Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Информатика»

МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

ПРАКТИКУМ
по выполнению лабораторных работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения

Электронный аналог печатного издания

Гомель 2020

УДК 004.9:005.8(075.8)
ББК 32.972я73
М54

Рекомендовано к изданию научно-методическим советом факультета автоматизированных и информационных систем ГГТУ им. П. О. Сухого (*протокол № 10 от 01.06.2020 г.*)

Составитель *Д. В. Прокопенко*

Рецензент: проф. каф. «Информационные технологии» ГГТУ им. П. О. Сухого д-р техн. наук, проф. *И. А. Мурашко*

М54 **Методы** защиты информации : практикум по выполнению лаборатор. работ для студентов специальности 1-40 04 01 «Информатика и технологии программирования» днев. формы обучения / сост. Д. В. Прокопенко. – Гомель : ГГТУ им. П. О. Сухого, 2020. – 78 с. – Систем. требования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD 16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: https://elib.gstu.by. – Загл. с титул. экрана.

ISBN 978-985-535-454-4.

Изложены основные темы, изучаемые в курсе «Методы защиты информации». Приведены теоретические сведения о криптографии, симметричных и ассиметричных алгоритмах шифрования.

Для студентов специальности 1-40 04 01 «Информатика и технологии программирования» дневной формы обучения.

УДК 004.9:005.8(075.8)
ББК 32.972я73

ISBN 978-985-535-454-4

Оглавление

ВВЕДЕНИЕ

Проблема защиты информации путем ее преобразования, исключающего ее прочтение посторонним лицом, волновала человеческий ум с давних времен. История криптографии – ровесница истории человеческого языка. Более того, первоначально письменность сама по себе была своеобразной криптографической системой, так как в древних обществах ею владели только избранные. Священные книги древнего Египта, древней Индии тому примеры. История человеческой цивилизации стала также историей создания систем безопасной передачи информации. Искусство шифрования и тайной передачи информации было присуще практически всем государствам. Криптография в прошлом использовалась, прежде всего, в военных целях. Однако сейчас, по мере образования информационного общества, криптография становится одним из основных инструментов, обеспечивающих конфиденциальность, доверие, авторизацию, корпоративную безопасность и бесчисленное множество других важных вещей. Практическое применение криптографии стало неотъемлемой частью жизни современного общества – ее используют в таких отраслях как электронная коммерция, электронный документооборот (включая цифровые подписи), телекоммуникации и др. Очень быстро после распространения компьютеров в деловой сфере практическая криптография сделала в своем развитии огромный скачок, причем сразу по нескольким направлениям:

– во-первых, были разработаны стойкие блочные шифры с секретным ключом, предназначенные для решения классической задачи – обеспечения секретности и целостности передаваемых или хранимых данных, они до сих пор остаются «рабочей лошадкой» криптографии, наиболее часто используемыми средствами криптографической защиты;

– во-вторых, были созданы методы решения новых, нетрадиционных задач сферы защиты информации, наиболее известными из которых являются задача подписи цифрового документа и открытого распределения ключей.

Глава 1. КРИПТОГРАФИЧЕСКАЯ ЗАЩИТА ИНФОРМАЦИИ

1.1. Периоды развития и этапы криптографии

История криптографии насчитывает около четырех тысяч лет. В качестве основного критерия периодизации криптографии возможно опираться на технологические характеристики используемых методов шифрования. В данном практикуме будем придерживаться такой периодизации.

Первый период (приблизительно с третьего тысячелетия до н. э.) характеризуется господством моноалфавитных шифров (основной принцип – замена алфавита исходного текста другим алфавитом через замену букв другими буквами или символами).

Второй период (хронологические рамки – с IX в. на Ближнем Востоке (Ал-Кинди) и с XV в. в Европе (Леон Баттиста Альберти) до начала XX в. ознаменовался введением в обиход полиалфавитных шифров.

Третий период (с начала и до середины XX в.) характеризуется внедрением электромеханических устройств в работу шифровальщиков. При этом продолжалось использование полиалфавитных шифров.

Четвертый период – с середины до 70-х гг. XX в. – период перехода к математической криптографии. В работе Клода Шеннона появляются строгие математические определения количества информации, передачи данных, энтропии, функций шифрования. Обязательным этапом создания шифра считается изучение его уязвимости к различным известным атакам – линейному и дифференциальному криптоанализу. Однако до 1975 г. криптография оставалась «классической», или же, более корректно, криптографией с секретным ключом.

Современный период развития криптографии (с конца 1970-х гг. по настоящее время) отличается зарождением и развитием нового направления – криптографии с открытым ключом. Ее появление знаменуется не только новыми техническими возможностями, но и сравнительно широким распространением криптографии для использования частными лицами (в предыдущие эпохи использование криптографии было исключительной прерогативой государства). Правовое регулирование использования криптографии частными лицами в разных странах сильно различается – от разрешения до полного запрета.

Историю криптографии условно можно также разделить на четрые этапа:

– наивная криптография;

– формальная криптография;

– научная криптография;

– компьютерная криптография.

Для наивной криптографии (до начала XVI в.) характерно использование любых (обычно примитивных) способов запутывания противника относительно содержания шифруемых текстов. На начальном этапе для защиты информации использовались методы кодирования и стеганографии, которые родственны, но не тождественны криптографии. Большинство из используемых шифров сводились к перестановке или моноалфавитной подстановке. Одним из первых зафиксированных примеров является шифр Цезаря, состоящий в замене каждой буквы исходного текста на другую, отстоящую от нее в алфавите на определенное число позиций. Другой шифр, полибианский квадрат, авторство которого приписывается греческому писателю Полибию, является общей моноалфавитной подстановкой, которая проводится с помощью случайно заполненной алфавитом квадратной таблицы (для греческого алфавита размер составляет 5 × 5). Каждая буква исходного текста заменяется на букву, стоящую в квадрате снизу от нее.

Этап формальной криптографии (конец XV в. – начало XX в.) связан с появлением формализованных и относительно стойких к ручному криптоанализу шифров. В европейских странах это произошло в эпоху Возрождения, когда развитие науки и торговли вызвало спрос на надежные способы защиты информации. Важная роль на этом этапе принадлежит Леону Батисте Альберти, итальянскому архитектору, который одним из первых предложил многоалфавитную подстановку. Данный шифр, получивший имя дипломата XVI в. Блеза Вижинера, состоял в последовательном «сложении» букв исходного текста с ключом (процедуру можно облегчить с помощью специальной таблицы). Его работа «Трактат о шифре» считается первой научной работой по криптологии. Одной из первых печатных работ, в которой обобщены и сформулированы известные на тот момент алгоритмы шифрования, является труд «Полиграфия» немецкого аббата Иоганна Трисемуса. Ему принадлежат два небольших, но важных открытия: способ заполнения полибианского квадрата (первые позиции заполняются с помощью легко запоминаемого ключевого слова, ос-

тальные – оставшимися буквами алфавита) и шифрование пар букв (биграмм). Простым, но стойким способом многоалфавитной замены (подстановки биграмм) является шифр Плейфера, который был открыт в начале XIX в. Чарльзом Уитстоном. Уитстону принадлежит и важное усовершенствование – шифрование ««двойным квадратом». Шифры Плейфера и Уитстона использовались вплоть до Первой мировой войны, так как с трудом поддавались ручному криптоанализу. В XIX в. голландец Керкхофф сформулировал главное требование к криптографическим системам, которое остается актуальным и поныне: секретность шифров должна быть основана на секретности ключа, но не алгоритма. Появление роторных криптосистем обеспечило еще более высокую криптостойкость, а также позволило автоматизировать (в смысле механизировать) процесс шифрования. Одной из первых подобных систем стала изобретенная в 1790 г. Томасом Джефферсоном, будущим президентом США, механическая машина. Многоалфавитная подстановка с помощью роторной машины реализуется вариацией взаимного положения вращающихся роторов, каждый из которых осуществляет «прошитую» в нем подстановку. Практическое распространение роторные машины получили только в начале XX в. Одной из первых практически используемых машин стала немецкая Enigma, разработанная в 1917 г. Эдвардом Хеберном и усовершенствованная Артуром Кирхом. Роторные машины активно использовались во время Второй мировой войны. Помимо немецкой машины Enigma использовались также устройства Sigaba (США), Typex (Великобритания), Red, Orange и Purple2 (Япония). Роторные системы – вершина формальной криптографии, так как относительно просто реализовывали очень стойкие шифры. Успешные криптоатаки на роторные системы стали возможны только с появлением ЭВМ в начале 40-х гг.

Главная отличительная черта научной криптографии (30–60-е гг. XX в.) – появление криптосистем со строгим математическим обоснованием криптостойкости. К началу 30-х гг. окончательно сформировались разделы математики, являющиеся научной основой криптологии: теория вероятностей и математическая статистика, общая алгебра, теория чисел, начали активно развиваться теория алгоритмов, теория информации, кибернетика. Своеобразным водоразделом стала работа Клода Шеннона «Теория связи в секретных системах», где сформулированы теоретические принципы криптографической защиты информации. Шеннон ввел понятия «рассеивание» и «перемешивание», обосновал возможность создания сколь угодно стойких крип-

тосистем. В 60-х гг. ведущие криптографические школы подошли к созданию блочных шифров, еще более стойких по сравнению с роторными криптосистемами, однако допускающих практическую реализацию только в виде цифровых электронных устройств. Компьютерная криптография (с 70-х гг. XX в.) обязана своим появлением вычислительным средствам с производительностью, достаточной для реализации криптосистем. Первым классом криптосистем, практическое применение которых стало возможно с появлением мощных и компактных вычислительных средств, стали блочные шифры. В 70-е гг. был разработан американский стандарт шифрования DES (принят в 1978 г.). Один из его авторов, Хорст Фейстел (сотрудник IBM), описал модель блочных шифров, на основе которой были построены другие, более стойкие симметричные криптосистемы, в том числе отечественный стандарт шифрования ГОСТ 28147–89. С появлением DES обогатился и криптоанализ, для атак на американский алгоритм было создано несколько новых видов криптоанализа (линейный, дифференциальный и т. д.), практическая реализация которых опять же была возможна только с появлением мощных вычислительных систем. В середине 70-х гг. произошел настоящий прорыв в современной криптографии – появление асимметричных криптосистем, которые не требовали передачи секретного ключа между сторонами. Здесь отправной точкой принято считать работу, опубликованную Уитфилдом Диффи и Мартином Хеллманом в 1976 г. под названием «Новые направления в современной криптографии». В ней впервые сформулированы принципы обмена шифрованной информацией без обмена секретным ключом. Независимо к идее асимметричных криптосистем подошел Ральф Меркли. Несколькими годами позже Рон Ривест, Ади Шамир и Леонард Адлеман открыли систему RSA, первую практическую асимметричную криптосистему, стойкость которой была основана на проблеме факторизации больших простых чисел. Асимметричная криптография открыла сразу несколько новых прикладных направлений, в частности системы электронной цифровой подписи и электронных денег. В 80–90-е гг. появились совершенно новые направления криптографии: вероятностное шифрование, квантовая криптография и др. Осознание их практической ценности еще впереди. Актуальной остается и задача совершенствования симметричных криптосистем. В 80–90-х гг. были разработаны нефейстеловские шифры (SAFER, RC6 и др.), а в 2000 г. после открытого международ-

ного конкурса был принят новый национальный стандарт шифрования США – AES.

1.2. Криптография в древние времена

История криптографии насчитывает не одно тысячелетие. Уже в исторических документах древних цивилизаций – Индии, Египте, Китае, Месопотамии – имеются сведения о системах и способах составления шифрованного письма. Видимо, первые системы шифрования появились одновременно с письменностью в четвертом тысячелетии до нашей эры.

В древнеиндийских рукописях приводится более шестидесяти способов письма, среди которых есть и такие, которые можно рассматривать как криптографические. Имеется описание системы замены гласных букв согласными, и наоборот. Один из сохранившихся шифрованных текстов Месопотамии представляет собой табличку, написанную клинописью и содержащую рецепт изготовления глазури для гончарных изделий. В этом тексте использовались редко употребляемые значки, игнорировались некоторые буквы, употреблялись цифры вместо имен. В рукописях Древнего Египта шифровались религиозные тексты и медицинские рецепты. Шифрование использовалось в Библии. Некоторые фрагменты библейских текстов зашифрованы с помощью шифра, который называется атбаш. Правило зашифрования состояло в замене i-й буквы алфавита ($n - i + 1$), где n – число букв в алфавите. Происхождение слова атбаш объясняется принципом замены букв. Это слово составлено из букв Алеф, Тае, Бет и Шин, т. е. первой и последней, второй и предпоследней букв древнесемитского алфавита.

В Древней Греции криптография уже широко использовалась в разных областях деятельности, в особенности в государственной сфере. Плутарх сообщает, что жрецы, например, хранили в форме тайнописи свои прорицания. В Спарте в 5–6 вв. до н. э. использовалось одно из первых шифровальных приспособлений – Сцитала. Это был жезл цилиндрической формы, на который наматывалась лента из пергамента. Кроме жезла могли использоваться рукоятки мечей, кинжалов, копий и т. д. Вдоль оси цилиндра на пергамент построчно записывался текст, предназначенный для передачи. После записи текста лента сматывалась с жезла и передавалась адресату, который имел точно такую же Сциталу. Ясно, что такой способ шифрования осуще-

ствлял перестановку букв сообщения. Ключом шифра служит диаметр Сциталы. Известен также и метод вскрытия такого шифра, приписываемый Аристотелю. Предлагалось заточить на конус длинный брус и, обернув в него ленту, начать сдвигать ее по конусу от малого диаметра до самого большого. В том месте, где диаметр конуса совпадал с диаметром Сциталы, буквы текста сочетались в слоги и слова. После этого оставалось лишь изготовить цилиндр нужного диаметра.

Греческий писатель Полибий использовал систему сигнализации, которая была широко принята как метод шифрования. Он записывал буквы алфавита в квадратную таблицу и заменял их координатами: парами чисел (i, j), где i – номер строки, j – номер столбца. Применительно к латинскому алфавиту квадрат Полибия имеет вид, представленный на рис. 1.1.

	1	2	3	4	5
1	A	B	C	D	E
2	F	G	H	I/J	K
3	L	M	N	O	P
4	Q	R	S	T	U
5	V	W	X	Y	Z

Рис. 1.1. Квадрат Полибия

Пары (i, j) передавались с помощью факелов. Например, для передачи буквы О нужно было взять три факела в правую руку и четыре – в левую.

Подобные шифровальные приспособления с небольшими изменениями просуществовали до эпохи военных походов Юлия Цезаря. Положение меняется в эпоху расцвета Рима, который первоначально представлял собой лишь небольшую гражданскую общину, со временем он разросся, подчинив себе Италию, а затем все Средиземноморье. Чтобы управлять наместниками в многочисленных провинциях, шифрованная связь для римских органов власти стала жизненно необходимой. Особую роль в сохранении тайны сыграл способ шифрования, предложенный Юлием Цезарем [2]. Гай Светоний писал: «...существуют и его письма к Цицерону и письма к близким о до-

машних делах: в них, если нужно было сообщить что-то негласно, он пользовался тайнописью, то есть менял буквы так, чтобы из них не складывалось ни одного слова. Чтобы разобрать и прочитать их, нужно читать всякий раз четвертую букву вместо первой, например, D вместо A и так далее». Таким образом, Цезарь заменял бук-вы в соответствии с подстановкой, нижняя строка которой представляет собой алфавит открытого текста, сдвинутый циклически на три буквы влево.

1.3. Криптография от средних веков до нового времени

Еще один значительный шаг вперед криптография сделала благодаря труду Леона Альберти. Известный философ, живописец, архитектор, он в 1466 г. написал труд о шифрах. В этой работе был предложен шифр, основанный на использовании шифровального диска. Сам Альберти называл его шифром, «достойным королей».

Шифровальный диск представлял собой пару соосных дисков разного диаметра. Больший из них – неподвижный, его окружность разделена на 24 равных сектора, в которые вписаны 20 букв латинского алфавита в их естественном порядке и 4 цифры (от 1 до 4). При этом из 24-буквенного алфавита были удалены 4 буквы, без которых можно обойтись, подобно тому, как в русском языке обходятся без Ъ, Е, Й. Меньший диск – подвижный, по его окружности, разбитой также на 24 сектора, были вписаны все буквы смешанного латинского алфавита (рис. 1.2).

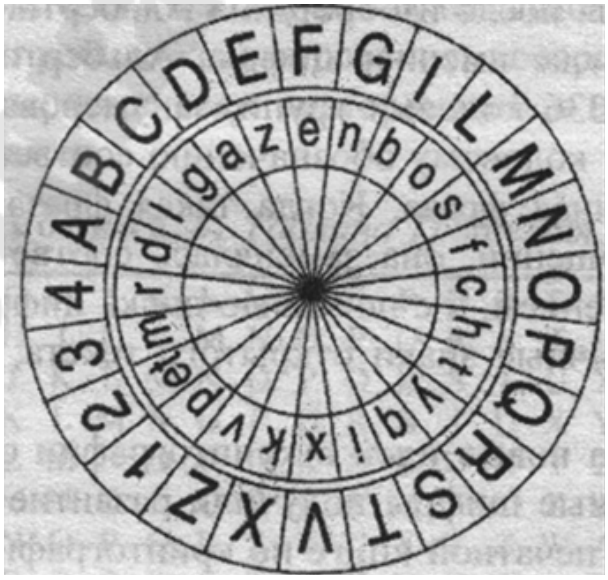

Рис. 1.2. Диск Альберти

Имея два таких прибора, корреспонденты догадывались о первой индексной букве на подвижном диске. При шифровании сообщения отправитель ставил индексную букву против любой буквы большого диска. Он информировал корреспондента о таком положении диска, записывая эту букву внешнего диска в качестве первой буквы шифротекста. Очередная буква открытого текста отыскивалась на неподвижном диске, и стоящая против нее буква меньшего диска являлась результатом ее зашифрования. После того как были зашифрованы несколько букв текста, положение индексной буквы изменялось, о чем также сообщалось корреспонденту.

Такой шифр имел две особенности, которые делают изобретение Альберти событием в истории криптографии. Во-первых, в отличие от шифров простой замены шифровальный диск использовал не один, а несколько алфавитов для зашифрования. Такие шифры получили название многоалфавитных. Во-вторых, шифровальный диск позволял использовать так называемые коды с перешифрованием, которые начали широко распространяться лишь в конце XIX в., т. е. спустя четыре столетия после изобретения Альберти. Для этой цели на внешнем диске имелись цифры. Альберти составил код из 336 кодовых групп, занумерованных от 11 до 4444. Каждому кодовому обозначению соответствовала некоторая законченная фраза. Когда такая фраза встречалась в открытом сообщении, она заменялась соответствующим кодовым обозначением, а с помощью диска цифры зашифровывались как обычные знаки открытого текста, превращаясь в буквы.

Богатым на новые идеи в криптографии оказался XVI век. Многоалфавитные шифры получили развитие в вышедшей в 1518 г. первой печатной книге по криптографии под названием «Полиграфия». Автором книги был один из самых знаменитых ученых того времени аббат Иоганнес Тритемий. В этой книге впервые в криптографии появляется квадратная таблица. Шифр-алфавиты записаны в строки таблицы один под другим, причем каждый из них сдвинут на одну позицию влево по сравнению с предыдущим.

Важное усовершенствование многоалфавитных систем, состоящее в идее использования в качестве ключа текста самого сообщения или же шифрованного текста, принадлежит Джероламо Кардано и Блезу де Виженеру. Такой шифр был назван самоключом. В книге Виженера «Трактат о шифрах» самоключ представлен следующим образом. В простейшем случае за основу бралась таблица Тритемия с добавленными к ней в качестве первой строки и первого столбца

алфавитами в их естественном порядке. Позже такая таблица стала называться таблицей Виженера. В общем случае таблица Виженера состоит из циклически сдвигаемых алфавитов, причем первая строка может быть произвольным смешанным алфавитом (рис. 1.3).

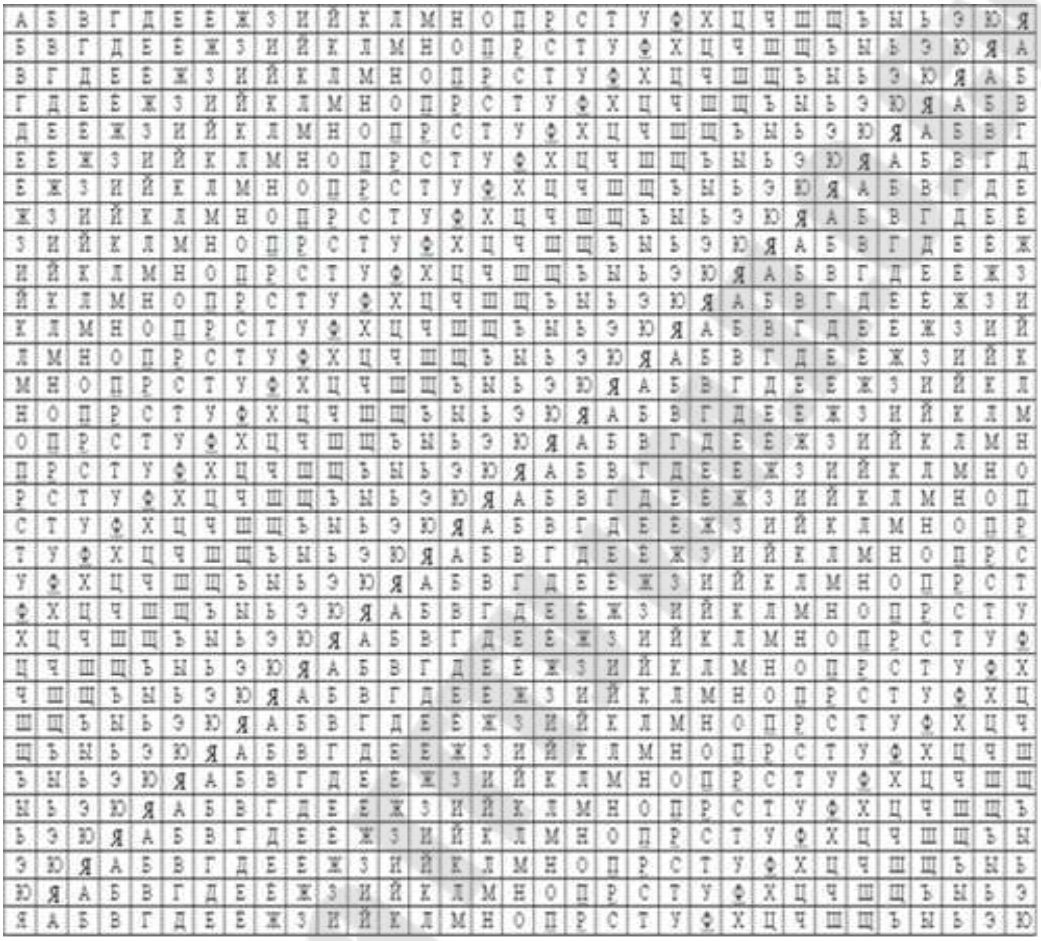

Рис. 1.3. Таблица Виженера

Первая строка служит алфавитом открытого текста, а первый столбец – алфавитом ключа. Для зашифрования открытого сообщения $T_0 = t_1 t_2 \ldots$. Виженер предлагал в качестве ключевой последовательности использовать само сообщение с добавленной к нему в качестве первой буквы, известной отправителю и получателю (этим идея Виженера отличалась от идеи Кардано, у которого не было начальной буквы и система которого не обеспечивала однозначности расшифрования). Последовательности букв подписывались друг под другом. При этом пара букв, стоящих друг под другом, указывала, соответственно, номера строк и столбцов таблицы, на пресечении которых находится знак у шифрованного текста.

Во втором варианте Виженер предлагал в качестве ключевой последовательности использовать шифрованный текст.

Самоключ Виженера был незаслуженно забыт на долгое время, а под шифром Виженера до сих пор понимают самый простой вариант с коротким ключевым словом и с таблицей, состоящей из обычных алфавитов.

В истории криптографии XVII–XVIII века называют эрой «черных кабинетов». В этот период во многих государствах Европы, в первую очередь во Франции, получили развитие дешифровальные подразделения, названные «черными кабинетами». Первый из них образован по инициативе кардинала Ришелье при дворе короля Людовика XIII. Его возглавил первый профессиональный криптограф Франции Антуан Россиньоль. Некоторые оригинальные идеи, возникшие в криптографии в это время, связаны с именем самого Ришелье, который использовал, например, для секретной переписки с королем оригинальный шифр перестановки с переменным ключом.

Много новых идей в криптографии принес XIX век. Изобретение в середине XIX в. телеграфа и других технических видов связи дало новый толчок развитию криптографии. Информация передавалась в виде токовых и бестоковых посылок, т. е. представлялась в двоичном виде. Поэтому возникла проблема «рационального» представления информации, которая решалась с помощью кодов. Коды позволяли передать длинное слово или целую фразу двумя-тремя знаками. Появилась потребность в высокоскоростных способах шифрования и в корректирующих кодах, необходимых в связи с неизбежными ошибками при передаче сообщений.

Во второй половине XIX в. появился весьма устойчивый способ усложнения числовых кодов – гаммирование. Он заключался в перешифровании закодированного сообщения с помощью некоторого ключевого числа, которое и называлось гаммой. Шифрование с помощью гаммы состояло в сложении всех кодированных групп сообщения с одним и тем же ключевым числом. Эту операцию стали называть «наложением гаммы». Например, результатом наложения гаммы 6413 на кодированный текст 3425 7102 8139 являлась числовая последовательность 9838 3515 4552.

Единицы переноса, появляющиеся при сложении между кодовыми группами, опускались. «Снятие гаммы» являлось обратной операцией.

1.4. Современная криптография

В семидесятых годах прошлого века произошло два события, серьезно повлиявших на дальнейшее развитие криптографии. Во-первых, был принят первый стандарт шифрования данных (DES), «легализовавший» принцип Керкгоффа в криптографии. Во-вторых, после работы американских математиков У. Диффи и М. Хеллмана родилась «новая криптография» – криптография с открытым ключом. Оба эти события были рождены потребностями бурно развивающихся средств коммуникаций, в том числе локальных и глобальных компьютерных сетей, для защиты которых потребовались легко доступные и достаточно надежные криптографические средства. Криптография стала широко востребоваться не только в военной, дипломатической, государственной сферах, но также в коммерческой, банковской и других областях.

Вслед за идеей Диффи и Хеллмана, связанной с гипотетическим понятием однонаправленной (или односторонней) функции с секретом, появились «кандидат» на такую функцию и реально осуществленная шифрсистема RSA с открытым ключом. Такая система была предложена в 1978 г. Райвестом, Шамиром и Адлеманом. Парадоксальным казалось то, что в RSA для зашифрования и расшифрования используются разные ключи, причем ключ зашифрования может быть открытым, т. е. всем известным. Вслед за RSA появился целый ряд других систем. В связи с несимметричным использованием ключей стал использоваться термин «асимметричная шифрсистема», в то время как традиционные шифрсистемы стали называться симметричными.

Наряду с идеей открытого шифрования Диффи и Хеллман предложили идею открытого распределения ключей, позволяющую избавиться от защищенного канала связи при рассылке криптографических ключей. Их идея основывалась на сложности решения задачи дискретного логарифмирования, т. е. задачи, являющейся обратной для задачи возведения в степень в конечном поле большого порядка.

1.5. Практическое задание

Порядок выполнения задания:

1. Разработать программу, выполняющую шифрование текста не менее чем двумя перестановочными методами.

2. Разработать программу, выполняющую шифрование текста подстановочным методом (аффинное преобразование).

3. Сохранить полученный криптотекст в текстовый файл.

4. Разработать программу, выполняющую расшифровку криптотекста выбранными перестановочными методами.

5. Разработать программу, выполняющую расшифровку криптотекста подстановочным методом (аффинное преобразование).

6. Оформить отчет по проделанной работе.

В качестве исходного текста взять Фамилию, Имя, Отчество.

Глава 2. АЛГОРИТМЫ СИММЕТРИЧНОГО ШИФРОВАНИЯ

2.1. Алгоритм DES

Стандарт шифрования данных DES опубликован в 1977 г. Национальным бюро стандартов США.

Стандарт DES предназначен для защиты от несанкционированного доступа к важной, но несекретной информации в государственных и коммерческих организациях США. Алгоритм, положенный в основу стандарта, распространялся достаточно быстро, и уже в 1980 г. был одобрен Национальным институтом стандартов и технологий США. С этого момента DES превращается в стандарт не только по названию, но и фактически. Появляются программное обеспечение и специализированные микроЭВМ, предназначенные для шифрования и расшифрования информации в сетях передачи данных.

К настоящему времени DES является наиболее распространенным алгоритмом, используемым в системах защиты коммерческой информации. Более того, реализация алгоритма DES в таких системах становится признаком хорошего тона.

Основные достоинства алгоритма DES:

– используется только один ключ длиной 56 битов;

– зашифровав сообщение с помощью одного пакета, для расшифровки можно использовать любой другой;

– относительная простота алгоритма обеспечивает высокую скорость обработки информации;

– достаточно высокая стойкость алгоритма.

2.1.1. Алгоритм шифрования

DES осуществляет шифрование 64-битовых блоков данных с помощью 56-битового ключа. Расшифрование в DES является операцией обратной шифрованию и выполняется путем повторения операций шифрования в обратной последовательности.

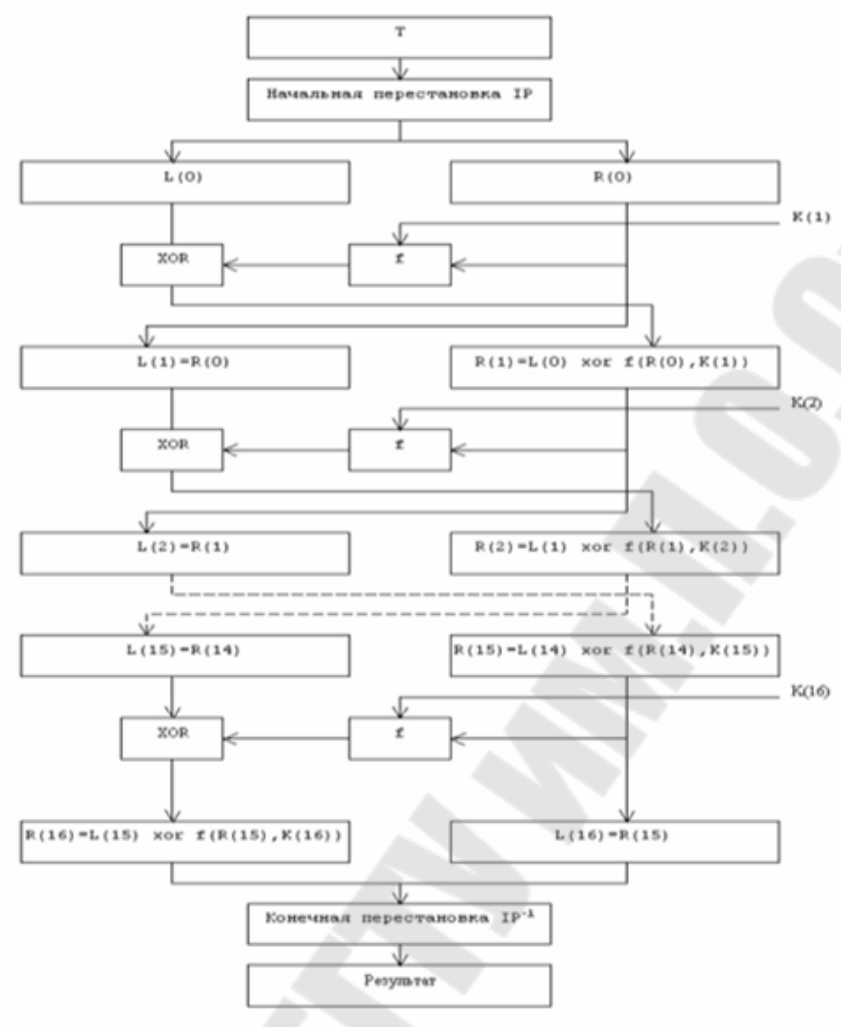

Рис. 2.1. Структура алгоритма шифрования DES

Алгоритм шифрования. Блок *T* преобразуется с помощью матрицы начальной перестановки *IP* (рис. 2.2) следующим образом: бит 58 блока *T* становится битом 1, бит 50 – битом 2 и т. д. Полученная последовательность битов *T* разделяется на две последовательности по 32 бита каждая: *L*(0) – левые или старшие биты, *R*(0) – правые или младшие биты.

58	50	42	34	26	18	10	2	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	59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Рис. 2.2. Таблица перестановки *IP*

Затем выполняется шифрование, состоящее из 16 итераций. Результат *i*-й итерации описывается следующими формулами:

$$L(i) = R\,(i-1)(i) = L\,(i-1)\ \text{xor}\ f\ (R(i-1), K(i)),$$

где xor – операция ИСКЛЮЧАЮЩЕЕ ИЛИ.

Функция *f* называется функцией шифрования. Ее аргументы – это 32-битовая последовательность $R\ (i-1)$, полученная на $(i-1)$-й итерации, и 48-битовый ключ $K(i)$, который является результатом преобразования 64-битового ключа *K*. Подробно функция шифрования и алгоритм получения ключей $K(i)$ описаны ниже. На 16-й итерации получают последовательности $R(16)$ и $L(16)$ (без перестановки), которые конкатенируют в 64-битовую последовательность $R(16)$ $L(16)$. Затем позиции битов этой последовательности переставляют в соответствии с матрицей IP^{-1} (рис. 2.3).

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

Рис. 2.3. Таблица перестановки IP^{-1}

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 матрицей IP^{-1}, а затем над последовательностью бит $R(16)$ и $L(16)$ выполняются те же действия, что и в процессе шифрования, но в обратном порядке.

Рассмотрим функцию шифрования $f\,(R(i-1), K(i))$ (рис. 2.4).

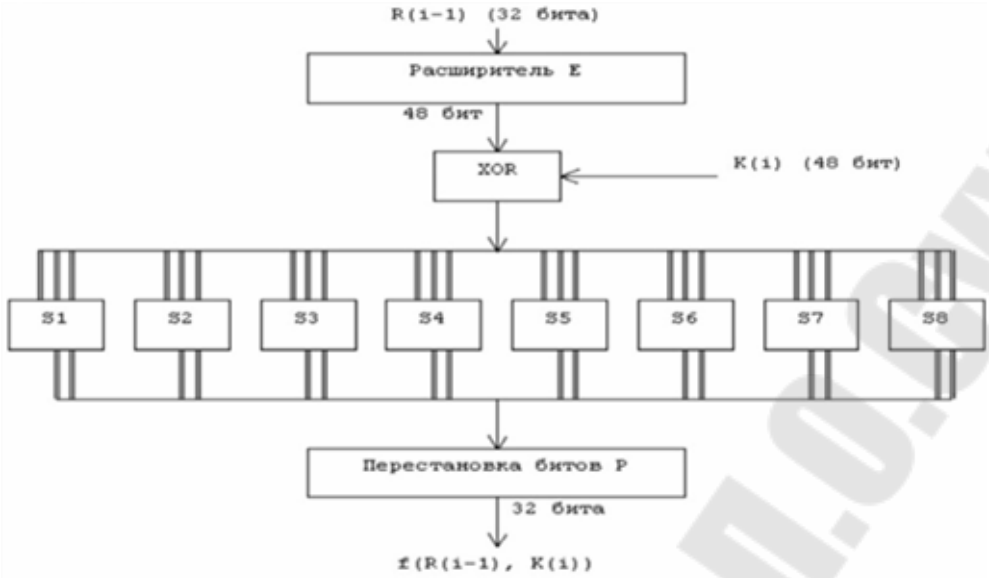

Рис. 2.4. Функция шифрования

Для вычисления значения функции f используются следующие функции-матрицы: E – расширение 32-битовой последовательности до 48-битовой, $S2,\ldots, S8$ – преобразование 6-битового блока в 4-битовый, P – перестановка бит в 32-битовой последовательности. Функция расширения E определяется по рис. 2.5. В соответствии с этой таблицей первые три бита $E\,(R(i-1))$ – это биты 32, 1 и 2, а последние – 31, 32 и 1.

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

Рис. 2.5. Функция расширения E

Результат функции E ($R(i–1)$) есть 48-битовая последовательность, которая складывается по модулю 2 (операция xor) с 48-битовым ключом $K(i)$. Получается 48-битовая последовательность, которая разбивается на восемь 6-битовых блоков $B(1)$ $B(2)$ $B(3)$ $B(4)$ $B(5)$ $B(6)$ $B(7)$ $B(8)$. То есть

$$E\,(R(i-1))\ \text{xor}\ K(i) = B(1)\,B(2)\ldots B(8).$$

Далее эти 8 блоков бит поступают на так называемые S-блоки (рис. 2.6).

Номер строки	Номер столбца 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	S1
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	S2
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	S3
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	S4
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	S5
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	S6
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	S7
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	S8
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	

Рис. 2.6. S-блоки

Алгоритм использования S-блоков. Пусть на вход функции-матрицы *Sj* поступает 6-битовый блок $B(j) = b1b2b3b4b5b6$, тогда двухбитовое число $b1b6$ указывает номер строки матрицы, а $b2b3b4b5$ –

номер столбца. Результатом *Sj* (*B*(*j*)) будет 4-битовый элемент, расположенный на пересечении указанных строки и столбца. Далее, для получения результата функции шифрования надо переставить биты этой последовательности. Для этого применяется функция перестановки *P* (рис. 2.7).

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

Рис. 2.7. Таблица перестановки *P*

Таким образом,

$$f\,(R(i-1), K(i)) = S1\,(B(1)),\ldots, S8\,(B(8)).$$

На каждой итерации используется новое значение ключа *K*(*i*), которое вычисляется из начального ключа *K*. *K* представляет собой 64-битовый блок с восемью битами контроля по четности, расположенными в позициях 8, 16, 24, 32, 40, 48, 56, 64. Для удаления контрольных битов и перестановки остальных используется функция *G* первоначальной подготовки ключа (рис. 2.8).

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

Рис. 2.8. Функция *G*

Результат преобразования *G*(*K*) разбивается на два 28-битовых блока *C*(0) и *D*(0), причем *C*(0) будет состоять из битов 57, 49,…, 44, 36 ключа *K*, а *D*(0) будет состоять из битов 63, 55,…, 12, 4 ключа *K*.

После определения $C(0)$ и $D(0)$ рекурсивно определяются $C(i)$ и $D(i)$, $i = 1…16$. Для этого применяют циклический сдвиг влево на один или два бита в зависимости от номера итерации, как показано на рис. 2.9.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Число сдвига	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Рис. 2.9. Количество сдвигов для вычисления ключа

Полученное значение вновь «перемешивается» в соответствии с функцией H (рис. 2.10).

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

Рис. 2.10. Функция H

Блок-схема алгоритма вычисления ключа приведена на рис. 2.11.

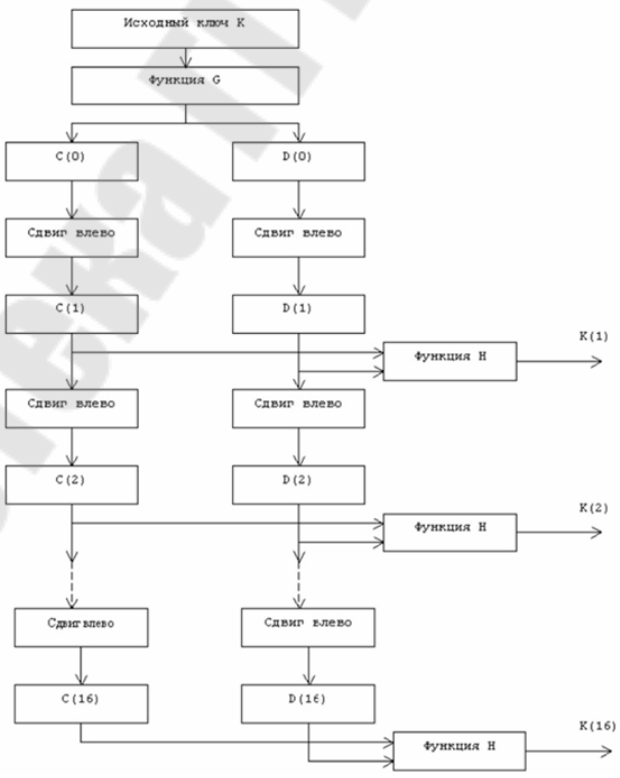

Рис. 2.11. Блок-схема алгоритма вычисления ключа

Восстановление исходного текста осуществляется по этому алгоритму, но вначале вы используете ключ $K(15)$, затем – $K(14)$ и т. д.

После прохождения через 16 итераций шифрования , блок перемешивается согласно обратной таблице перестановки IP^{-1} (рис. 2.12).

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

Рис. 2.12. Обратная перестановка IP^{-1}

2.1.2. Режимы работы

Для наиболее полного удовлетворения всем требованиям, предъявляемым к коммерческим системам шифрования, реализованы несколько режимов работы алгоритма DES. Наиболее широкое распространение получили режимы:

– режим электронной кодовой книги (ECB – Electronic Code Book);

– режим сцепления блоков (CBC – Cipher Block Chaining);

– режим обратной связи по шифротексту (CFB – Cipher Feed Back);

– режим обратной связи по выходу (OFB – Output Feed Back).

DES–ECB. В этом режиме исходный файл M разбивается на 64-битовые блоки (по 8 байтов): $M = M(1)\ M(2)\ldots\ M(n)$. Каждый из этих блоков кодируется независимо с использованием одного и того же ключа шифрования (рис. 2.13). Основное достоинство этого алгоритма – простота реализации. Недостаток – относительно слабая криптостойкость.

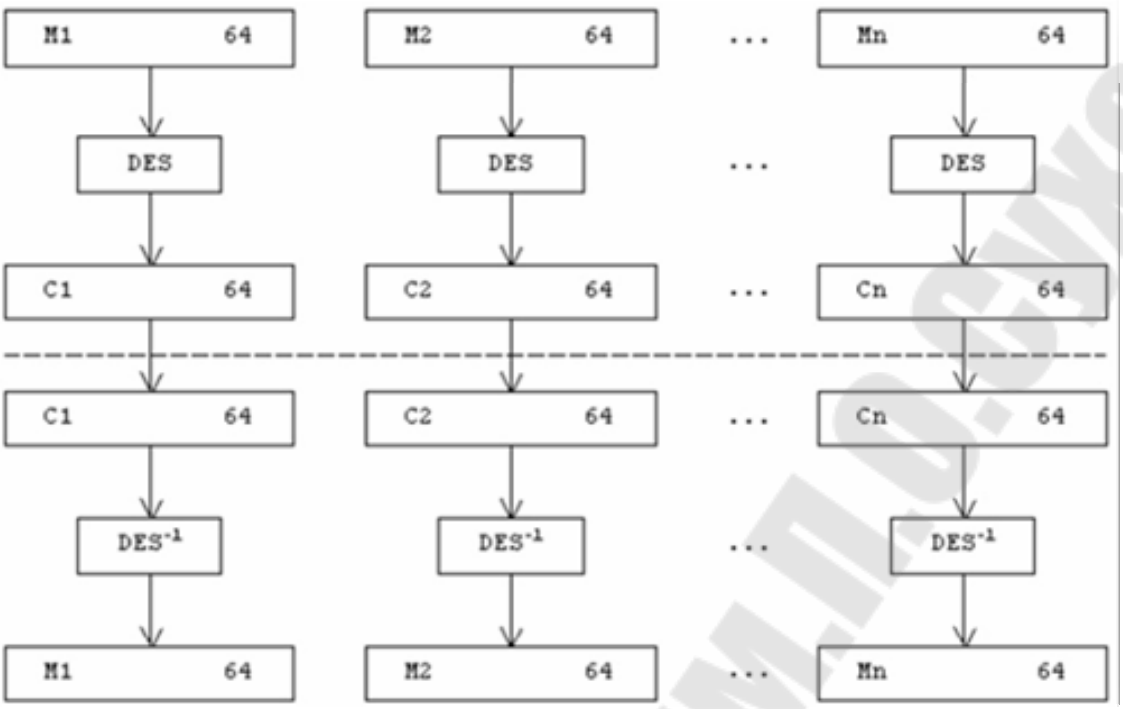

Рис. 2.13. Работа алгоритма DES в режиме ECB

DES–CBC. В этом режиме исходный файл M также, как и в режиме ECB, разбивается на 64-битовые блоки: $M = M(1)\,M(2)\ldots M(n)$. Первый блок $M(1)$ складывается по модулю 2 с 64-битовым начальным вектором IV, который меняется ежедневно и держится в секрете. Полученная сумма затем шифруется с использованием ключа DES, известного и отправителю, и получателю информации. Полученный 64-битовый блок шифротекста $C(1)$ складывается по модулю 2 со вторым блоком исходного текста, результат шифруется и получается второй 64-битовый блок шифротекста $C(2)$ и т. д. Процедура повторяется до тех пор, пока не будут обработаны все блоки исходного текста (рис. 2.14).

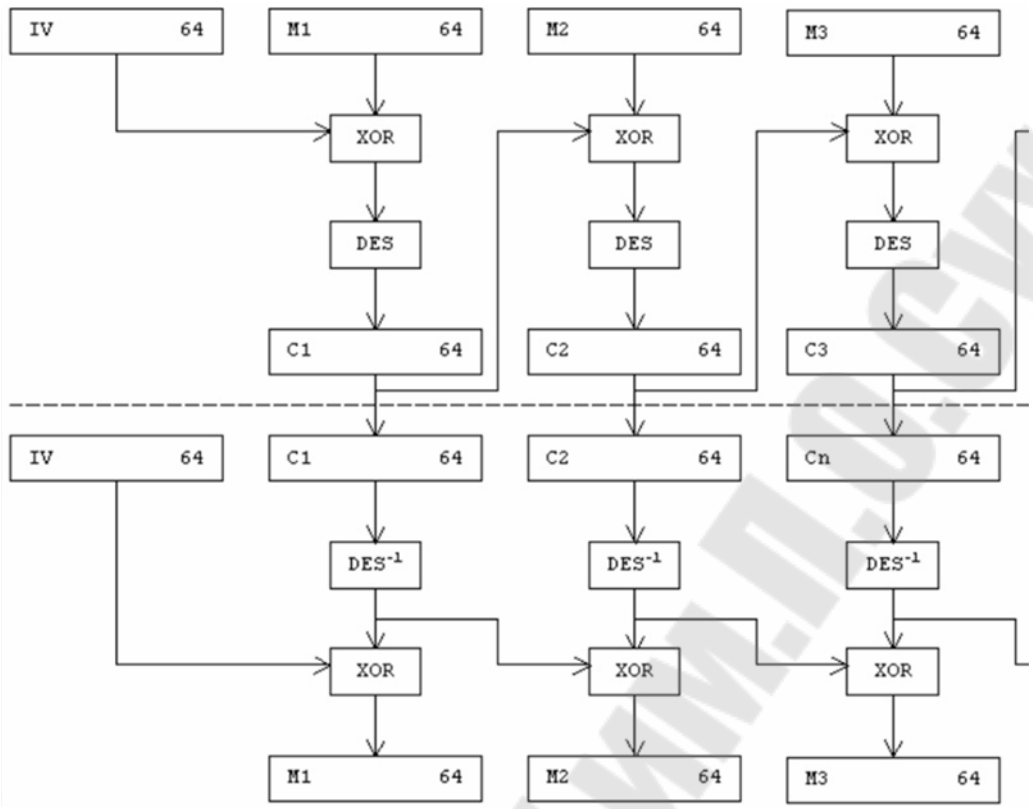

Рис. 2.14. Работа алгоритма DES в режиме CBC

При шифровании в этом режиме шифрования ошибка при передаче приведет к потере только двух блоков исходного текста.

DES–CFB. В режиме CFB вырабатывается блочная «гамма» $Z_0, Z_1, ..., Z_i = \mathrm{DES}_k(C_{i-1})C_i = M_i \oplus Z_i$. Начальный вектор C_0 является синхропосылкой и предназначен для того, чтобы разные наборы данных шифровались по-разному с использованием одного и того же секретного ключа. Синхропосылка посылается получателю в открытом виде вместе с зашифрованным файлом (рис. 2.15).

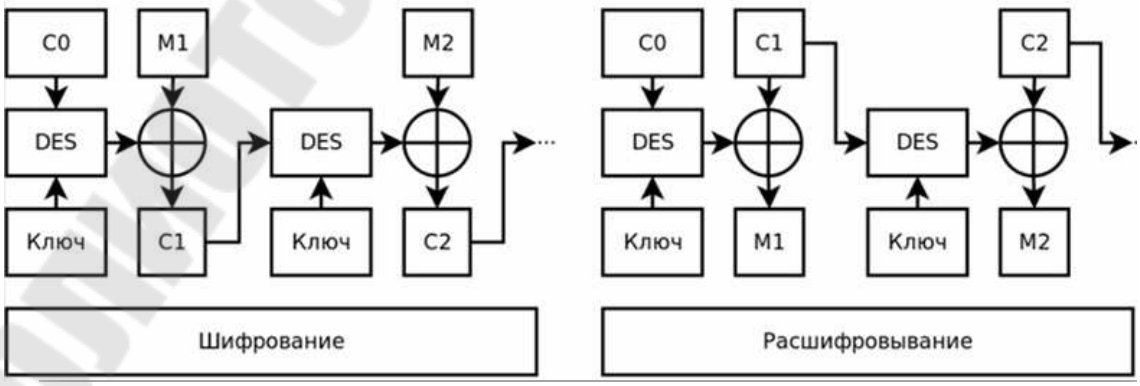

Рис. 2.15. Работа алгоритма DES в режиме CFB

DES–OFB. Режим OFB очень похож на режим CFB (рис. 2.16). В режиме OFB вырабатывается блочная «гамма» $Z_0, Z_1, ..., Z_i = \mathrm{DES}_k(Z_{i-1}) C_i = M_i \oplus Z_i$, $i \geq 1$.

В режимах ECB и OFB искажение при передаче одного 64-битового блока шифротекста приводит к искажению после расшифрования только соответствующего открытого блока, поэтому такие режимы используются для передачи по каналам связи с большим числом искажений.

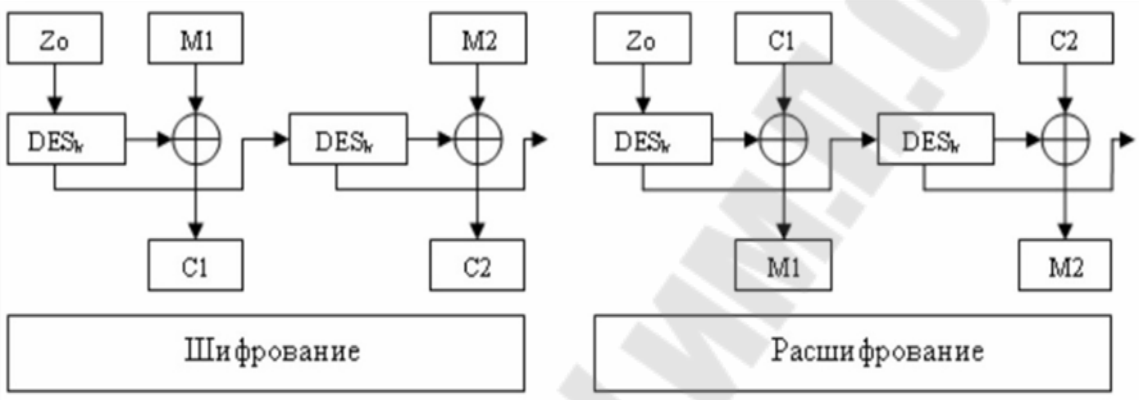

Рис. 2.16. Работа алгоритма DES в режиме OFB

2.1.3. Криптостойкость алгоритма

Нелинейность преобразований в DES средствами только *S*-блоков и использование слабых *S*-блоков позволяет осуществлять контроль над шифрованной перепиской. Выбор *S*-блоков требует соблюдения нескольких условий:

– каждая строка каждого блока должна быть перестановкой множества {0, 1, 2, …, 15};

– *S*-блоки не должны являться линейной или афинной функцией своих аргументов;

– изменение одного бита на входе *S*-блока должно приводить к изменению, по крайней мере, двух битов на выходе;

– для каждого *S*-блока и любого аргумента x значения $S(x)$ должны различаться, по крайней мере, двумя битами.

Из-за небольшого числа возможных ключей появляется возможность их полного перебора на быстродействующей вычислительной технике за реальное время.

В 1998 г. Electronic Frontier Foundation, используя специальный компьютер DES-Cracker, удалось взломать DES за три дня.

2.2. Алгоритм IDEA

IDEA (International Data Encryption Algorithm) – это алгоритм блочного симметричного шифрования, который был предложен на замену стандарта DES [1]. Начальная версия алгоритма IDEA появилась в 1990 г. Разработчики алгоритма, Ксуеджа Лай (Xuejia Lai) и Джеймс Мэсси (James Massey) из швейцарского института ETH Zurich, дали ему название PES (Proposed Encryption Standard – предлагаемый стандарт шифрования). Спустя год алгоритм был модифицирован, чтобы усилить криптостойкость к дифференциальному криптоанализу (атаки на основе намеренно сделанных ошибок при шифровании). Новая версия получила название IPES (Improved PES – улучшенный PES), а еще через год алгоритм сменил название на IDEA (International Data Encryption Algorithm – международный алгоритм шифрования данных).

IDEA запатентован в США и в большинстве европейских стран. Владельцем патента является Ascom-Tech. Использование шифра IDEA в некоммерческих целях бесплатно.

2.2.1. Алгоритм шифрования

В алгоритме шифрования IDEA существует два входа: незашифрованный блок и ключ. Незашифрованный блок имеет длину 64 бита, ключ – 128 бит. Алгоритм IDEA состоит из восьми раундов, за которыми следует заключительное преобразование. Алгоритм разделяет блок на четыре 16-битных подблока. Каждый раунд получает на входе четыре 16-битных подблока и создает четыре 16-битных выходных подблока. Заключительное преобразование также получает на входе четыре 16-битных подблока и создает четыре 16-битных подблока. Каждый раунд использует шесть 16-битных ключей, заключительное преобразование использует четыре подключа, т. е. всего в алгоритме используется 52 подключа (рис. 2.17).

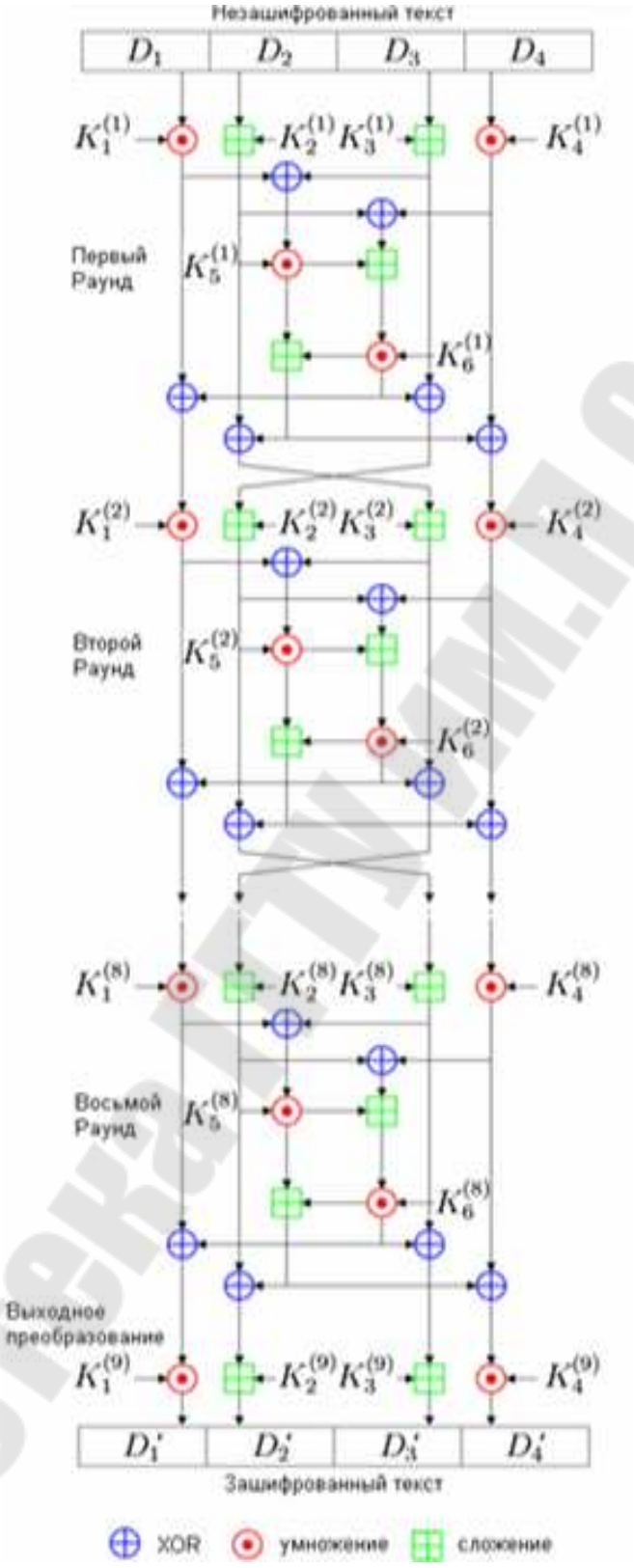

Рис. 2.17. Алгоритм IDEA

Одним из основных элементов алгоритма, обеспечивающих диффузию, является структура, называемая MA (умножение/сложение):

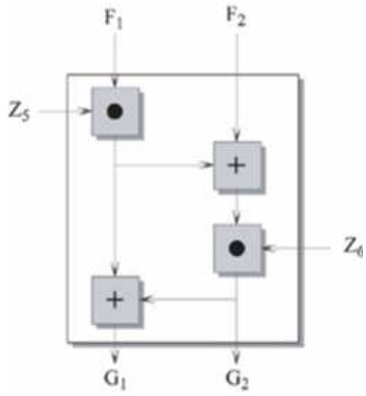

Рис. *2.18*. Структура MA (умножение/сложение)

На вход структуры MA подаются два 16-битных значения и два 16-битных подключа, на выходе создаются два 16-битных значения. Так, каждый бит выхода этой структуры зависит от каждого бита входов незашифрованного блока и от каждого бита подключей. Эта структура повторяется в алгоритме восемь раз, обеспечивая высокоэффективную диффузию. Раунд начинается с преобразования, которое комбинирует четыре входных подблока с четырьмя подключами, используя операции сложения и умножения. Четыре выходных блока этого преобразования комбинируются, используя операцию XOR для формирования двух 16-битных блоков, которые являются входами MA структуры. Кроме того, MA структура имеет на входе еще два подключа и создает два 16-битных выхода.

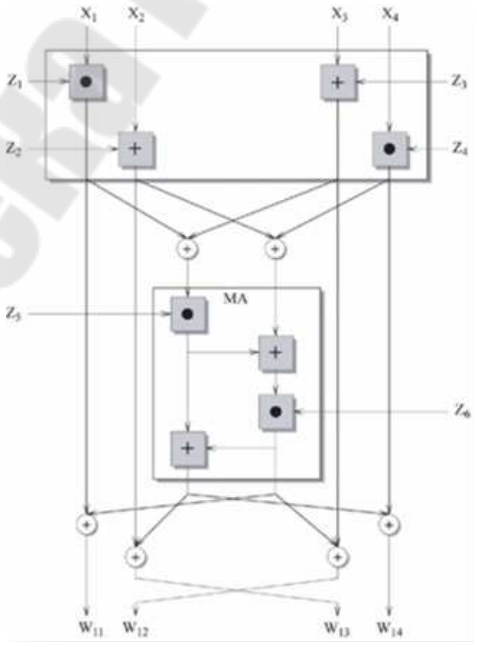

Рис. 2.19. Первый раунд шифрования

В конце раунда четыре выходных подблока первого преобразования комбинируются с двумя выходными подблоками МА структуры, используя XOR для создания четырех выходных подблоков данной итерации. Очень важно, что два выхода, которые частично создаются вторым и третьим входами (*X*2 и *X*3), меняются местами для создания второго и третьего выходов (*W*12 и *W*13). Это увеличивает перемешивание битов и делает алгоритм более стойким для дифференциального криптоанализа.

Рассмотрим девятый раунд алгоритма, обозначенный как заключительное преобразование. Это та же структура, что была описана выше. Единственная разница состоит в том, что второй и третий входы меняются местами. Это сделано для того, чтобы дешифрование имело ту же структуру, что и шифрование. Последняя стадия требует только четыре входных подключа (рис. 2.20).

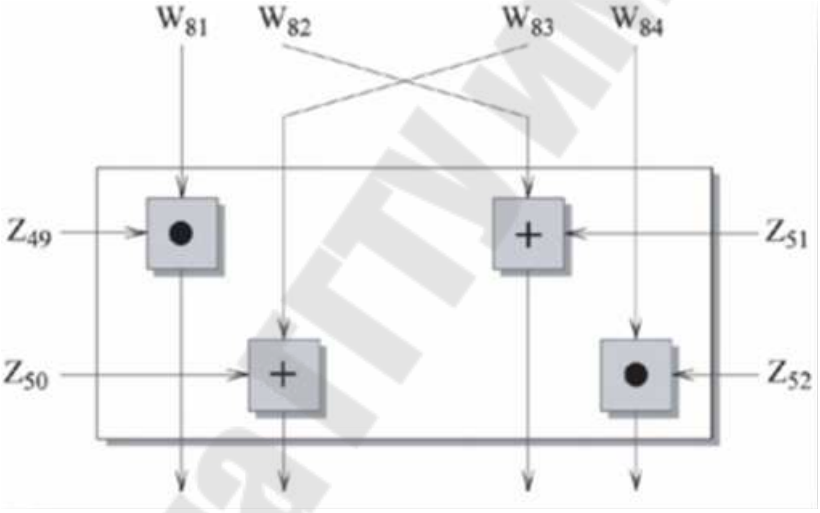

Рис. 2.20. Заключительное преобразование

Процесс генерации ключей заключается в следующем:

– 128-битный ключ разбивается на восемь 16-битных блоков (первые восемь подключей – $K_1^{(1)}K_2^{(1)}K_3^{(1)}K_4^{(1)}K_5^{(1)}K_6^{(1)}K_1^{(2)}K_2^{(2)}$);

– 128-битный ключ циклически сдвигается влево на 25 позиций, после чего новый 128-битный блок снова разбивается на восемь 16-битных блоков – $K_3^{(2)}K_4^{(2)}K_5^{(2)}K_6^{(2)}K_1^{(3)}K_2^{(3)}K_3^{(3)}K_4^{(3)}$.

Дешифрование текста по существу такое же, как и при его шифровании. Единственное отличие состоит в том, что для расшифровки используются другие подключи. В процессе расшифровки подключи должны использоваться в обратном порядке.

2.3. Алгоритм ГОСТ 28147–89

ГОСТ 28147–89 – советский и российский стандарт симметричного шифрования, введенный в 1990 г., также является стандартом СНГ. Полное название – «ГОСТ 28147–89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования».

ГОСТ 28147–89 – это блочный шифроалгоритм, но при использовании метода шифрования с гаммированием может выполнять функции поточного шифроалгоритма.

По некоторым сведениям, история этого шифра гораздо более давняя. Алгоритм, положенный впоследствии в основу стандарта, родился, предположительно, в недрах Восьмого Главного управления КГБ СССР в одном из подведомственных ему закрытых НИИ, вероятно, еще в 1970-х гг. в рамках проектов создания программных и аппаратных реализаций шифра для различных компьютерных платформ.

С момента опубликования ГОСТа на нем стоял ограничительный гриф «Для служебного пользования», и формально шифр был объявлен «полностью открытым» только в мае 1994 г. История создания шифра и критерии разработчиков до сих пор не опубликованы.

2.3.1. Алгоритм шифрования

ГОСТ 28147–89 – блочный шифр с 256-битным ключом и 32 циклами преобразования, оперирующий 64-битными блоками. Основа алгоритма шифра – Сеть Фейстеля.

Алгоритм ГОСТ 28147–89 базируется на схеме Фейстеля и шифрует информацию блоками по 64 бит, которые разбиваются на два подблока по 32 бита (*A* и *B*) (рис. 2.21). Подблок *A* обрабатывается функцией раундового преобразования, после чего его значение складывается со значением подблока *B*, затем подблоки меняются местами. Алгоритм имеет 16 или 32 раунда в зависимости от режима шифрования.

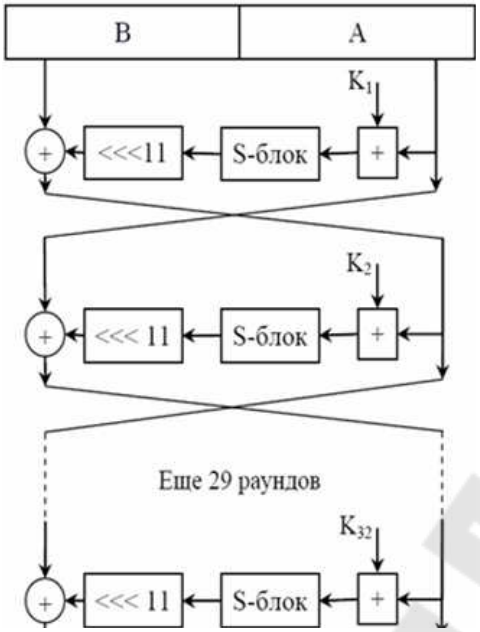

Рис. 2.21. Схема алгоритма

В каждом раунде алгоритма выполняются следующие преобразования.

1. *Наложение ключа*. Содержание подблока B_i складывается по модулю 2^{32} с ключом раунда K. K_j – это 32-битовая часть исходного ключа, используемая в качестве раундового. Алгоритм ГОСТ 28147–89 не использует процедуру расширения ключа, исходный 256-битный ключ шифрования представляется в виде конкатенации (сцепления) восьми 32-битовых подключей: K_0, K_1, K_2, K_3, K_4, K_5, K_6, K_7.

2. *Табличная замена*. После наложения ключа подблок A_i разбивается на восемь частей по четыре бита, значение каждой из которых по отдельности заменяется в соответствии со своей таблицей замены (*S*-блоком). Всего используется восемь *S*-блоков – *S*0, *S*1, *S*2, *S*3, *S*4, *S*5, *S*6, *S*7. Каждый *S*-блок алгоритма ГОСТ 28147–89 представляет собой вектор (одномерный массив) с элементами, пронумерованными от 0 до 15. Значениями *S*-блока являются 4-битовые числа, т. е. целые числа от 0 до 15 (рис. 2.22). Из таблицы *S*-блока берется элемент, порядковый номер которого совпадает со значением, пришедшим на вход подстановки. Таблицы замен не определены стандартом, как это сделано, например, в шифре DES. Сменные значения таблиц замен существенно затрудняют криптоанализ алгоритма. В то же время стойкость алгоритма существенно зависит от их правильного выбора.

Номер S-блока	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	4	10	9	2	13	8	0	14	6	11	1	12	7	15	5	3
1	14	11	4	12	6	13	15	10	2	3	8	1	0	7	5	9
2	5	8	1	13	10	3	4	2	14	15	12	7	6	0	9	11
3	7	13	10	1	0	8	9	15	14	4	6	12	11	2	5	3
4	6	12	7	1	5	15	13	8	4	10	9	14	0	3	11	2
5	4	11	10	0	7	2	1	13	3	6	8	5	9	12	15	14
6	13	11	4	1	3	15	5	9	0	10	14	7	6	8	2	12
7	1	15	13	0	5	7	10	4	9	2	3	14	6	11	8	12

Рис. 2.22. *S*-блоки

3. *Циклический сдвиг*. Полученное значение сдвигается циклически на 11 бит влево и передается на следующий шаг.

4. *Сложение*. Значение, полученное на предыдущем шаге, побитно складывается по модулю 2 с блоком *B*.

5. *Обмен частями*. На этом шаге происходит обмен частями.

2.3.2. Режимы работы

Режимы работы алгоритма следующие:

– режим простой замены;

– режим гаммирования;

– режим гаммирования с обратной связью;

– режим выработки имитовставки.

Шифрование в режиме простой замены проходит согласно указанному выше алгоритму.

Режим простой замены имеет следующие недостатки:

– может применяться только для шифрования открытых текстов с длиной, кратной 64 бит;

– при шифровании одинаковых блоков открытого текста получаются одинаковые блоки шифротекста, что может дать определенную информацию криптоаналитику.

Таким образом, применение ГОСТ 28147–89 в режиме простой замены желательно лишь для шифрования ключевых данных.

Режим гаммирования. При работе ГОСТ 28147–89 в режиме гаммирования описанным ниже образом формируется криптографи-

ческая гамма, которая затем побитно складывается по модулю 2 с исходным открытым текстом для получения шифротекста (рис. 2.23). Шифрование в режиме гаммирования лишено недостатков, присущих режиму простой замены. Так, даже идентичные блоки исходного текста дают разный шифротекст, а для текстов с длиной, не кратной 64 бит, «лишние» биты гаммы отбрасываются. Кроме того, гамма может быть выработана заранее, что соответствует работе шифра в поточном режиме.

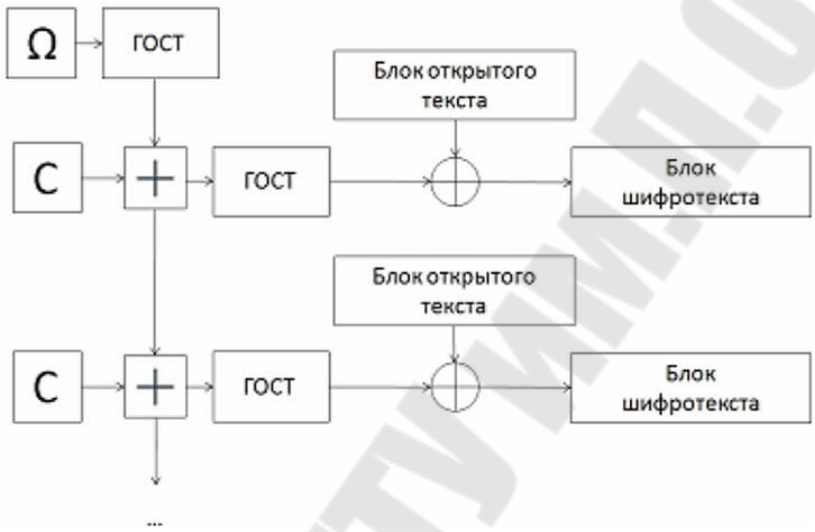

Рис. 2.23. Схема работы в режиме гаммирования

Особенность работы ГОСТ 28147–89 в режиме гаммирования заключается в том, что при изменении одного бита шифротекста изменяется один бит расшифрованного текста.

Режим гаммирования с обратной связью. Алгоритм шифрования похож на режим гаммирования, однако гамма формируется на основе предыдущего блока зашифрованных данных (рис. 2.24).

При изменении одного бита шифротекста, полученного с использованием гаммирования с обратной связью, в соответствующем блоке расшифрованного меняется только один бит, так же затрагивается последующий блок открытого текста. При этом все остальные блоки остаются неизменными.

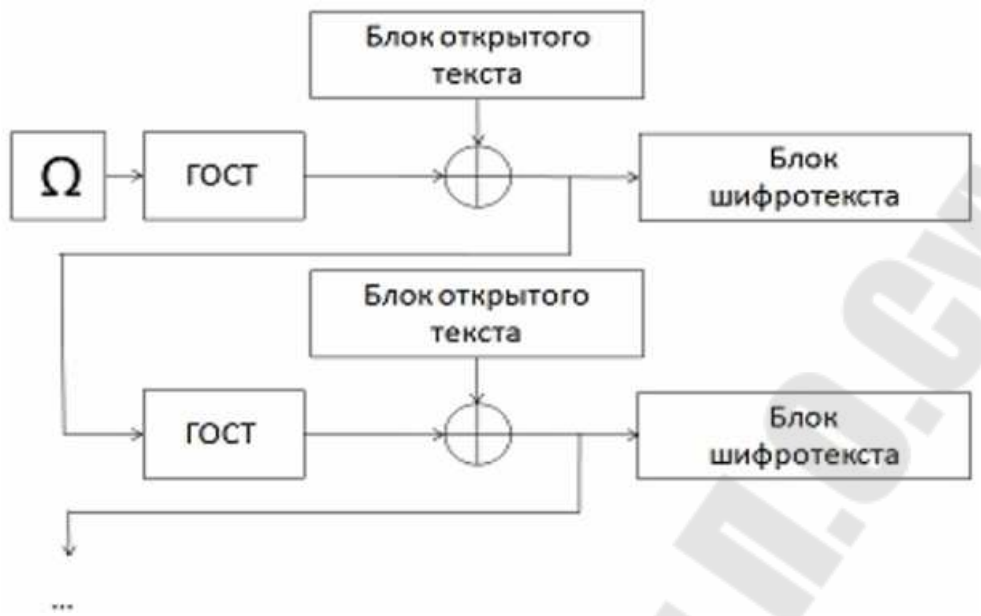

Рис. 2.24. Схема работы в режиме гаммирования с обратной связью

Режим выработки имитовставки. При работе в режиме имитовставки создается некоторый дополнительный блок, зависящий от всего текста и ключевых данных (рис. 2.25). Данный блок используется для проверки того, что в шифротексте не были внесены изменения. Это особенно важно для шифрования в режиме гаммирования, где злоумышленник может изменить конкретные биты, даже не зная ключа.

Рис. 2.25. Схема работы в режиме имитовставки

Алгоритм:

• Блок открытых данных записывается в $N1$ и $N2$, после чего они подвергаются преобразованию, соответствующему 16 циклам шифрования в режиме простой замены.

• К полученному результату побитно по модулю 2 прибавляется следующий блок открытых данных. Сумма также шифруется в соответствии с предыдущим пунктом.

• После добавления и шифрования последнего блока из результата выбирается имитовставка длиной L бит: с бита с номером $32 - L$ до 32 (отсчет начинается с 1). Имитовставка передается по каналу связи после зашифрованных данных.

2.3.3. Криптостойкость алгоритма

Считается, что ГОСТ устойчив к таким широко применяемым методам, как линейный и дифференциальный криптоанализ. Обратный порядок использования ключей в последних восьми раундах обеспечивает защиту от атак скольжения и отражения. А. Г. Ростовцев, Е. Б. Маховенко, А. С. Филиппов, А. А. Чечулин в своей работе описали вид криптоанализа, который сводится к построению алгебраической целевой функции и нахождению ее экстремума. Были выделены классы слабых ключей, в частности, показано, что разреженные ключи (со значительным преобладанием 0 или 1) являются слабыми. По мнению авторов, их метод в любом случае лучше, чем полный перебор, однако без численных оценок.

В мае 2011 г. известный криптоаналитик Николя Куртуа доказал существование атаки на данный шифр, имеющей сложность в 2^8 раз меньше сложности прямого перебора ключей при условии наличия 2^{64} пар «открытый текст – закрытый текст». Данная атака не может быть осуществлена на практике ввиду слишком высокой вычислительной сложности. В большинстве других работ также описываются атаки, применимые только при некоторых предположениях, таких как определенный вид ключей или таблиц замен, некоторая модификация исходного алгоритма, или же требующие все еще недостижимых объемов памяти или вычислений. Вопрос о наличии применимых на практике атак без использования слабости отдельных ключей или таблиц замены остается открытым.

2.4. Алгоритм AES

В 1998 г. NIST (National Institute of Standards and Technology) объявили конкурс по созданию алгоритма симметричного шифрования под названием Advanced Encryption Standard (сокращенно AES). Этот алгоритм должен был стать новым стандартом USA на замену уже устаревшему стандарту Digital Encryption Standard (сокращенно DES), принятым в 1977 г. Причиной для смены стандарта являлась малая длина ключа DES, которая составляла 56 бит, что позволяло злоумышленникам взламывать DES, применяя простой метод перебора ключей. Также проблему вызывала архитектура DES, которая была направлена на аппаратную реализацию и позволяла программным реализациям получать достаточную скорость быстродействия в связи с нехватками ресурсов. Хотя и модификация TripleDES обладала достаточно большой длиной ключа, но была еще медленнее, чем классический DES. 2 января 1997 г. NIST объявляет о намерении выбрать новый алгоритм взамен устаревшей DES. 12 сентября 1997 г. был объявлен конкурс. Было разрешено представлять свой алгоритм любым организациям и группам исследователей. Все параметры для кандидатов были в свободном доступе, но NIST потребовал предварительного объявления основных принципов работы алгоритмов. В этот раз NIST решил не повторять ошибки при выборе DES, которые заключались в закрытии публикации данных об исследовании алгоритмов-кандидатов.

Основные требования к кандидатам на стандарт AES:

– метод шифрования должен быть блочным;

– длина обрабатываемого блока должна равняться 128 битам;

– размерность ключей должна равняться 128, 192 или 256 битам.

Дополнительные рекомендации для конкурсантов:

– ориентироваться на 32-разрядные процессоры;

– должны использоваться только те операции, которые возможно было реализовать как программно, так и аппаратно на микрочипах;

– не усложнять структуру для простого криптоанализа шифра, для проверки заинтересованными лицами возможных скрытых особенностей.

2 октября 2000 г. объявили победителя конкурса: им стал алгоритм Rijndael. Сразу после завершения конкурса была начата процедура стандартизации, и уже 28 февраля 2001 г. был опубликован готовый проект, а 26 ноября 2001 г. AES был принят как новый стандарт.

2.4.1. Алгоритм шифрования

В алгоритме AES используются определенные функции как для шифрования, так и для дешифрования, основные функции алгоритма шифрования приведены в табл. 2.1.

Таблица 2.1

Функции алгоритма AES

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
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, берет четырехбайтовое слово и выполняет циклическую перестановку
ShiftRows()	Преобразование в процессе шифрования, которое выполняется над state, циклически сдвигая последние три 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

При шифровании алгоритмом AES необходим секретный ключ размером 128 бит, ключ представляется как матрица размерностью 4×4 байта. В начале происходит разбитие текста на блоки размерностью 128 бит, если в последнем блоке не хватает данных для его полного заполнения, то свободное место забивается нулями [2]. Двумерный массив размерностью 16 байт, в который заносят по очереди блоки с начальными данными, обычно обозначается как state [*a*] [*b*]. Шифрование этих блоков происходит независимо друг от друга, что позволяет распараллеливать эти процессы, и конечный результат записывается

в один массив. Для шифрования каждого блока требуется как минимум 10 раундов, в каждый раунд входят следующие функции:

– KeyExpansion;
– AddKey;
– SubBytes;
– ShiftRows;
– MixColumns;
– AddRoundKey.

Сначала из основного ключа с помощью функции KeyExpansion генерируются раундовые ключи, каждый раундовый ключ используется один раз в одном раунде. После происходит суммирование данных с основным ключом в функции AddRoundKey. После выполнения функции AddRoundKey программа уходит в цикл, пока счетчик не сравняется с заранее установленным количеством раундов для данного случая. В теле цикла вызывается функция замены байтов SubBytes, затем циклический сдвиг строк ShiftRows, перемешивание MixColumns и суммирование с раундовым ключом AddRoundKey. В последнем 10-м раунде функция MixColumns пропускается, наглядный пример работы алгоритма шифрования представлен в блок-схеме (рис. 2.26).

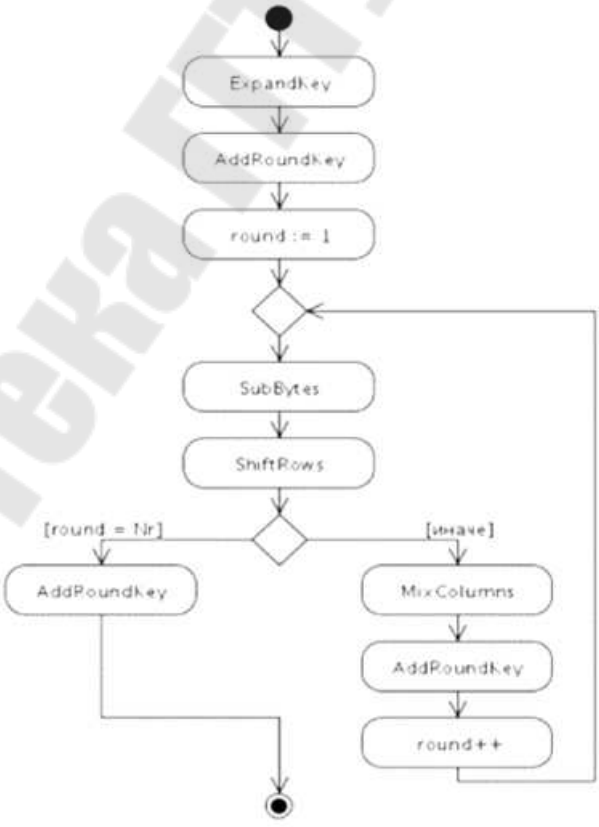

Рис. 2.26. Алгоритм шифрования в последнем 10-м раунде

Расширение ключа KeyExpansion. Для создания раундовых ключей, которые используются в функции AddRoundKey, алгоритм AES берет секретный ключ и производит расширение ключа.

Расширенный ключ состоит из 16 байт (четыре байта на слово), обозначаемых ниже как w_i, где i находится в диапазоне [0…44]. Полная длина расширенного ключа для всех раундов программы составляет 1408 бит, на каждый раунд дается ключ длиной 128 бит. Для расширения базового ключа используется массив констант, который называется R_{con}. Нумерация массива начинается от 1 до 256+3. Их значения определяются следующим образом:

$$R_{\text{con}1} = 1;$$

$$R_{\text{con}k} = 2R_{\text{con}k-1} = 2^{k-1}, \text{ для } k = 2, 3, \ldots 255;$$

$$R_{\text{con}k} = 0, \text{ для } k = 256, 257, 258.$$

Процедуру расширения ключа можно описать в виде последовательности следующих операций:

1. Основной ключ шифрования копируется в первые четыре слова расширенного ключа.

2. Для получения остальных частей ключа происходит их генерация следующим образом:

– если i кратно 4, то $w_i = \text{SubBytes}\,(\text{RotByte}(w_{i-1})) + R_{\text{con}\,(i/4)}$;

– если i не кратно 4, то $w_i = w_{i-4} + w_{i-1}$.

Преобразование AddRoundKey и AddKey. В данной функции происходит добавление раундового ключа в обрабатываемый массив по средствам поразрядного суммирования с таблицей раундовых ключей, для AddKey – с обычным ключом. Данное преобразование является обратным самим для себя.

Преобразование SubBytes. Функция SubBytes производит поочередную замену элементов матрицы с исходными данными на элементы из фиксированной матрицы *S*-box (рис. 2.27). Основной целью этой замены является усложнение линейного дифференциального криптоанализа.

	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Рис. 2.27. Таблица подстановки *S*-box

Преобразование ShiftRows. Функция ShiftRows производит побайтовый сдвиг строк влево по следующему алгоритму: первая строка не изменяется, вторая сдвигается на 1 байт влево, третья строка на 2 байта, а четвертая на 3 байта.

Преобразование MixColumns. При MixColumn столбцы состояния рассматриваются как многочлены над GF(2^8) и умножаются по модулю x^4+1 на фиксированный многочлен $a(x)$:

$$a(x)=\{03\}x3+\{01\}x2+\{01\}x+\{02\}.$$

Этот многочлен – взаимно простой с x^4+1 и поэтому обратим.

Дешифрование. Все преобразования, которые были произведены при шифровании, однозначны и вследствии этого имеют обратное преобразование, чтобы выполнить дешифрование для алгоритма AES.

Порядок обратного преобразования выглядит так: производится расширение основного ключа KeyExpansion, после чего программа уходит в цикл до полного прохождения 9 раундов из 10, в которых выполняются следующие функции по порядку:

1. AddRoundKey – в данной функции происходит суммирование обрабатываемых данных с раундовым ключом.

2. InvMixColumns – в этой функции происходит перемешивание столбцов state: обратная функции MixColumns.

3. InvShiftRows – данная функция производит обратный функции ShiftRows циклический сдвиг строк state.

4. InvSubBytes – в этой функции происходит замена байтов state по таблице замен для дешифрования.

После успешного прохождения девяти раундов программа выходит из цикла и исполняет оставшиеся функции для последнего раунда:

– AddRoundKey;

– InvShiftRows;

– InvSubBytes.

Программа выполняет данные процедуры со всеми блоками исходных данных, после чего объединяет их в выходном файле.

Подробное описание функций алгоритма дешифрования AES представлено ниже.

Преобразование InvMixColumns. Преобразование InvMixColumns является обратным для преобразования MixColumns, и преобразование в этой функции подобно самой функции.

MixColumns, где каждый столбец умножается на особый многочлен $d(x)$, который определяется:

$$('03'x\text{^}3+'01'x\text{^}2+'01'x+'02')\oplus='01';$$

$$d(x)='0B'x\text{^}3+'0D'x\text{^}2+'09'+'0E'.$$

Преобразование InvShiftRows. Преобразование InvShiftRows является обратным для ShiftRows. В данном случае байты 2, 3 и 4 ряда сдвигаются вправо. Вторая строка (нумерация строк начинается с 0) смещается на 1 байт, в третьей строке – на 2 байта, а в четвертой – на 3 байта.

Преобразование InvSubBytes. Преобразование InvSubBytes тоже выполняет замену байт, но уже с помощью обратной таблицы замен, называемой InvSbox. Обратная таблица замен приведена на рис. 2.28.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
1	7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
3	08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
5	6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
6	90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06
7	D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Рис. 2.28. Таблица замен InvSbox

2.4.2. Криптостойкость алгоритма

Для трех вариантов ключей AES полный перебор требует 2^{127}, 2^{191} или 2^{255} операций соответственно.

В соответствии с оценками разработчиков шифр устойчив против следующих видов криптоаналитических атак:

– дифференциальный криптоанализ;

– линейный криптоанализ;

– криптоанализ на основе связанных ключей (слабых ключей в алгоритме нет).

Эту ситуацию известный криптограф Брюс Шнайер комментирует так: «Я не думаю, что сегодня существует какая-либо практическая атака на AES. Если Вы знаете, как взломать *n*-раундовый шифр, то удвойте или утройте число раундов. Это может быть или AES-128 с 16, или AES-192 с 20, или AES-256 с 28 раундами. А полного пересмотра стандарта снова и снова мы не хотим» (2009).

2.5. Практическое задание

Задание 1. Алгоритм шифрования DES и ГОСТ

I. Реализовать приложение для шифрования, позволяющее выполнять следующие действия (варианты заданий – в табл. 2.2):

1. Шифровать данные по заданному в варианте алгоритму:

– шифруемый текст должен храниться в одном файле, а ключ шифрования – в другом;

– зашифрованный текст должен сохраняться в файл;

– в процессе шифрования предусмотреть возможность просмотра и изменения ключа, шифруемого и зашифрованного текстов в шестнадцатеричном и символьном виде.

II. Реализовать приложение для дешифрования, позволяющее выполнять следующие действия:

2. Дешифровать данные по заданному в варианте алгоритму:

– зашифрованный текст должен храниться в одном файле, ключ – в другом;

– расшифрованный текст должен сохраняться в файл;

– в процессе дешифрования предусмотреть возможность просмотра и изменения ключа, зашифрованного и расшифрованного текстов в шестнадцатеричном и символьном виде.

Таблица 2.2

Варианты заданий

Вариант	Алгоритм шифрования	Режим работы
1	ГОСТ 28147–89	Режим простой замены
2	ГОСТ 28147–89	Гаммирование
3	ГОСТ 28147–89	Гаммирование с обратной связью
4	ГОСТ 28147–89	Выработка имитовставки
5	DES	Режим электронной кодовой книги (ECB)
6	DES	Режим сцепления блоков (CBC)
7	DES	Режим обратной связи по шифротексту (CFB)
8	DES	Режим обратной связи по выходу (OFB)

Задание 2. Алгоритм шифрования AES

I. Реализовать приложение для шифрования, позволяющее выполнять следующие действия:

1. Шифровать данные по заданному в варианте алгоритму:

– шифрумый текст должен храниться в одном файле, а ключ шифрования – в другом;

– зашифрованный текст должен сохраняться в файл;

– в процессе шифрования предусмотреть возможность просмотра и изменения ключа, шифруемого и зашифрованного текстов в шестнадцатеричном и символьном виде.

II. Реализовать приложение для дешифрования, позволяющее выполнять следующие действия:

2. Дешифровать данные по заданному в варианте алгоритму:

– зашифрованный текст должен храниться в одном файле, ключ – в другом;

– расшифрованный текст должен сохраняться в файл;

– в процессе дешифрования предусмотреть возможность просмотра и изменения ключа, зашифрованного и расшифрованного текстов в шестнадцатеричном и символьном виде.

В качестве исходного текста взять Фамилию, Имя, Отчество.

Глава 3. АЛГОРИТМЫ АСИММЕТРИЧНОГО ШИФРОВАНИЯ

3.1. Математические основы криптографии

Наибольший общий делитель. Наибольшим общим делителем (НОД, или англ. – Greatest Common Divider – GCD) целых чисел $a1$, $a2$, …, an называется такой положительный общий делитель этих чисел, который делится на любой другой общий делитель этих чисел.

Пример:

НОД(21, 15) = 3;

НОД(27, 44) = 1;

НОД(120, 66) = 6.

Алгоритм Евклида. Используется для нахождения наибольшего общего делителя двух чисел. Идея: НОД(a, b) = НОД(b, r), где $a = bq + r$.

Пример: НОД(22, 8) = ?

$22 = 8 \cdot 2 + 6$

(22, 8) = (8, 6)

$8 = 6 \cdot 1 + 2$

(8, 6) = (6, 2)

$6 = 2 \cdot 2 + 2$

(6, 2) = (2, 2)

$2 = 2 \cdot 1 + 0$

Получили: НОД(22, 8) = 2.

Бинарный алгоритм. Данный алгоритм также используется для нахождения наибольшего общего делителя двух чисел и базируется на следующих четырех утверждениях:

1) если оба числа a и b – четные, то НОД(a, b) = 2 · НОД(a/2, b/2);

2) если a – четное, b – нечетное, то НОД(a, b) = НОД(a/2, b);

3) НОД(a, b) = НОД(b, $a – b$);

4) если a и b – нечетны, то ($a – b$) – четно.

Пример: НОД(1173, 323) = ?

(1173, 323) = (323, 850) = (323, 425) = (323, 102) = (323, 51) = (51, 272) = (51, 136) = (51, 68) = (51, 34) = (51, 17) = (17, 34) = (17, 17) = 17.

Получили: НОД(1173, 323) = 17.

Простые числа. Положительное целое не равное нулю число называется простым, если оно делится только на самого себя и на единицу.

Примеры: 11 – простое; 29 – простое; 56 – составное ($56 = 7 \cdot 4 \cdot 2$).

Два числа m и n называются взаимно простыми, если они не имеют общих делителей кроме единицы, т. е. наибольший общий делитель НОД(m, n) = 1.

Функция Эйлера. Функцией Эйлера $\varphi(n)$ ($n \geq 1$) называют число положительных целых чисел меньших n и взаимно простых с n.

Примеры: $\varphi(1) = 0$; $\varphi(2) = 1$; $\varphi(3) = 2$; $\varphi(4) = 2$; $\varphi(5) = 4$; $\varphi(6) = 2$; $\varphi(7) = 6$; $\varphi(8) = 4$; $\varphi(9) = 6$; $\varphi(10) = 4$; $\varphi(11) = 10$.

Если n – простое число, то $\varphi(n) = n - 1$.

Пример: $\varphi(31) = 30$.

Если $n = p \cdot q$, где p и q – простые числа, то $\varphi(n) = (p - 1)(q - 1)$.

Пример: $\varphi(35) = \varphi(5) \cdot \varphi(7) = 4 \cdot 6 = 24$.

Обобщенный алгоритм вычисления функции Эйлера для произвольного числа n следующий. Если n представить как произведение простых чисел в соответствующих степенях:

$$n = p_1^{a_1} \cdot p_2^{a_2} \cdot p_3^{a_3} \cdot \ldots \cdot p_r^{a_r} \, (p_1, p_1, \ldots, p_r - \text{простые}),$$

то

$$\varphi(n) = n(1 - 1/\, p_1)(1 - 1/\, p_2) \cdot \ldots \cdot (1 - 1/\, p_r).$$

Пример:

3

$\varphi(2700) = ?$

2700 можно представить как $2^2 \cdot 3^3 \cdot 5^2$.

Тогда $\varphi(2700) = 2700(1 - 1/2)(1 - 1/3)(1 - 1/5) = 720$.

Теорема Эйлера. Если $n \geq 0$ – положительное целое число и (a, n) = 1, где a – целое, то справедливо:

$$a^{\varphi(n)} = 1 \bmod n.$$

Пример: $41126 = ? \bmod 127$.

НОД(41,127) = 1, $\varphi(127) = 126$, поэтому $41126 = 1 \bmod 127$.

Пример: $4336 = ? \bmod 377$.

НОД(336, 377) = 1, $377 = 13 \cdot 29$, 13 и 29 – простые числа, поэтому $\varphi(377) = \varphi(13) \cdot \varphi(29) = 12 \cdot 28 = 336$, поэтому $4336 = 1 \bmod 377$.

Взаимообратные числа. Для числа n взаимообратным по модулю r называется такое число m, для которого выполняется:

$$(n \cdot m) \bmod r = 1,$$

или

$$n \cdot m = 1 \bmod.$$

Доказательство: по теореме Эйлера: $n^{\varphi(n)} = 1 \bmod r$ или

$$1 = n^{\varphi(n)} \bmod r.$$

Перемножив два последних уравнения, получим $n \cdot m = n^{\varphi(n)} \bmod r$; разделив обе части на n, получим $m = n^{\varphi(n)-1} \bmod r$.

Китайская теорема об остатках. Одним из важных результатов теории чисел является так называемая китайская теорема об остатках (КТО). По существу эта теорема утверждает, что можно восстановить целое число по множеству его остатков от деления на числа из некоторого набора попарно взаимно простых чисел. Эта теорема была доказана приблизительно в 100 г. до н. э. Существуют несколько формулировок КТО. Представим здесь некоторые из них.

Пусть $m_1, m_2, ..., m_k$ – натуральные попарно взаимно простые числа, а $r_1, r_2, ..., r_k$ – некоторые целые числа, тогда существует такое целое число x, что оно будет решением системы сравнений:

$$\begin{cases} x \equiv r_1 \bmod m_1; \\ x \equiv r_2 \bmod m_2; \\ \quad ... \\ x \equiv r_k \bmod m_k. \end{cases}$$

То есть решение системы сравнений существует и единственное по модулю $m_1, m_2, ..., m_k$.

Алгоритм решения представленной системы выглядит следующим образом:

1. Найти $M = m_1 \cdot m_2 \cdot ... \cdot m_k$.

2. Найти $M_1 = \frac{M}{m_1}, M_2 = \frac{M}{m_2}, ..., M_k = \frac{M}{m_k}$.

3. Используя соответствующие модули $m_1, m_2, ..., m_k$, найти мультипликативную инверсию $M_1, M_2, ..., M_k$. Обозначим ее $M^{-1}_1, M^{-1}_2, ..., M^{-1}_k$.

4. Решить систему уравнений:

$$x = \left(a_1 \cdot M_1 \cdot M_1^{-1} \cdot a_2 \cdot M_2 \cdot M_2^{-1} + ... + a_k \cdot M_k \cdot M_k^{-1}\right) \bmod m.$$

5. В числе Z два числа a и b мультипликативно инверсны друг другу, если

$$a \cdot b \equiv 1 \bmod n.$$

Пример. Если модуль равен 13, то мультипликативная инверсия 2 является 7. Другими словами:

$$2 \cdot 7 \equiv 1 \bmod 13.$$

Следующий пример содержит систему уравнений с различными модулями:

$$\begin{cases} x \equiv 2 \bmod 3; \\ x \equiv 3 \bmod 5; \\ x \equiv 2 \bmod 7. \end{cases}$$

Решаем систему уравнений:

$$M = 3 \cdot 5 \cdot 7 = 105;$$

$$M_1 = \frac{105}{3} = 35,\ M_2 = \frac{105}{5} = 21,\ M_3 = \frac{105}{7} = 15.$$

Находим мультипликативную инверсию $M^{-1}_1 = 2, M^{-1}_2 = 1,$ $M^{-1}_3 = 1.$

Решаем следующую систему уравнений:

$$x = \left(a_1 \cdot M_1 \cdot M_1^{-1} + a_2 \cdot M_2 \cdot M_2^{-1} + a_3 \cdot M_3 \cdot M_3^{-1}\right) \bmod m = 23.$$

Проверка:

$$\begin{cases} 23 \equiv 2 \bmod 3; \\ 23 \equiv 3 \bmod 5; \\ 23 \equiv 2 \bmod 7. \end{cases}$$

3.2. Алгоритм шифрования с открытым ключом RSA

Опубликованная в ноябре 1976 г. статья Уитфилда Диффи и Мартина Хеллмана «Новые направления в криптографии» перевернула представление о криптографических системах, заложив основы криптографии с открытым ключом [3]. Разработанный впоследствии

алгоритм Диффи–Хеллмана позволял двум сторонам получить общий секретный ключ, используя незащищенный канал связи. Однако этот алгоритм не решал проблему аутентификации. Без дополнительных средств пользователи не могли быть уверены, с кем именно они сгенерировали общий секретный ключ. Изучив эту статью, трое ученых Рональд Ривест (Ronald Linn Rivest), Ади Шамир (Adi Shamir) и Леонард Адлеман (Leonard Adleman) из Массачусетского технологического института (MIT) приступили к поискам математической функции, которая бы позволяла реализовать сформулированную Уитфилдом Диффи и Мартином Хеллманом модель криптографической системы с открытым ключом. После работы над более чем 40 возможными вариантами им удалось найти алгоритм, основанный на различии в том, насколько легко находить большие простые числа и насколько сложно раскладывать на множители произведение двух больших простых чисел, получивший впоследствии название RSA. Система была названа по первым буквам фамилий ее создателей.

3.2.1. Алгоритм шифрования

Алгоритм RSA (буквенная аббревиатура от фамилий Rivest, Shamir и Adleman) – криптографический алгоритм с открытым ключом, основывающийся на вычислительной сложности задачи факторизации больших целых чисел. Криптосистема RSA стала первой системой, пригодной и для шифрования, и для цифровой подписи.

Первым этапом любого асимметричного алгоритма является создание пары ключей (открытого и закрытого) и распространение открытого ключа по всему миру.

Создание ключей. Для алгоритма RSA этап создания ключей состоит из следующих операций:

1. Выбираются два очень больших простых числа p и q.

2. Вычисляется их произведение $n = pq$, которое называется модулем.

3. Вычисляется значение функции Эйлера от числа n:

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

4. Выбирается произвольное число $e(1 < e < \varphi(n))$, взаимно простое со значением функции $\varphi(n)$. Число называется открытой экспонентой.

5. С помощью алгоритма Евклида вычисляется число d, которое удовлетворяет условию

$$de \equiv 1 \bmod \varphi(n).$$

6. Пара $\{e, n\}$ публикуется в качестве открытого ключа RSA.

7. Пара $\{d, n\}$ играет роль закрытого ключа RSA и держится в секрете.

Шифрование и расшифрование. Предположим, отправитель хочет послать получателю сообщение m. Сообщениями являются целые числа в интервале от 0 до $n - 1$, т. е. $m \in Z_n$. На рис. 3.1 представлена схема алгоритма RSA.

Рис. 3.1. Схема алгоритма RSA

Алгоритм Отправителя:

1. Взять открытый ключ $\{e, n\}$ получателя.

2. Взять открытый текст m.

3. Зашифровать сообщение с использованием открытого ключа получателя:

$$c = E(m) = m^e \bmod n. \qquad (3.1)$$

Алгоритм Получателя:

1. Принять зашифрованное сообщение c.

2. Взять свой закрытый ключ $\{d, n\}$.

3. Применить закрытый ключ для расшифрования сообщения:

$$m = D(m) = c^d \bmod n. \qquad (3.2)$$

Уравнения (3.1) и (3.2), на которых основана схема RSA, определяют взаимно обратные преобразования множества.

Пример использования. На рис. 3.2 представлен пример использования алгоритма RSA. Отправитель отправил зашифрованное со-

общение «111111», и получатель, используя свой закрытый ключ, расшифровал его.

Этап	Описание операции	Результат операции
Генерация ключей	Выбрать два простых числа	$p = 3557$ $q = 2579$
	Вычислить модуль	$n = p \cdot q = 3557 \cdot 2579 = 9173503$
	Вычислить функцию Эйлера	$\varphi(n) = (p-1)(q-1) = 9167368$
	Выбрать открытую экспоненту	$e = 3$
	Вычислить секретную экспоненту	$d = e^{-1} \mod \varphi(n)$ $d = 6111579$
	Опубликовать "открытый ключ"	$\{e, n\} = \{3, 9173503\}$
	Сохранить "закрытый ключ"	$\{d, n\} = \{6111579, 9173503\}$
Шифрование	Выбрать текст для зашифровки	$m = 111111$
	Вычислить шифротекст	$c = E(m)$ $= m^e \mod n$ $= 111111^3 \mod 9173503$ $= 4051753$
Расшифрование	Вычислить исходное сообщение	$m = D(c) =$ $= c^d \mod n$ $= 4051753^{6111579} \mod 9173503$ $= 111111$

Рис. 3.2. Схема алгоритма RSA

3.2.2. Криптостойкость алгоритма

Зная n, e, M, C, (mod n), найти d.

Можно определить четыре возможных подхода для криптоанализа алгоритма RSA:

1. Метод «грубой силы»: перебор всех возможных закрытых ключей.

2. Факторизация (разложение числа n на два простых сомножителя). Это дает возможность вычислить

$$\Phi(n) = (p-1)(q-1) \text{ и } d = e^{-1}(\mathrm{mod}(n)).$$

3. Определить $\Phi(n)$ непосредственно, без начального определения p и q. Это также даст возможность определить

$$d = e^{-1} (\mathrm{mod}\, \Phi(n)).$$

4. Определить d непосредственно, без начального определения $\Phi(n)$.

Контрмера против атаки методом «грубой силы» стандартная: использовать большие ключи. Однако при этом надо учитывать, что сложность вычислений таких функций не является линейной от количества битов ключа, а возрастает быстрее, чем ключ. Таким образом, размер ключа должен быть достаточно большим, чтобы сделать лобовую атаку непрактичной, и достаточно маленьким для возможности практического шифрования.

Сложность остальных атак (при достаточно больших n) эквивалентна сложности задачи факторизации, которая в настоящее время относится к классу вычислительно сложных (не имеет эффективного алгоритма решения).

Задача факторизации имеет большую историю, однако работы в этой области активизировались с появлением криптографии с открытым ключом.

В 2003 г. были разработаны СБИС для вскрытия криптосистемы RSA. Стоимость около $10 млн. С их использованием факторизация числа длиной 512 бит выполняется за 10 мин., а длиной 1024 бита –менее, чем за 1 год. Поэтому рекомендуемая длина модуля n – не менее 2048 бит. Для того чтобы избежать выбора значения n, которое могло бы легко раскладываться на сомножители, на p и q должно быть наложено много дополнительных ограничений, например, p и q должны не сильно отличаться по длине.

3.3. Криптография на основе эллиптических кривых

Работать с большими числами очень непросто, однако это обеспечивает безопасность ключей. Хотелось бы сохранить тот же уровень безопасности, но операции выполнять проще.

Преимущество перед RSA или криптосистемами Диффи–Хеллмана – обеспечивается эквивалентная защита при меньшей длине ключа.

Для перехода к новому алгоритму шифрования необходимо обосновать переход от работы с числами к работе с точками эллиптической кривой:

– определить множество элементов;
– ввести операции над ними;
– описать способ их применения.

Эллиптическая кривая – гладкая кривая на плоскости (рис. 3.3):

$$y^2 = x^3 + ax + b.$$

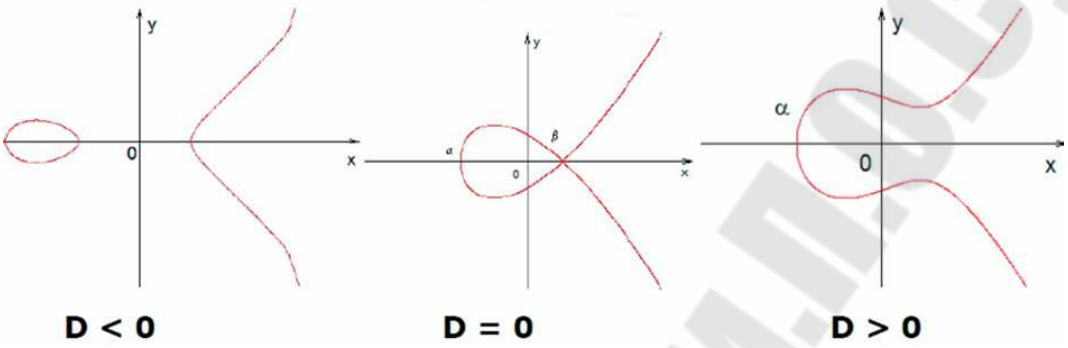

Рис. 3.3. Пример эллиптической кривой
с различным значением дискриминанта

3.3.1. Алгоритм шифрования

Перед описанием алгоритма шифрования необходимо:
– обосновать переход от работы с числами к работе с точками эллиптической кривой;
– определить множество элементов;
– ввести операции над ними;
– описать способ их применения.

Свойства эллиптических кривых:
– любая вертикальная (параллельная оси Oy) прямая не пересекает кривую ни разу или пересекает ее дважды;
– любая другая прямая пересекает кривую один или три раза;
– условимся точку касания считать за два пересечения.

Сложение точек уравнения эллептической кривой делается следующим образом:
– точка 0 – бесконечность;
– $p + (-p) = 0$;
– точка «касания» – берется дважды (сложить с собой);
– сложить две точки P и Q, провести через них прямую, пересечение в третьей точке R: $P + Q + R = 0$;

– отобразить третью точку относительно оси X (обратный элемент): $P + Q = -R$ (рис. 3.4).

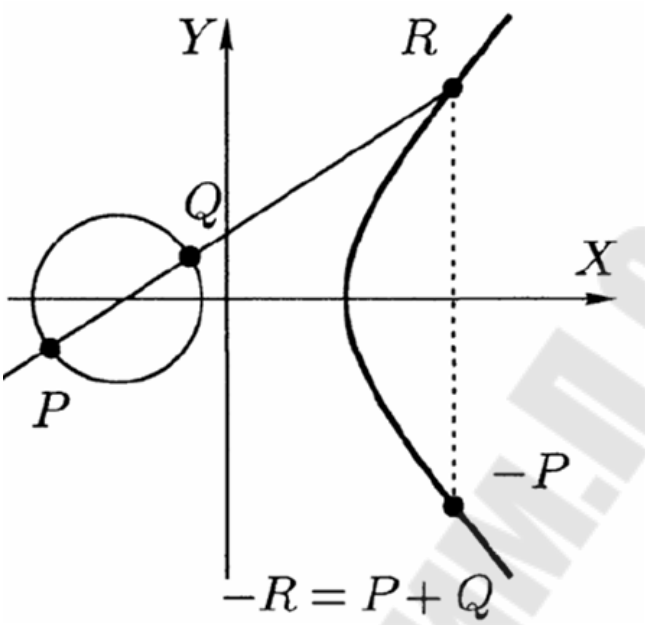

Рис. 3.4. Сложение точек P и Q

На рис. 3.5 и 3.6 показано сложение точек (когда они равны друг другу) и удвоение точки эллептической кривой.

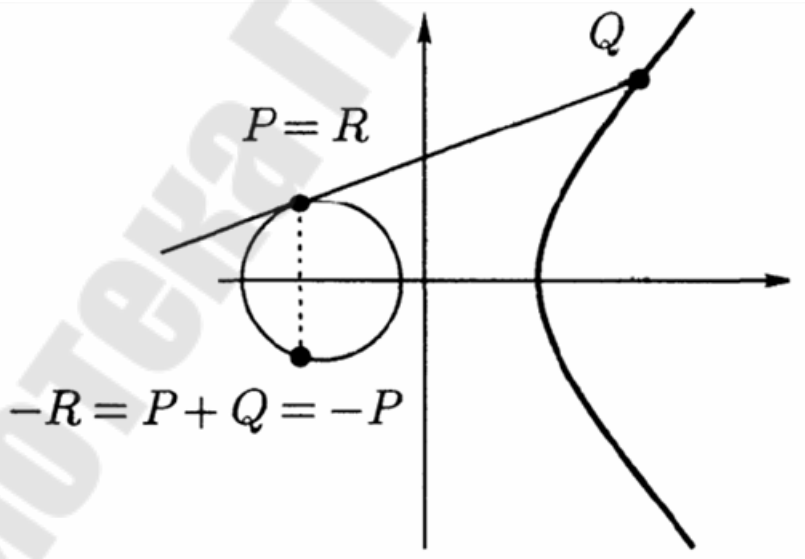

Рис. 3.5. Сложение точек при $P = R$

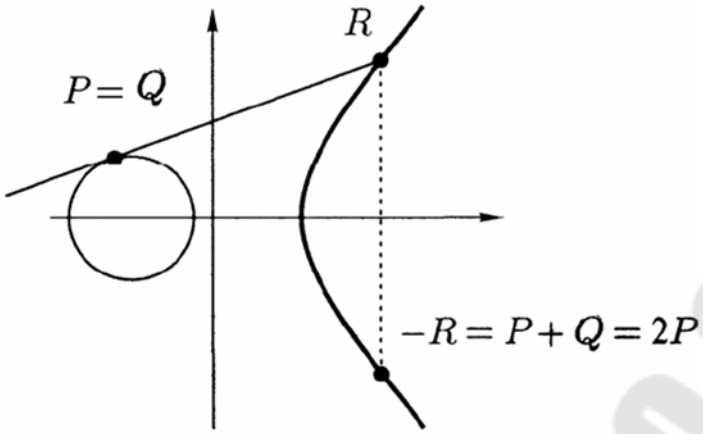

Рис. 3.6. Удвоение точки *P*

Множество точек эллиптической кривой – коммутативная группа. Введем 0 – фиктивную «бесконечную точку» $P + 0 = 0 + P = P$, тогда:

– 0 – «нейтральный элемент» группы;

– $P + (-P) = 0$ (обратный элемент);

– $P + Q + R = 0$ (для остальных);

– $(P + Q) + R = P + (Q + R)$ (ассоциативность);

– $P + Q = Q + P$ (коммутативность).

Суть применения эллиптических кривых в криптографии. Группа чисел по простому модулю (как в RSA или криптосистеме Диффи–Хеллмана) заменяется группой решений уравнения $y^2 = x^3 + ax + b$ в поле *GF*(*p*).

Все значения вычисляются по модулю *p*, где *p* – простое число. Элементами данной эллиптической кривой являются пары неотрицательных целых чисел, которые удовлетворяют частному виду эллиптической кривой:

$$y^2 = x^3 + ax + b(\mathrm{mod}\, p).$$

Такую кривую обозначают $E_p(a, b)$. При этом числа *a* и *b* должны быть меньше *p* и удовлетворять условию

$$Y = 4a^3 + 27b^2(\mathrm{mod}\, p) \neq 0.$$

Вычисление множества точек на эллиптической кривой:

– для каждого *x*, где $0 \le x \le p$, вычисляется

$$Y = x^3 + ax + b(\mathrm{mod}\, p);$$

– для Y выясняется, имеет ли оно квадратный корень по модулю p;

– если «нет», то в $E_p(a, b)$ нет точек для такого x;

– если корень существует, то имеется два значения y, соответствующих операции извлечения квадратного корня (исключением является случай, когда единственным значением оказывается $y = 0$);

– эти значения (x, y) и будут точками $E_p(a, b)$.

Эллиптический аналог криптосистемы Диффи–Хеллмана:

– Алиса и Боб выбирают (и сообщают всем) эллиптическую кривую, поле и точку B, лежащую на этой кривой;

– затем Алиса секретно выбирает число a и подсчитывает точку aB (складывает B с самой собой a раз), а Боб выбирает число b и вычисляет bB;

– затем они обмениваются полученными результатами, и их общим секретным ключом становится точка abB.

3.3.2. Криптостойкость алгоритма

Преимущество шифров, основанных на эллиптических кривых, заключается в том, что в них можно использовать меньшие по величине простые числа, чем в классических системах с открытым ключом.

• *Ограниченность числа рабочих схем*. В отличие от алгоритмов классической криптографии, которые могут быть созданы в неограниченном количестве путем комбинирования различных элементарных преобразований, каждая «современная» схема базируется на определенной «нерешаемой» задаче. Как следствие, количество рабочих схем криптографии с открытым ключом весьма невелико.

• *Постоянная «инфляция» размера блоков данных и ключей, обусловленная прогрессом математики и вычислительной техники.* Так, если в момент создания криптосистемы RSA считался достаточным размер чисел в 512 бит, то сейчас рекомендуется не менее 4 Кбит. Иными словами, «безопасный» размер чисел в RSA вырос практически на порядок; похожая картина наблюдается и для других схем, тогда как в традиционной криптографии этот размер увеличился всего вдвое.

• *Потенциальная ненадежность базиса.* В настоящее время теорией вычислительной сложности исследуется вопрос о возможности решения задач данного типа за полиномиальное время. В рамках теории уже доказана связь большинства используемых вычислительно сложных задач с другими аналогичными задачами.

• *Отсутствие дальней перспективы.* Уже известен предполагаемый «могильщик» современной криптографии – это квантовые вычисления, с помощью которых оказалось возможным решать многие задачи гораздо быстрее, чем на традиционных компьютерах. Правда, в настоящее время они существуют лишь в теории, из практических достижений можно отметить только успешную факторизацию числа 15 «микромакетом» квантового вычислителя. Специалисты полагают, что «серьезные» квантовые компьютеры появятся примерно через 25–30 лет, и за пределами этого срока будущее современной криптографии туманно. Из всего вышеизложенного следует, что для современной криптографии актуальна проблема повышения стойкости и уменьшения размера блоков данных путем модификации уже существующих криптосистем.

3.4. Криптографические хеш-функции

Функция хеширования $H(m)$, или хеш-функция (*hash-function*) – это детерминированная функция, на вход которой подается строка битов произвольной длины, а выходом всегда является битовая строка фиксированной длины n (рис. 3.7).

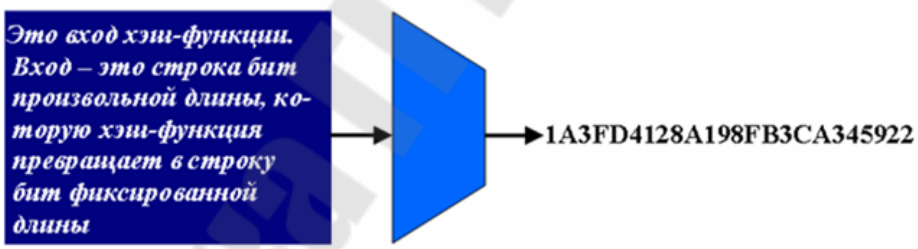

Рис. 3.7. Пример хеш-функции

Значение хеш-функции $H(m)$ для входа m называют хешем. Исходная строка m, для которой вычислено хеш-значение, называется прообразом хеш-функции.

Свойства, которые должны быть присущи криптографическим хеш-функциям:

1. Стойкость к поиску первого прообраза – отсутствие эффективного полиномиального алгоритма вычисления обратной функции, т. е. нельзя восстановить текст m по известной его свертке $H(m)$ за реальное время (необратимость). Это свойство эквивалентно тому, что хеш-функция является односторонней функцией.

2. Стойкость к поиску второго прообраза (коллизиям первого рода) – вычислительно невозможно, зная сообщение m и его свертку $H(m)$, найти такое другое сообщение $m' \neq m$, чтобы $H(m) = H(m')$.

3. Стойкость к коллизиям (коллизиям второго рода). Коллизией для хеш-функции называется такая пара значений m и m', $m' \neq m$, для которой $H(m) = H(m')$. Так как количество возможных открытых текстов больше числа возможных значений свертки, то для некоторой свертки найдется много прообразов ⇒ коллизии для хеш-функций обязательно существуют. Например, пусть длина хеш-прообраза 6 битов, длина свертки 4 бита. Тогда число различных сверток – 24 = 16, а число хеш-прообразов – 26 = 64, т. е. в 4 раза больше ⇒ хотя бы одна свертка из всех соответствует 4 прообразам. Стойкость хеш-функции к коллизиям означает, что нет эффективного полиномиального алгоритма, позволяющего находить коллизии.

Существует два вида криптографических функций (рис. 3.8).

Рис. 3.8. Типы криптографической хеш-функции

Разработка хеш-функций, удовлетворяющих всем требованиям, сложная задача. Практически отвечают этим требованиям хеш-функции из группы алгоритмов MD и SHA, характеристики которых приведены в табл. 3.1.

Таблица 3.1

Характеристики основных криптографических хеш-функций

Алгоритм	Описание	Длина хеша, бит	Примечание
MD2	Односторонняя функция	128	Медленнее, чем MD4 и MD5

Окончание табл. 3.1

Алгоритм	Описание	Длина хеша, бит	Примечание
MD4	Односторонняя функция	128	
MD5	Односторонняя функция	128	Сложнее, но медленнее, чем MD4
HAVAL	Односторонняя функция	Длина различна	Модификация MD5, но более стойкая к атакам
SHA	Односторонняя функция	160	Используется в DSA
SHA-1, SHA-256, -384, -512	Обновленная версия SHA	SHA-1 создает хеш длиной 160 бит, SHA-256 – длиной 256 бит и т. д.	–

Алгоритмы группы MD разработаны Роном Ривестом [4]. Название обозначает «дайджест сообщения». Алгоритм безопасного хеширования (SHA – Secure Hash Algorithm) – стандарт, разработанный NIST и принятый как федеральный стандарт обработки информации в США. SHA основан на схеме Меркеля–Дамгарда.

Рассмотрим кратко основные криптографические алгоритмы хеширования.

MD5 (рис. 3.9). Алгоритм хеширования следующий:

– MD5 выдает 128-битовое хеш-значение;

– в MD5 четыре этапа (по 16 операций в каждом);

– в каждом действии используется уникальная прибавляемая константа $K_i = 2^{32}|\sin(i)|$;

– каждое действие добавляется к результату предыдущего этапа, что обеспечивает быстрый лавинный эффект;

– в MD5 используется четыре переменных A, B, C и D;

– значения циклического сдвига влево на каждом этапе были приближенно оптимизированы для ускорения лавинного эффекта. Четыре сдвига, используемые на каждом этапе, отличаются от значений, используемых на других этапах.

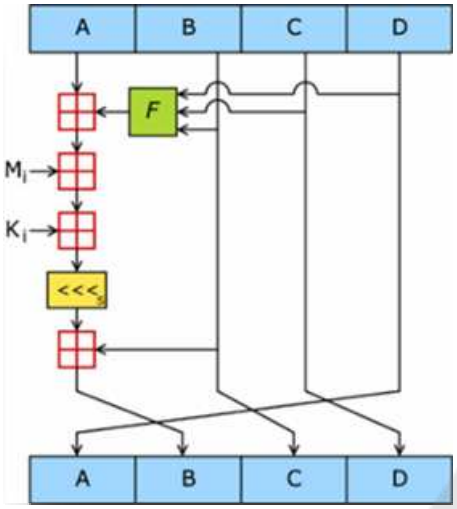

Рис. 3.9. Алгоритм хеширования MD5

SHA (рис. 3.10). Алгоритм хеширования следующий:

– SHA выдает 160-битовое хеш-значение;

– в SHA четыре этапа. На четвертом этапе используется та же нелинейная функция *f*, что и на втором;

– для каждого этапа используется своя константа K_t;

– на третьем этапе в SHA используется версия функции из MD4;

– в SHA пять переменных *A*, *B*, *C*, *D* и *E*.

SHA на каждом этапе использует постоянное значение сдвига.

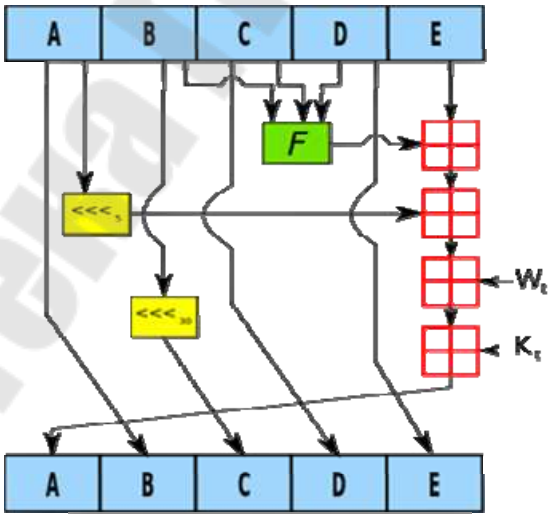

Рис. 3.10. Алгоритм хеширования SHA

ГОСТ Р 34.11–94. Алгоритм хеширования следующий:

– обрабатывается блок длиной 256 бит, и выходное значение тоже имеет длину 256 бит;

– определяет контрольную сумму, рассчитанную по всем блокам исходного сообщения, которая является частью финального вычисления хеша, что несколько затрудняет коллизионную атаку;

– применены меры борьбы против поиска коллизий, основанные на неполноте последнего блока;

– обработка блоков происходит по алгоритму шифрования ГОСТ 28147–89, который содержит преобразования на *S*-блоках, что существенно осложняет применение метода дифференциального криптоанализа к поиску коллизий.

3.5. Электронная цифровая подпись

Электронная цифровая подпись для электронных документов играет ту же роль, что и подпись, поставленная от руки в документах на бумаге: это данные, присоединяемые к передаваемому сообщению, подтверждают, что владелец подписи составил или заверил это сообщение. Получатель сообщения с помощью цифровой подписи может проверить, что автором сообщения является именно владелец подписи и что в процессе передачи не была нарушена целостность полученных данных. При разработке механизма цифровой подписи возникают следующие задачи:

– формирование подписи таким образом, чтобы ее невозможно было подделать;

– обеспечение возможности проверки того, что подпись действительно принадлежит указанному субъекту;

– предотвращение отказа субъекта от своей подписи.

3.5.1. Классическая схема создания цифровой подписи

При создании цифровой подписи по классической схеме отправитель должен выполнить следующие действия:

Вычислить хеш-образ m исходного сообщения M при помощи хеш-функции h.

Вычислить цифровую подпись S по хеш-образу сообщения с использованием секретного ключа K_c создания подписи.

Сформировать новое сообщение (M, S), состоящее из исходного сообщения и добавленной к нему цифровой подписи.

Получив подписанное сообщение (M', S), получатель должен выполнить следующие действия (принятое сообщение обозначено как M'

по причине того, что оно могло быть преднамеренно либо случайно искажено в процессе передачи по каналу связи и может не совпадать с отправленным).

Вычислить хеш-образ m' сообщения M' при помощи хеш-функции h.

С использованием открытого ключа проверки подписи (K_o) извлечь хеш-образ m сообщения из цифровой подписи S.

Сравнить вычисленное значение m' с извлеченным из цифровой подписи значением хеш-образа m. Если хеш-образы совпадают, то подпись признается подлинной.

3.5.2. Электронная цифровая подпись RSA

Первой и наиболее известной во всем мире конкретной системой электронной цифровой подписи стала система RSA, математическая схема которой была разработана в 1977 г. в Массачусетском технологическом институте США. Сначала необходимо вычислить пару ключей (секретный ключ и открытый ключ). Для этого отправитель сообщения (документа) выбирает два больших простых числа p и q, а затем находит их произведение $r = p \cdot q$ и значение функции Эйлера от данного произведения $\varphi(r) = (p-1)(q-1)$. Далее отправитель вычисляет значение K_o из условий: $K_o < \varphi(r)$, $\text{НОД}(K_o, \varphi(r)) = 1$ и значение K_c из условий: $K_c < \varphi(r)$, $K_o K_c = 1 \bmod \varphi(r)$.

Пара значений (K_o, r) является открытым ключом. Эту пару чисел автор передает партнерам по переписке для проверки его цифровых подписей. Значение K_c сохраняется автором как секретный ключ подписи.

Обобщенная схема формирования и проверки цифровой подписи RSA показана на рис. 3.11.

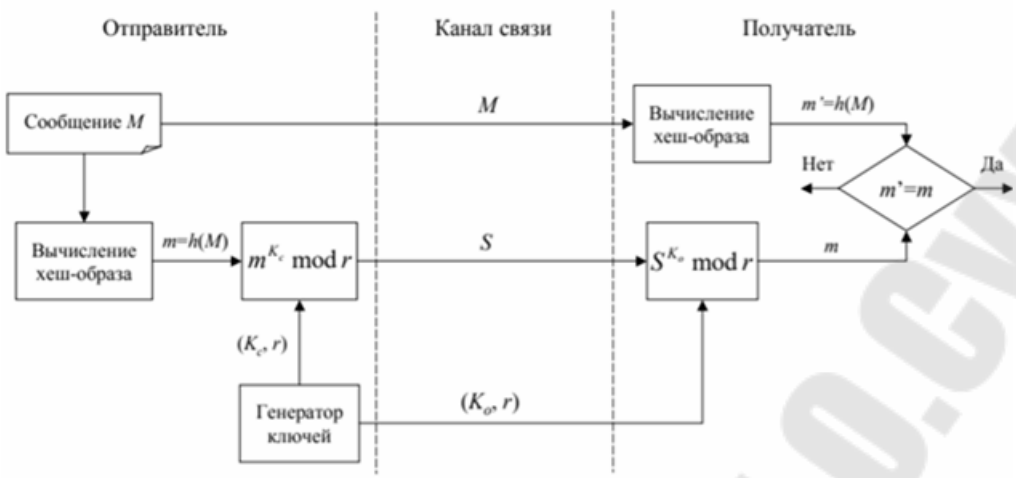

Рис. 3.11. Обобщенная схема цифровой подписи RSA

Допустим, что отправитель хочет подписать сообщение M перед его отправкой. Сначала сообщение M сжимают с помощью хеш-функции h в целое число m: $m = h(M)$. Затем вычисляют цифровую подпись S под электронным документом M, на основе хеш-образа m и секретного значения K_c: $S = m^{K_c} \bmod r$.

Для возведения в степень можно воспользоваться алгоритмом быстрого возведения в степень по модулю, позволяющему вычислить $x = a^z \bmod n$. Пара (M, S) передается получателю как электронный документ M, подписанный цифровой подписью S, причем подпись S сформирована обладателем секретного ключа K_c.

После приема пары (M', S) получатель вычисляет хеш-образ сообщения M' двумя различными способами. Прежде всего, он восстанавливает хеш-образ m, применяя криптографическое преобразование подписи S с использованием открытого ключа K_o:

$$m = S^{K_o} \bmod r.$$

Кроме того, он находит результат хеширования m' принятого сообщения M' с помощью такой же хеш-функции h: $m' = h\,(M)$. Если вычисленные значения совпадают, т. е. $S^{K_o} \bmod r = h(M')$, то получатель признает пару (M', S) подлинной. Фальсификация сообщения при его передаче по каналу связи возможна только при получении злоумышленником секретного ключа K_c либо за счет проведения успешной атаки против хеш-функции. При использовании достаточно больших значений p и q определение секретного значения K_c по открытому ключу (K_o, r) является чрезвычайно трудной задачей, соот-

ветствующей по сложности разложению модуля r на множители. Используемые в реальных приложениях хеш-функции обладают характеристиками, делающими атаку против цифровой подписи практически не осуществимой. Пример – хеш-функция SHA-1, принятая в США в качестве стандарта в 1995 г., формирующая 160-битовый хеш-образ при обработке сообщения блоками по 512 бит. Вероятность коллизии при использовании данной хеш-функции составляет 2^{-160}, или приблизительно $6{,}84 \cdot 10^{-49}$.

3.5.3. Электронная цифровая подпись Эль-Гамаля

Идея Эль-Гамаля – использование для электронной цифровой подписи задачи дискретного логарифмирования, вычислительно более сложной, чем факторизация чисел.

Пусть пользователь хочет подписать сообщение M и отправить его пользователю B.

Схема алгоритма

Генерация ключей:

– необходимо выбрать большое простое целое число p (порядка $\sim 10^{308}$ или $\sim 2^{1024}$) и какой-либо первообразный корень g по модулю p (размер $g \sim 10^{154}$ или $\sim 2^{512}$);

– отправитель A выбирает случайное целое число x из интервала $(1, p-1)$ и вычисляет $y = g^x \bmod p$.

Числа p и g – параметры домена (т. е. открытые параметры системы, используемые одновременно многими пользователями).

Открытый ключ отправителя – тройка чисел (p, g, y), передается всем потенциальным получателям документов. Закрытый ключ отправителя – число x держится в секрете.

Формирование подписи отправителем A:

– вычисляется хеш сообщения $H = H(M)$. При этом должно выполняться условие $1 < H(M) < p-1$;

– выбирается случайное число r (это рандомизатор, держится в секрете) из интервала $(1, p-1)$, взаимно простое с $p-1$, и вычисляется

$$a = g^r \pmod p.$$

Находится число b из уравнения $H = ax + rb \pmod{p-1}$, т. е.

$$b = (H - ax) r^{-1} \pmod{p-1};$$

– подписью сообщения M является пара (a, b). Получателю отправляется тройка (M, a, b).

Проверка подписи получателем B. Зная открытый ключ (p, g, y), подпись (a, b) сообщения M проверяется следующим образом:

– проверяется выполнимость условий: $0 < a < p$ и $0 < b < p - 1$;

– если хотя бы одно из них не выполнено, то подпись – незаконна;

– вычисляется хеш сообщения $H = H(M)$;

– вычисляется $y^a a^b \pmod p$ и $g^H \pmod p$;

– подпись принимается только при условии, что

$$y^a a^b \equiv g^H \pmod p.$$

3.5.4. Электронная цифровая подпись ГОСТ Р34.10–94

Российский стандарт электронной цифровой подписи разрабатывался позже первоначального варианта – американского, поэтому параметры этого алгоритма выбраны с учетом возросших возможностей потенциального противника по вскрытию криптосистем. В частности, увеличена длина значения хеш-функции, что снижает вероятность столкновений, и, соответственно, порядок элемента-генератора, что делает более сложным решение задачи дискретного логарифма для восстановления секретного ключа.

В этом алгоритме используются следующие переменные:

– p – простое число, $2^{509} < p < 2^{512}$ или $2^{1020} < p < 2^{1024}$;

– q – простое число, $2^{254} < q < 2^{256}$ и q является делителем для $(p-1)$;

– a – целое число, $1 < a < p - 1$, при этом

$$a^q \bmod p = 1 a^q \bmod p;$$

– x – секретный ключ пользователя, для формирования подписи $0 < x < q$;

– y – открытый ключ пользователя для проверки пользователя

$$y = a^x \bmod p.$$

Процедура подписи ГОСТ Р34.10–94 заключается в выполнении следующих действий:

1. Вычислить значение хеш-функции $h(M)$.

2. Выработать простое число $k, 0 < k < q$.

3. Вычислить значения:

$$r = \left(a^k \bmod p\right) \bmod q.$$

Если $r = 0$, перейти к этапу 2 и выбрать другое значение k.

С использованием секретного ключа x вычислить:

$$s = \left(k \cdot h(M) + x \cdot r\right) \bmod q.$$

Если $s = 0$, перейти к этапу 2, в противном случае закончить работу алгоритма

$$(r, s) \text{ – электронная цифровая подпись.}$$

Процедура проверки подписи:

– проверить условие $0 < s < q$ и $0 < r < q$. Если хотя бы одно условие не выполнено, то подпись считается недействительной;

– вычислить значение хеш-функции от полученного сообщения $h(M)$;

– вычислить значение:

$$w = h(M)^{-1} \bmod q;$$

– вычислить значения:

$$u_1 = s \cdot \upsilon \bmod q;$$

$$u_2 = (q - r)\upsilon \bmod q;$$

– вычислить значение:

$$\upsilon = \left(a^{u_1} \cdot y^{u_2} \bmod p\right) \bmod q;$$

– проверить условие:

$$r = \upsilon.$$

3.6. Квантовая криптография

Квантовые компьютеры и связанные с ними технологии в последнее время становятся все актуальнее. Исследования в этой области не прекращаются вот уже десятилетия, и ряд революционных достижений налицо. Квантовая криптография – одно из них.

3.6.1. История квантовой криптографии

Идея использовать квантовые объекты для защиты информации от подделки и несанкционированного доступа впервые была высказана Стефаном Вейснером в 1970 г. В 1984 г. Чарльз Беннет из IBM и Жиль Брассар из Монреальского университета, которые были знакомы с работами Вейснера, предположили, что фотоны могут быть использованы в криптографии для получения фундаментально защищенного канала. Для представления нулей и единиц они решили взять фотоны, поляризованные в различных направлениях, и предложили простую схему квантового распределения ключей шифрования, названную ими BB84. В 1989 г. Беннет и Брассар в Исследовательском центре IBM построили первую работающую квантово-криптографическую систему. Она состояла из квантового канала, содержащего передатчики на концах (традиционно называемые передатчиками Боба и Алисы), размещенные на оптической скамье длиной около метра в светонепроницаемом полутораметровом кожухе размером 0,5 × 0,5 м. Собственно квантовый канал представлял собой свободный воздушный канал длиной около 32 см. Макет управлялся от персонального компьютера, который содержал программное представление пользователей Алисы и Боба, а также злоумышленника. Позднее, в 1991 г., идея была развита Экертом.

Первая презентация коммерческой системы квантовой криптографии произошла на выставке CeBIT–2002. Там швейцарские инженеры компании Gap-Optique из Женевского университета представили первую систему квантового распределения ключей (QKD – Quantum Key Distribution). GAP-Optique под руководством Николаса Гисина удалось передать ключ на расстояние 67 км из Женевы в Лозанну с помощью почти промышленного образца аппаратуры. В последующие годы к проектированию и изготовлению систем квантовой криптографии подключились такие коммерческие монстры как Toshiba, NEC, IBM, HP, Mitsubishi, NTT. Но наряду с ними стали появляться на рынке и маленькие, но высотехнологичные компании: MagiQ, ID Quantique (поддерживается министерством обороны Великобритании), Smart Quantum. Это говорит об уже достаточно широком коммерческом применении квантовой криптографии.

В основе метода квантовой криптографии лежит наблюдение квантовых состояний фотонов. Отправитель задает эти состояния, а получатель их регистрирует. Здесь используется квантовый принцип

неопределенности Гейзенберга, когда две квантовые величины не могут быть измерены одновременно с требуемой точностью. Таким образом, если отправитель и получатель не договорились между собой, какой вид поляризации квантов брать за основу, получатель может разрушить посланный отправителем сигнал, не получив никакой полезной информации. Эти особенности поведения квантовых объектов легли в основу протокола квантового распространения ключа.

3.6.2. Схема BB84

Схема BB84 работает следующим образом. Сначала Алиса генерирует и посылает Бобу последовательность фотонов, поляризация которых выбрана случайным образом и может составлять 0°, 45°, 90° и 135°. Боб принимает эти фотоны и для каждого из них случайным образом решает, замерять его поляризацию как перпендикулярную или диагональную. Затем по окрытому каналу Боб объявляет для каждого фотона, какой тип измерений им был сделан (перпендикулярный или диагональный), но не сообщает результат этих измерений, например 0°, 45°, 90° или 135°. По этому же окрытому каналу Алиса сообщает ему, правильный ли вид измерений был выбран для каждого фотона. Затем Алиса и Боб отбрасывают все случаи, когда Боб сделал неправильные замеры. Если квантовый канал не перехватывался, оставшиеся виды поляризации и будут поделенной между Алисой и Бобом секретной информацией, или ключом. Этот этап работы квантово-криптографической системы называется первичной квантовой передачей.

Алиса посылает фотоны, имеющие одну из четырех возможных поляризаций, которую она выбирает случайным образом.

Для каждого фотона Боб выбирает случайным образом тип измерения: он изменяет либо прямолинейную поляризацию (+), либо диагональную (*x*).

Боб записывает результаты изменения и сохраняет в тайне.

Боб открыто объявляет, какого типа измерения он проводил, а Алиса сообщает ему, какие измерения были правильными.

Алиса и Боб сохраняют все данные, полученные в тех случаях, когда Боб применял правильное измерение. Эти данные затем переводятся в биты (0 и 1), последовательность которых и является результатом первичной квантовой передачи.

Принципы первичной квантовой передачи. Рассматривается простой пример создания общего секретного ключа в квантово-криптографической системе.

Следующим важным этапом является оценка попыток перехвата информации в квантово-криптографическом канале связи. Это может производиться Алисой и Бобом по открытому каналу путем сравнения и отбрасывания случайно выбранных ими подмножеств полученных данных. Если такое сравнение выявит наличие перехвата, Алиса и Боб отбрасывают все свои данные и начинают повторное выполнение первичной квантовой передачи. В противном случае они оставляют прежнюю поляризацию, принимая фотоны с горизонтальной или 45°-й поляризацией за двоичный «0», а с вертикальной или 135°-й поляризацией – за двоичную «1». Согласно принципу неопределенности, злоумышленник не может замерить как прямоугольную, так и диагональную поляризацию одного и того же фотона. Даже если он для какого-либо фотона произведет измерение и перешлет Бобу этот фотон в соответствии с результатом своих измерений, то в итоге количество ошибок намного увеличится, и это станет заметно Алисе. Это приведет к стопроцентной уверенности Алисы и Боба в состоявшемся перехвате фотонов.

Более эффективной проверкой для Алисы и Боба является проверка на четность, осуществляемая по открытому каналу. Например, Алиса может сообщить: «Я просмотрела 1-й, 4-й, 4-й, 8-й... и 998-й из моих 1000 бит, и они содержат четное число единиц». Тогда Боб подсчитывает число «1» на тех же самых позициях. Можно показать, что, если данные у Боба и Алисы отличаются, проверка на четность случайного подмножества этих данных выявит количество ошибок. Достаточно повторить такой тест 20 раз с 20 различными случайными подмножествами, чтобы вычислить процент ошибок. Если ошибок слишком много, то считается, что производился перехват в квантово-криптографической системе.

3.6.3. Современное состояние и проблемы квантовой криптографии

Активные исследования в области квантовой криптографии ведут IBM, GAP-Optique, Mitsubishi, Toshiba, Национальная лаборатория в Лос-Аламосе, Калифорнийский технологический институт, молодая компания MagiQ и холдинг QinetiQ, поддерживаемый британским министерством обороны.

Рекордное расстояние, на которое был передан квантовый ключ, составляет 250 км. Эта передача была осуществлена группой специа-

листов из Университета Женевы (Швейцария) и компанией Corning Incorporated в 2009 г.

Квантовая криптография как сегмент рынка только начинает формироваться, и здесь пока на равных могут играть и мировые компьютерные корпорации, и небольшие начинающие компании. Например, компанией ID Quantique создано несколько промышленных систем распределения квантовых ключей. Интерес к квантовой криптографии со стороны коммерческих и военных организаций растет, так как эта технология может гарантировать абсолютную защиту. На сегодняшний день квантовая криптография доступна для коммерческого применения уже несколько лет. Но технология практична лишь в руках организаций государственного масштаба и крупного частного сектора, которые в состоянии позволить себе иметь собственные оптоволоконные сети.

Но кроме успешного создания и ввода в действие систем распределения квантовых ключей, есть и успешные эксперименты по их взлому. Например, в 2007 г. физики из Торонтского университета (Канада) провели экспериментальную демонстрацию необнаруживаемого перехвата сообщений в системе распределения квантовых ключей, реализованной швейцарской компанией ID Quantique. Представители ID Quantique, впрочем, не видят в этом результате ничего сенсационного. «Он важен и интересен, но только как доказательство того, что системы квантовой криптографии не отличаются от других технологий обеспечения защиты: их тоже необходимо тестировать, – поясняет инженер компании Грегори Риборди – ID-500, изготовленная в 2004 г., уже устарела, к тому же она не была рассчитана на промышленное применение». В современных установках ID Quantique, поставляемых на рынок, «безопасный» уровень ошибок снижен до 8 %.

В 2010 г. ученые из Норвегии и Германии нашли сравнительно простой и не слишком затратный способ перехвата сообщений в системах распределения квантовых ключей. «Слабым звеном» в них оказался лавинный диод, который используется для регистрации отдельных фотонов. Незаметно перехватывать сообщения ученым удалось при переводе лавинного диода в линейный режим. Они решили эту задачу довольно простым способом: направили на детекторы излучение лазера, работающего в непрерывном режиме с мощностью около 1 мВт. Это позволило взломать системы Clavis2 и QPN 5505, поставляемые ID Quantique и MagiQ Technologies. Представителей компаний, разумеется, уведомили о том, что их продукция оказалась нена-

дежна; в ID Quantique, по словам разработчиков, уже приняли ответные меры. «Этот способ атаки гораздо более практичен и опасен, чем все, с чем мы сталкивались прежде», – подтверждает сотрудник ID Quantique Грегори Риборди.

3.7. Практическое задание

Задание 1

1. Вычислить НОД (m, n) по алгоритму Евклида.
2. Вычислить НОД(m, n), используя бинарный алгоритм.
3. Вычислить функцию Эйлера для n =… (произвольное число).
4. Показать, что $a^b = 1 \bmod n$.
5. Вычислить взаимообратное число для m по модулю r.
6. Показать, что $m^n \bmod r = k$.

Варианты заданий

Вариант 1
1. $m = 347$, $n = 723$.
2. $m = 112$, $n = 679$.
3. $n = 324$.
4. $a = 4$, $b = 336$, $n = 377$.
5. $r = 29$, $m = 5$.
6. $m = 8, n = 7, r = 13, k = 5$.

Вариант 2
1. $m = 552$, $n = 874$.
2. $m = 1934$, $n = 725$.
3. $n = 863$.
4. $a = 6$, $b = 480$, $n = 527$.
5. $r = 23$, $m = 3$.
6. $m = 5, n = 8, r = 19, k = 4$.

Вариант 3
1. $m = 1236$, $n = 935$.
2. $m = 1778$, $n = 994$.
3. $n = 632$.
4. $a = 10$, $b = 220$, $n = 253$.
5. $r = 26$, $m = 5$.
6. $m = 7, n = 9, r = 17, k = 10$.

Вариант 4
1. $m = 845$, $n = 652$.
2. $m = 964$, $n = 1277$.
3. $n = 953$.
4. $a = 14$, $b = 448$, $n = 493$.
5. $r = 55$, $m = 6$.
6. $m = 9, n = 8, r = 13, k = 3$.

Вариант 5
1. $m = 1974$, $n = 528$.
2. $m = 998$, $n = 1285$.
3. $n = 746$.
4. $a = 13$, $b = 504$, $n = 551$.
5. $r = 91$, $m = 8$.
6. $m = 9, n = 7, r = 19, k = 4$.

Вариант 6
1. $m = 1532$, $n = 643$.
2. $m = 994$, $n = 778$.
3. $n = 488$.
4. $a = 13$, $b = 672$, $n = 731$.
5. $r = 13$, $m = 5$.
6. $m = 4, n = 11, r = 7, k = 13$.

Вариант 7
1. $m = 2674$, $n = 1699$.
2. $m = 2674$, $n = 1118$.
3. $n = 886$.
4. $a = 7$, $b = 264$, $n = 299$.
5. $r = 18$, $m = 7$.
6. $m = 9, n = 7, r = 17, k = 2$.

Вариант 8
1. $m = 933$, $n = 525$.
2. $m = 525$, $n = 1385$.
3. $n = 724$.
4. $a = 12$, $b = 648$, $n = 703$.
5. $r = 26$, $m = 11$.
6. $m = 6, n = 12, r = 13, k = 1$.

Вариант 9
1. $m = 835$, $n = 1562$.
2. $m = 838$, $n = 1200$.
3. $n = 678$.
4. $a = 8$, $b = 360$, $n = 407$.
5. $r = 33$, $m = 5$.
6. $m = 8, n = 9, r = 19, k = 18$.

Вариант 10
1. $m = 2525$, $n = 1186$.
2. $m = 745$, $n = 1375$.
3. $n = 884$.
4. $a = 15$, $b = 756$, $n = 817$.
5. $r = 44$, $m = 7$.
6. $m = 5$, $n = 9$, $r = 17$, $k = 12$.

Вариант 11
1. $m = 472$, $n = 844$.
2. $m = 844$, $n = 1483$.
3. $n = 625$.
4. $a = 24$, $b = 936$, $n = 1007$.
5. $r = 28$, $m = 7$.
6. $m = 5$, $n = 8$, $r = 19$, $k = 4$.

Вариант 12
1. $m = 552$, $n = 938$.
2. $m = 938$, $n = 1366$.
3. $n = 728$.
4. $a = 9$, $b = 832$, $n = 901$.
5. $r = 32$, $m = 9$.
6. $m = 6$, $n = 9$, $r = 23$, $k = 16$.

Вариант 13
1. $m = 702$, $n = 1157$.
2. $m = 774$, $n = 1266$.
3. $n = 872$.
4. $a = 8$, $b = 624$, $n = 689$.
5. $r = 18$, $m = 5$.
6. $m = 8$, $n = 9$, $r = 17$, $k = 8$.

Вариант 14
1. $m = 1045$, $n = 836$.
2. $m = 686$, $n = 1078$.
3. $n = 842$.
4. $a = 5$, $b = 520$, $n = 583$.
5. $r = 25$, $m = 6$.
6. $m = 5$, $n = 7$, $r = 11$, $k = 3$.

Вариант 15
1. $m = 1265$, $n = 2024$.
2. $m = 1092$, $n = 689$.
3. $n = 634$.
4. $a = 6$, $b = 312$, $n = 371$.
5. $r = 16$, $m = 3$.
6. $m = 5$, $n = 9$, $r = 11$, $k = 9$.

Вариант 16
1. $m = 686$, $n = 1078$.
2. $m = 1045$, $n = 836$.
3. $n = 556$.
4. $a = 8$, $b = 624$, $n = 689$.
5. $r = 32$, $m = 9$.
6. $m = 8$, $n = 9$, $r = 19$, $k = 18$.

Вариант 17
1. $m = 1092$, $n = 689$.
2. $m = 1265$, $n = 2024$.
3. $n = 734$.
4. $a = 14$, $b = 448$, $n = 493$.
5. $r = 29$, $m = 5$.
6. $m = 5$, $n = 8$, $r = 19$, $k = 4$.

Вариант 18
1. $m = 938$, $n = 1366$.
2. $m = 552$, $n = 938$.
3. $n = 867$.
4. $a = 7$, $b = 264$, $n = 299$.
5. $r = 91$, $m = 8$.
6. $m = 9$, $n = 8$, $r = 13$, $k = 3$.

Вариант 19
1. $m = 994$, $n = 778$.
2. $m = 1532$, $n = 643$.
3. $n = 462$.
4. $a = 10$, $b = 220$, $n = 253$.
5. $r = 44$, $m = 7$.
6. $m = 4$, $n = 11$, $r = 17$, $k = 13$.

Вариант 20
2. $m = 112$, $n = 679$.
3. $m = 347$, $n = 723$.
4. $n = 532$.
5. $a = 15$, $b = 756$, $n = 817$.
6. $r = 18$, $m = 7$.
7. $m = 5$, $n = 9$, $r = 11$, $k = 9$.

Вариант 21
1. $m = 998$, $n = 1285$.
2. $m = 1974$, $n = 528$.
3. $n = 474$.
4. $a = 8$, $b = 624$, $n = 689$.
5. $r = 33$, $m = 5$.
6. $m = 6$, $n = 9$, $r = 23$, $k = 16$.

Вариант 22
1. $m = 1934$, $n = 725$.
2. $m = 552$, $n = 874$.
3. $n = 375$.
4. $a = 8$, $b = 624$, $n = 689$.
5. $r = 25$, $m = 6$.
6. $m = 5$, $n = 9$, $r = 11$, $k = 9$.

Вариант 23
1. $m = 1778$, $n = 994$.
2. $m = 1236$, $n = 935$.
3. $n = 552$.
4. $a = 5$, $b = 520$, $n = 583$.
5. $r = 18$, $m = 5$.
6. $m = 6$, $n = 9$, $r = 23$, $k = 16$.

Вариант 24
1. $m = 844$, $n = 1483$.
2. $m = 472$, $n = 844$.
3. $n = 423$.
4. $a = 24$, $b = 936$, $n = 1007$.
5. $r = 44$, $m = 7$.
6. $m = 8$, $n = 9$, $r = 19$, $k = 18$.

Вариант 25	Вариант 26	Вариант 27
1. $m = 1379$, $n = 254$.	1. $m = 844$, $n = 1483$.	1. $m = 2778$, $n = 1194$.
2. $m = 1116$, $n = 975$.	2. $m = 472$, $n = 844$.	2. $m = 2236$, $n = 1135$.
3. $n = 752$.	3. $n = 493$.	3. $n = 374$.
4. $a = 5$, $b = 520$, $n = 583$.	4. $a = 24$, $b = 936$, $n = 907$.	4. $a = 5$, $b = 510$, $n = 511$.
5. $r = 17$, $m = 14$.	5. $r = 31$, $m = 9$.	5. $r = 19$, $m = 11$.
6. $m = 6$, $n = 9$, $r = 23$, $k = 16$.	6. $m = 8$, $n = 9$, $r = 19$, $k = 18$.	6. $m = 9$, $n = 5$, $r = 23$, $k = 17$.

Вариант 28	Вариант 29	Вариант 30
1. $m = 2844$, $n = 1483$.	1. $m = 2778$, $n = 1994$.	1. $m = 844$, $n = 2483$.
2. $m = 3472$, $n = 1844$.	2. $m = 2236$, $n = 335$.	2. $m = 2472$, $n = 544$.
3. $n = 925$.	3. $n = 472$.	3. $n = 583$.
4. $a = 24$, $b = 936$, $n = 807$.	4. $a = 5$, $b = 520$, $n = 583$.	4. $a = 7$, $b = 936$, $n = 1007$.
5. $r = 47$, $m = 7$.	5. $r = 19$, $m = 7$.	5. $r = 43$, $m = 11$.
6. $m = 8$, $n = 7$, $r = 17$, $k = 18$.	6. $m = 6$, $n = 9$, $r = 23$, $k = 17$.	6. $m = 8$, $n = 5$, $r = 17$, $k = 18$.

Задание 2

1. Изучить теоретический материал по лабораторной работе.

2. Используя заданные в соответствии с вариантом значения p, q, выполнить шифрование по алгоритму RSA открытым ключом своей ФИО. Для представления букв в числовой форме использовать следующее соответствие: ‘А’ – 2, ‘Б’ – 3, ‘В’ – 4, ..., ‘Ё’ – 8, ..., ‘Я’ – 34, ‘_’ – 35.

3. Выполнить проверку правильности расшифрования полученных зашифрованных данных при помощи закрытого ключа.

Варианты заданий

1. $p = 5$, $q = 7$.	16. $p = 3$, $q = 17$.
2. $p = 17$, $q = 5$.	17. $p = 13$, $q = 3$.
3. $p = 11$, $q = 5$.	18. $p = 5$, $q = 13$.
4. $p = 7$, $q = 11$.	19. $p = 7$, $q = 13$.
5. $p = 7$, $q = 17$.	20. $p = 3$, $q = 19$.
6. $p = 3$, $q = 17$.	21. $p = 5$, $q = 7$.
7. $p = 13$, $q = 3$.	22. $p = 17$, $q = 5$.
8. $p = 5$, $q = 13$.	23. $p = 11$, $q = 5$.
9. $p = 7$, $q = 13$.	24. $p = 7$, $q = 11$.
10. $p = 3$, $q = 19$.	25. $p = 7$, $q = 17$.
11. $p = 5$, $q = 7$.	26. $p = 3$, $q = 17$.
12. $p = 17$, $q = 5$.	27. $p = 13$, $q = 3$.

13. $p = 11$, $q = 5$.
14. $p = 7$, $q = 11$.
15. $p = 7$, $q = 17$.
28. $p = 5$, $q = 13$.
29. $p = 7$, $q = 13$.
30. $p = 3$, $q = 19$.

Задание 3

1. Найти хеш-образ своей фамилии, используя хеш-функцию

$$H_i = (H_{i-1} + M_i)^2 \bmod n, \text{ где } n = p \cdot q.$$

2. Показать, как меняется хеш-образ при изменении одной из букв в фамилии (вычислить $h(M')$, где M' – фамилия с одной измененной буквой).

3. Показать (вычислить $h(M'')$), как меняется хеш-образ при перестановке любых двух букв в фамилии.

4. Используя полученный ранее хеш-образ, вычислить электронную цифровую подпись для своей фамилии по схеме RSA, Эль-Гамалая и ГОСТ Р34.10–94. При вычислении подписи использовать алгоритм быстрого возведения в степень по модулю.

Варианты заданий

1. $p = 13$, $q = 17$.
2. $p = 23$, $q = 13$.
3. $p = 19$, $q = 11$.
4. $p = 17$, $q = 23$.
5. $p = 19$, $q = 13$.
6. $p = 11$, $q = 29$.
7. $p = 19$, $q = 23$.
8. $p = 11$, $q = 23$.
9. $p = 11$, $q = 17$.
10. $p = 13$, $q = 29$.
11. $p = 17$, $q = 19$.
12. $p = 7$, $q = 23$.
13. $p = 7$, $q = 29$.
14. $p = 7$, $q = 19$.
15. $p = 7$, $q = 31$.
16. $p = 7$, $q = 37$.
17. $p = 5$, $q = 31$.
18. $p = 5$, $q = 37$.
19. $p = 5$, $q = 29$.
20. $p = 29$, $q = 11$.
21. $p = 23$, $q = 19$.
22. $p = 17$, $q = 13$.
23. $p = 19$, $q = 17$.
24. $p = 13$, $q = 23$.
25. $p = 23$, $q = 11$.
26. $p = 29$, $q = 13$.
27. $p = 23$, $q = 7$.
28. $p = 31$, $q = 7$.
29. $p = 29$, $q = 17$.
30. $p = 23$, $q = 29$.

Литература

1. Аверченко, В. И. Криптографические методы защиты информации : учеб. пособие / В. И. Аверченко, М. Ю. Рытов, С. А. Шпичак. – Брянск : БГТУ, 2011. – 216 с.

2. Кирпичников, А. П. Криптографические методы защиты компьютерной информации : учеб. пособие / А. П. Кирпичников, З. М. Хайбуллина ; М-во образования и науки России, Казан. науч.-исслед. технол. ун-т. – Казань : Казан. науч.-исслед. технол. ун-т, 2016. – 100 с. : табл., схемы. – Режим доступа: http://biblioclub.ru/index.php?page=book&id=560536. – Дата доступа: 25.07.2020.

3. Романец, Ю. В. Защита информации в компьютерных системах и сетях / Ю. В. Романец, П. А. Тимофеев, В. Ф. Шаньгин. – 2-е изд., перераб. и доп. – М. : Радио и связь, 2001. – 376 с.

4. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер. – М. : Триумф, 2012. – 600 с.

Учебное электронное издание комбинированного распространения

Учебное издание

МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ

Практикум
по выполнению лабораторных работ
для студентов специальности 1-40 04 01
«Информатика и технологии программирования»
дневной формы обучения

Электронный аналог печатного издания

Составитель **Прокопенко** Дмитрий Викторович

Редактор *Н. В. Гладкова*
Компьютерная верстка *И. П. Минина*

Подписано в печать 20.11.20.
Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».
Ризография. Усл. печ. л. 4,65. Уч.-изд. л. 4,01.
Изд. № 19.
http://www.gstu.by

Издатель и полиграфическое исполнение
Гомельский государственный
технический университет имени П. О. Сухого.
Свидетельство о гос. регистрации в качестве издателя
печатных изданий за № 1/273 от 04.04.2014 г.
пр. Октября, 48, 246746, г. Гомель