я А Б В Более сложной разновидностью шифра замены является «квадрат Полибия».

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | **A** | **в** | **C** | **D** | **E** |
| 2 | **F** | **G** | **н** | **J** | **К** |
| 3 | **L** | **M** | **N** | **O** | **P** |
| 4 | *Q* | *R* | *S* | *T* | **И** |
| 5 | *V* | *W* | *X* | *Y* | *Z* |

При шифровании буквы открытого текста заменяются парой чисел: номер столбца и номер строки соответствующей буквы в таблице

Открытый текст: R E A D Y
Шифротекст: 24 51 11 41 45

**Задание:**

Пусть

X=**ШИФР_СКРЫВАЕТ_СОДЕРЖАНИЕ_ТЕКСТА,**

- Открытый текст, и к нему применяется следующий ключ (см.табл.) замены один раз, затем к полученному слову еще раз и т.д. Через сколько замен впервые появится исходный открытый текст?

| А | Б | В | Г | Д | Е | Ж | З | И | Й | К | Л | М | Н | О | П | Р | С | Т | У | Ф |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Т | У | Ф | Х | Ц | Ч | Ш | Щ | Ъ | А | Б | В | Г | Д | Е | Ж | З | И | Й | К | Л |

| Х | Ц | Ч | Ш | Щ | Ъ | Ы | Ь | Э | Ю | Я | _ |
|---|---|---|---|---|---|---|---|---|---|---|---|
| М | Н | О | П | Р | С | Ь | Э | Ю | Я | _ | Ы |

**Лабораторная работа № 2: Метод гаммирования.**

Генерация псевдослучайных последовательностей чисел в системах защиты информации. Оценка статистических характеристик датчика псевдослучайных чисел с заданным законом распределения

***Рекомендации:***

Рассмотрим алгоритм конгруэнтного линейного датчика псевдослучайных чисел, который определяет последовательность порождающих чисел: $S_1$, $S_2$…$S_L$.

$S_1=(AS_0+C)(modM)$

$S_2=(AS_1+C)(modM)$

…

$S_L=(AS_{L-1}+C)(modM)$

Где A и C – константы, $S_0$- исходная начальная величина.( оптимальными значениями A и C признаны: Amod 4 = 1 и C — нечетное.), M- выбирается в зависимости от требуемой длины гаммы и равно $2^{\wedge b}$, где b — длина (в битах) гаммы и шифруемого слова компьютера соответственно, рекомендуемое значение 256.

**Задание:**

Зашифровать открытый текст T методом гаммирования, наложить гамму к шифру применяя логическую операцию «исключающее ИЛИ», получаемую с помощью

генератора случайных чисел, определить длину гаммы L.

**T= ГАММИРОВАНИЕ ЭТО НАЛОЖЕНИЕ ПО ОПРЕДЕЛЕННОМУ ПРАВИЛУ ГАММЫ ШИФРА НА ОТКРЫТЫЕ ДАННЫЕ.**

**Лабораторная работа № 3: Симметричная криптография.**

Разработка и реализация варианта симметричного криптографического алгоритма

***Рекомендации:***

- Длина блока – 64 бит.
- Длина ключа 56 бит.
- Для шифрования и дешифрования используются одинаковые алгоритм и ключ.
- Состоит из 16 циклов.

Схема работы: Открытый текст ->Начальная перестановка ->16 циклов -> Обращение начальной перестановки->Шифртекст.

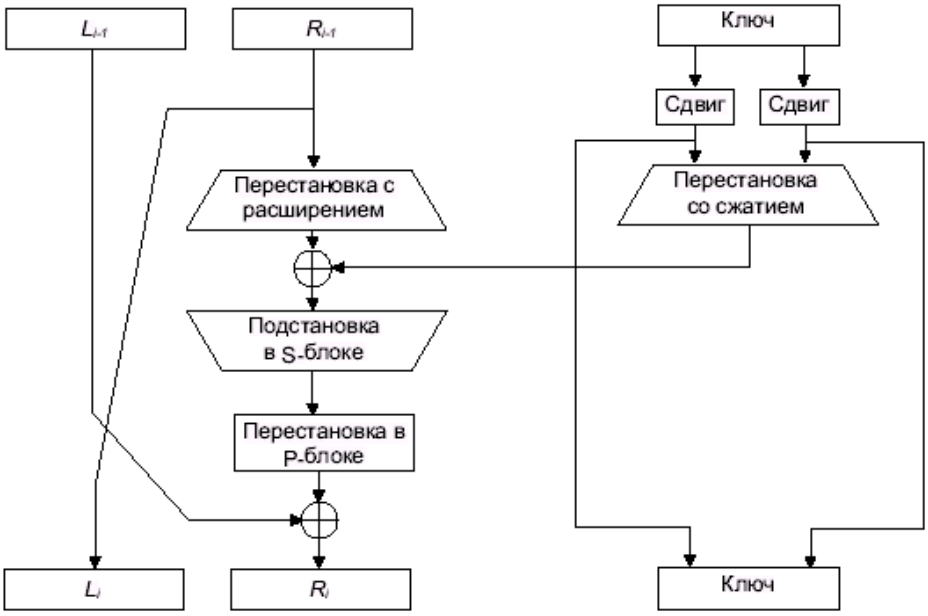
Начальная и конечная перестановки не влияют на криптостойкость алгоритма.

Генерация ключа: для каждого из 16 циклов из 56 битного ключа генерируется 48 битный подключ. Для этого сначала 56 битный подключ делится на 2 28-битных половины. Затем половины циклически сдвигаются на 1 или 2 бита в зависимости от номера блока. После этого происходит перестановка со сжатием, из 56-48 бит (не зависит от номера цикла).

На каждом цикле 32 битный блок преобразуется в 48 битный (перестановка с расширением)

Смысл операции: из-за влияния одного бита на 2 подстановки быстрее возрастает зависимость битов результата от битов исходных данных. Называется также E блок (expansion).

После этого выполняется операция XOR с расширенным ключом . Затем выполняется операция подстановки (Применение S box). Используется всего 8 S box с 6 битовыми входами, каждая из них дает на выходе 4 бита. Подстановка является ключевым этапом DES, обеспечивая нелинейность алгоритма. Затем происходит перестановка с помощью P блоков.




Нестоек к линейному и дифференциальному криптоанализу. Силовая атака – не более 40 дней стойкости.

Улучшенный вариант - шифрование на одном ключе, дешифрование на другом, шифрование опять на первом. Но размер блока также 64 бита, поэтому стойкость недостаточна.

Тройной Des – тройное DES шифрование.

**Задание:** разработать алгоритм DES

### Лабораторная работа № 4: Криптоанализ

Проведение сравнительного анализа систем ГОСТ 28147-89 и DES

## Лабораторная работа № 5 Асимметричные криптографические системы.

Программная реализация алгоритма RSA
***Рекомендации:***

RSA – криптографическая система открытого ключа, обеспечивающая такие механизмы защиты как шифрование и цифровая подпись (аутентификация – установление подлинности). Криптосистема RSA разработана в 1977 году и названа в честь ее разработчиков Ronald Rivest, Adi Shamir и Leonard Adleman.

*Математические основы*. RSA

*Теорема 1. (Малая теорема Ферма.)*

Если p - простое число, то

$x^{p-1} = 1 \pmod{p}$ (1)

для любого *x*, простого относительно p, и

$x^{p} = x \pmod{p}$ (2)

для любого *x*.

*Определение. Функцией Эйлера* φ(n) называется число положительных целых, меньших *n* и простых относительно *n*.

| n | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| φ(n) | 1 | 2 | 2 | 3 | 2 | 6 | 4 | 6 | 4 | 10 | 4 |

*Теорема 2.* Если *n=pq*, (*p* и *q* - отличные друг от друга простые числа), то

$\varphi(n)=(p-1)(q-1).$

*Теорема 3.* Если *n=pq*, (*p* и *q* - отличные друг от друга простые числа) и x - простое относительно p и *q*, то

$x^{\varphi(n)} = 1 \pmod{n}.$

*Следствие .* Если *n=pq*, (*p* и *q* - отличные друг от друга простые числа) и e простое относительно φ(n), то отображение

$E_{e,n}: x \rightarrow x^{e} \pmod{n}$

является взаимно однозначным на $\mathbf{Z}_n$.

Если e - простое относительно φ(n), то существует целое d, такое, что

$ed = 1 \pmod{\varphi(n)}$ (3)

*Алгоритм:*

Алгоритм RSA работает следующим образом:

- берутся два достаточно больших простых числа p и q и вычисляется их произведение n = p*q; n называется модулем.

- Затем выбирается число e, удовлетворяющее условию

1< e < (p - 1)*(q - 1) и не имеющее общих делителей кроме 1 (взаимно простое) с числом (p - 1)*(q - 1).

- Затем вычисляется число d таким образом, что (e*d - 1) делится на (p - 1)*(q – 1).

e – открытый (public) показатель

d – секретный (private) показатель.

(n; e) – открытый (public) ключ

(n; d). – секретный (private) ключ.

Делители (факторы) p и q можно либо уничтожить либо сохранить вместе с секретным (private) ключом.

Если бы существовали эффективные методы разложения на сомножители (факторинга), то, разложив n на сомножители (факторы) p и q, можно было бы получить секретный (private) ключ d. Таким образом надежность криптосистемы RSA основана на трудноразрешимой – практически неразрешимой – задаче разложения n на сомножители (то есть на невозможности факторинга n) так как в настоящее время эффективного способа поиска сомножителей не существует.

## Лабораторная работа № 6 Алгоритм обмена ключами

Разработка и программная реализация протокола обмена симметричными ключами на

основе алгоритма Diffyie-Hellman

*Рекомендации:*

Исторически первый и наиболее известный протокол распределения ключей был предложен Диффи и Хеллманом в 1976 г. [DH]. Пусть обоим участникам A и B известны некоторое большое простое число $p$ и некоторый порождающий $g$ группы Z* (хорошо известно, что мультипликативная группа произвольного конечного поля циклическая).

Параметры **pug** могут быть выбраны, например, центром доверия. В роли параметра безопасности здесь выступает длина **p,** а в роли пространства ключей - группа $Z^*_p$ .

Протокол заключается в следующем:

A выбирает $x_a \in \{0,1,...,p-2\}$, вычисляет $y_A = g^{x_a} \bmod p$ и посылает его B, сохраняя $x^a$ в секрете. B выбирает $x_b \in \{0,1,...,p-2\}$, вычисляет $y_B = g^{x_b} \bmod p$ и посылает его B, сохраняя $x^b$ в секрете. A вычисляет $\kappa_b = y_a^{x_b} \bmod p$ , B - $\kappa_a = y_b^{x_a} \bmod p$

Ясно, что $к_а = к_ь$, является искомым общим секретным ключом.

Этот метод не позволяет противнику активно вмешиваться в выполнение протокола.

## Лабораторная работа № 7 Цифровая электронная подпись

Программная реализация алгоритмов цифровой подписи
***Рекомендации:***

***Процедура обмена сообщением в криптосистеме RSA***

Предположим, что пользователь А хочет передать пользователю В сообщение в зашифрованном виде, используя криптосистему RSA [3]. В таком случае пользователь А выступает в роли отправителя сообщения, а пользователь В — в роли получателя. Как отмечалось выше, ключи RSA должен сформировать получатель сообщения, т.е. пользователь В. Рассмотрим последовательность действий пользователя В и пользователя А.

1) Пользователь В выбирает два произвольных больших простых числа *P* и *Q*.
2) Пользователь В вычисляет значение *N = PQ*.
3) Пользователь В вычисляет функцию Эйлера *φ{N) = (P-1)(Q*-1) и выбирает случайным образом значение открытого ключа.
4) Пользователь В вычисляет значение секретного ключа $к_в$, используя расширенный алгоритм Евклида.
5) Пользователь В пересылает пользователю А пару чисел (А, $к_в$) по незащищенному каналу.
6) Пользователь А разбивает исходный открытый текст *M* на блоки M,, каждый из которых может быть представлен в виде числа $M \in \{0, 1, ... A-1\}$.
7) Пользователь А шифрует текст, представленный в виде последовательности чисел *M,*

   по формуле $C_i = M_i^{K_B} (\mathrm{mod}\, N)$ и отправляет криптограмму *C = (C1, C2, ...)* пользователю *B.*

8) Пользователь В расшифровывает принятую криптограмму С, используя секретный ключ $к_в$, по формуле $M_i = C_i^{k_B} (\mathrm{mod}\, N)$.

В результате будет получена последовательность чисел M,, которые представляют собой исходное сообщение *M.* Чтобы алгоритм RSA имел практическую ценность, необходимо иметь возможность без существенных затрат генерировать большие простые числа, уметь оперативно вычислять значения ключей $К_в$ *а* $к_в$.

Приложение 3

СОГЛАСОВАНО
Протокол заседания кафедры
№_____ от_____

УТВЕРЖДЕНО
Ректор УНПК «МУК»
_____
(название)
_____
(подпись, ф.и.о.)

**ЛИСТ ИЗМЕНЕНИЙ**

в учебно-методический комплекс (модуле) дисциплины _____
(название дисциплины)
по направлению подготовки (специальности)_____

на 20__/20__ учебный год

1. В _____ вносятся следующие изменения:
(элемент УМК)
1.1. ……..;
1.2. ……...;
…
1.9. …… .

2. В _____ вносятся следующие изменения:
(элемент УМК)
2.1. ……..;
2.2. ……...;
…
2.9. …… .

3. В _____ вносятся следующие изменения:
(элемент УМК)
3.1. ……..;
3.2. ……...;
…
3.9. …… .

Составитель подпись расшифровка подписи
дата