Практическая работа №4

КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ В УПРАВЛЯЮЩИХ СИСТЕМАХ НА ТРАНСПОРТЕ

1 КРАТКИЕ СВЕДЕНИЯ ИЗ ТЕОРИИ

1.1 Методы криптографического преобразования информации

Наукой, изучающей математические методы защиты информации путем ее преобразования, является **криптология** [κρυπτοζ – тайный, λογοζ – наука (слово) (греч.)]. Криптология разделяется на два направления – криптографию и криптоанализ.

Под **криптографической защитой информации** понимается такое преобразование исходной информации, в результате которого она становится недоступной для ознакомления и использования лицами, не имеющими на это полномочий.

Известны различные подходы к классификации методов криптографического преобразования информации. По виду воздействия на исходную информацию **методы криптографического преобразования информации** могут быть разделены на четыре группы (рисунок 1).

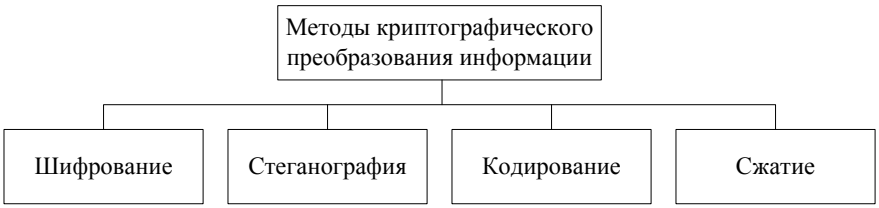

Рисунок 1 – Классификация методов криптографического преобразования информации

1.1.1 Шифрование информации

Основным видом криптографического преобразования информации в компьютерных системах является шифрование. Под **шифрованием** понимается процесс преобразования открытой информации в зашифрованную (шифротекст) или процесс обратного преобразования зашифрованной информации в открытую. Процесс преобразования открытой информации в закрытую получил название шифрование, а процесс преобразования закрытой информации в открытую – расшифрование (рисунок 2).

Процесс шифрования заключается в проведении обратимых математических, логических, комбинаторных и других преобразований исходной информации, в результате которых зашифрованная информация представляет собой хаотический набор букв, цифр, других символов и двоичных кодов.

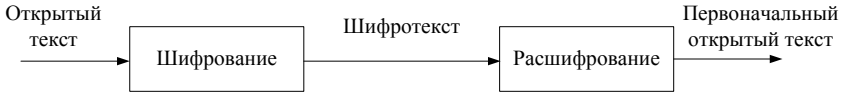

Рисунок 2 – Шифрование и расшифрование

Для шифрования информации используются алгоритм преобразования и ключ. Как правило, алгоритм для определенного метода шифрования является неизменным. Исходными данными для алгоритма шифрования служат информация, подлежащая шифрованию, и ключ шифрования. Ключ – конкретное значение некоторых параметров алгоритма криптографического преобразования, обеспечивающее выбор преобразования из семейства. Он содержит управляющую информацию, которая определяет выбор преобразования на определенных шагах алгоритма и величины операндов, используемые при реализации алгоритма шифрования.

За многовековую историю использования шифрования информации человечеством изобретено множество методов шифрования, или шифров. Методом шифрования (шифром) называется совокупность обратимых преобразований открытой информации в закрытую в соответствии с алгоритмом шифрования. Большинство методов шифрования не выдержали проверку временем, а некоторые используются и до сих пор. Появление ЭВМ инициировало процесс разработки новых шифров, учитывающих их возможности использования как для шифрования/дешифрования информации, так и для атак на шифр. Атака на шифр (криптоанализ) – это процесс дешифрования закрытой информации без знания ключа и, возможно, при отсутствии сведений об алгоритме шифрования. Процесс восстановления первоначального открытого текста на основе шифрованного без знания ключа называют дешифрованием.

Современные *методы шифрования должны отвечать следующим требованиям*:

– стойкость шифра противостоять криптоанализу (криптостойкость) должна быть такой, чтобы вскрытие его могло быть осуществлено только путем решения задачи полного перебора ключей;

– криптостойкость обеспечивается не секретностью алгоритма шифрования, а секретностью ключа;

– шифротекст не должен существенно превосходить по объему исходную информацию;

– ошибки, возникающие при шифровании, не должны приводить к искажениям и потерям информации;

– время шифрования не должно быть большим;

– стоимость шифрования должна быть согласована со стоимостью закрываемой информации.

Криптостойкость шифра является его основным показателем эффективности. Она измеряется временем или стоимостью средств, необходимых криптоаналитику для получения исходной информации по шифротексту, при условии, что ему неизвестен ключ.

Сохранить в секрете широко используемый алгоритм шифрования практически невозможно. Поэтому алгоритм не должен иметь скрытых слабых мест, которыми могли бы воспользоваться криптоаналитики. Если это условие выполняется, то криптостойкость шифра определяется длиной ключа, так как единственный путь вскрытия зашифрованной информации – перебор комбинаций ключа и выполнение алгоритма дешифрования. Таким образом, время и средства, затрачиваемые на криптоанализ, зависят от длины ключа и сложности алгоритма шифрования.

В качестве примера удачного метода шифрования можно привести шифр DES (Data Encryption Standard), применяемый в США с 1978 года в качестве государственного стандарта. Алгоритм шифрования не является секретным и был опубликован в открытой печати. За все время использования этого шифра не было обнародовано ни одного случая обнаружения слабых мест в алгоритме шифрования.

В конце 70-х годов использование ключа длиной в 56 бит гарантировало, что для раскрытия шифра потребуется несколько лет непрерывной работы самых мощных по тем временам компьютеров. Прогресс в области вычислительной техники позволил значительно сократить время определения ключа путем полного перебора. Согласно заявлению специалистов Агентства национальной безопасности США, 56-битный ключ для DES может быть найден менее чем за 453 дня с использованием суперЭВМ Cray T3D, которая имеет 1024 узла и стоит 30 млн дол. Используя чип FPGA (Field Progammably Gate Array – программируемая вентильная матрица) стоимостью 400 дол., можно восстановить 40-битный ключ DES за 5 часов. Потратив 10000 дол. за 25 чипов FPGA, 40-битный ключ можно найти в среднем за 12 мин. Для вскрытия 56-битного ключа DES при опоре на серийную технологию и затратах в 300000 дол. требуется в среднем 19 дней, а если разработать специальный чип, – то 3 часа. При затратах в 300 млн дол. 56-битные ключи могут быть найдены за 12 с. Расчеты показывают, что в настоящее время для надежного закрытия информации длина ключа должна быть не менее 90 бит.

Современная криптография включает в себя четыре основные группы криптографических методов защиты информации:

– симметричные криптосистемы;

– асимметричные криптосистемы (криптосистемы с открытым ключом);

– системы электронной цифровой подписи;

– управление ключами.

В *симметричных криптосистемах* для шифрования и расшифрования используется один и тот же ключ (рисунок 3).

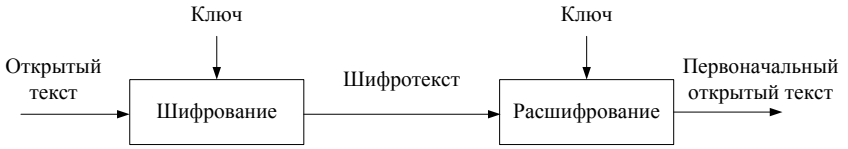

Рисунок 3 – Шифрование и расшифрование с одним ключом

В *асимметричных криптосистемах* используются два ключа – открытый и секретный, которые математически связаны друг с другом (рисунок 4). Информация шифруется с помощью открытого ключ, который доступен всем желающим, а расшифровывается с помощью закрытого ключа, известного только получателю сообщения.

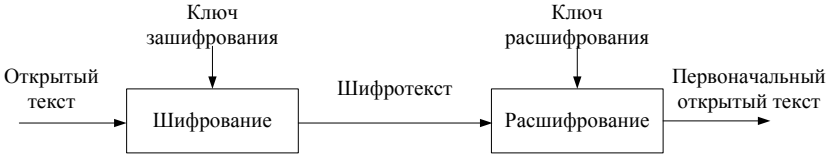

Рисунок 4 – Шифрование и расшифрование с двумя ключами

Электронной цифровой подписью называется присоединение к тексту его криптографического преобразования, которое позволяет при получении текста другим пользователем проверить авторство и целостность сообщения.

Управление ключами – это процесс системы обработки информации, вырабатывающий и распределяющий ключи (открытые и секретные) между пользователями.

Далее данные группы методов будут рассмотрены подробнее.

1.1.2 Стеганография

В отличие от других методов криптографического преобразования информации, методы **стеганографии** позволяют скрыть не только смысл хранящейся или передаваемой информации, но и сам факт хранения или передачи закрытой информации. В компьютерных системах практическое использование стеганографии только начинается, но проведенные исследова-

ния показывают ее перспективность. В основе всех методов стеганографии лежит маскирование закрытой информации среди открытых файлов. Обработка мультимедийных файлов в компьютерных системах открыла практически неограниченные возможности перед стеганографией.

Существует несколько методов скрытой передачи информации. Одним из них является простой метод скрытия файлов при работе в операционной системе MS DOS. За текстовым открытым файлом записывается скрытый двоичный файл, объем которого много меньше текстового файла. В конце текстового файла помещается метка EOF (комбинация клавиш Ctrl и Z). При обращении к этому текстовому файлу стандартными средствами ОС считывание прекращается по достижению метки EOF, и скрытый файл остается недоступен. Для двоичных файлов никаких меток в конце файла не предусмотрено. Конец такого файла определяется при обработке атрибутов, в которых хранится длина файла в байтах. Доступ к скрытому файлу может быть получен, если файл открыть как двоичный. Скрытый файл может быть зашифрован. Если кто-то случайно обнаружит скрытый файл, то зашифрованная информация будет воспринята как сбой в работе системы.

Графическая и звуковая информации представляются в числовом виде. Так, в графических объектах наименьший элемент изображения может кодироваться одним байтом. В младшие разряды определенных байтов изображения в соответствии с алгоритмом криптографического преобразования помещаются биты скрытого файла. Если правильно подобрать алгоритм преобразования и изображение, на фоне которого помещается скрытый файл, то человеческому глазу практически невозможно отличить полученное изображение от исходного. Очень сложно выявить скрытую информацию и с помощью специальных программ. Наилучшим образом для внедрения скрытой информации подходят изображения местности: фотоснимки со спутников, самолетов и т. п. С помощью средств стеганографии могут маскироваться текст, изображение, речь, цифровая подпись, зашифрованное сообщение. Комплексное использование стеганографии и шифрования многократно повышает сложность решения задачи обнаружения и раскрытия конфиденциальной информации.

1.1.3 Кодирование информации

Содержанием процесса **кодирования информации** является замена смысловых конструкций исходной информации (слов, предложений) кодами. В качестве кодов могут использоваться сочетания букв, цифр, букв и цифр. При кодировании и обратном преобразовании используются специальные таблицы или словари. Кодирование информации целесообразно применять в системах с ограниченным набором смысловых конструкций. Такой вид криптографического преобразования применим, например, в ко-

мандных линиях автоматизированных систем управления. Недостатками кодирования конфиденциальной информации является необходимость хранения и распространения кодировочных таблиц, которые необходимо часто менять, чтобы избежать раскрытия кодов статистическими методами обработки перехваченных сообщений.

1.1.4 Сжатие информации

Сжатие информации может быть отнесено к методам криптографического преобразования информации с определенными оговорками. Целью сжатия является сокращение объема информации. В то же время сжатая информация не может быть прочитана или использована без обратного преобразования. Учитывая доступность средств сжатия и обратного преобразования, эти методы нельзя рассматривать как надежные средства криптографического преобразования информации. Даже если держать в секрете алгоритмы, то они могут быть сравнительно легко раскрыты статистическими методами обработки. Поэтому сжатые файлы конфиденциальной информации подвергаются последующему шифрованию. Для сокращения времени целесообразно совмещать процесс сжатия и шифрования информации.

В некоторых источниках стеганография, кодирование и сжатие информации относятся к отраслям знаний, смежных с криптографией, но не входящих в нее.

1.2 Симметричные криптосистемы

1.2.1 Традиционные методы шифрования

К традиционным (классическим) методам шифрования относятся шифры перестановки, шифры простой и сложной замены, а также некоторые их модификации и комбинации. Комбинации шифров перестановок и шифров замены образуют все многообразие применяемых на практике симметричных шифров.

Шифры перестановки. При шифровании перестановкой символы шифруемого текста переставляются по определенному правилу в пределах блока этого текста. Шифры перестановки являются самыми простыми и, вероятно, самыми древними шифрами.

Шифрующие таблицы. В качестве ключа в шифрующих таблицах используются: размер таблицы, слово или фраза, задающие перестановку, особенности структуры таблицы.

Одним из самых примитивных табличных шифров перестановки является простая перестановка, для которой ключом служит размер таблицы. Например, сообщение В БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТРАНСПОРТА записывается в таблицу поочередно по столб-

цам. Результат заполнения таблицы из 6 строк и 8 столбцов показан на рисунке 5. После заполнения таблицы текстом сообщения по столбцам для формирования шифротекста считывают содержимое таблицы по строкам.

В	У	Г	Р	Н	В	Е	С
Б	С	О	С	Ы	Е	Т	П
Е	С	С	Т	Й	Р	Т	О
Л	К	У	В	У	С	Р	Р
О	И	Д	Е	Н	И	А	Т
Р	Й	А	Н	И	Т	Н	А

Рисунок 5 – Заполнение сообщения в таблице из 6 строк и 8 столбцов

Если шифротекст записывать группами по восемь букв, получается такое шифрованное сообщение: ВУГРНВЕС БСОСЫЕТП ЕССТЙРТО ЛКУВУСРР ОИДЕНИАТ РЙАНИТНА.

Естественно, отправитель и получатель сообщения должны заранее условиться об общем ключе в виде размера таблицы. Следует заметить, что объединение букв шифротекста в 8-буквенные группы не входит в ключ шифра и осуществляется для удобства записи несмыслового текста. При расшифровании действия выполняют в обратном порядке.

Несколько большей стойкостью к раскрытию обладает метод шифрования, называемый *одиночной перестановкой* по ключу. Этот метод отличается от предыдущего тем, что столбцы таблицы переставляются по ключевому слову, фразе или набору чисел длиной в строку таблицы. Применим в качестве ключа, например, слово АЛГОРИТМ, а текст сообщения возьмем из предыдущего примера. На рисунке 6 показаны две таблицы, заполненные текстом сообщения и ключевым словом, при этом левая таблица соответствует заполнению до перестановки, а правая – после перестановки.

А	Л	Г	О	Р	И	Т	М
1	4	2	6	7	3	8	5
В	У	Г	Р	Н	В	Е	С
Б	С	О	С	Ы	Е	Т	П
Е	С	С	Т	Й	Р	Т	О
Л	К	У	В	У	С	Р	Р
О	И	Д	Е	Н	И	А	Т
Р	Й	А	Н	И	Т	Н	А

До перестановки

А	Г	И	Л	М	О	Р	Т
1	2	3	4	5	6	7	8
В	Г	В	У	С	Р	Н	Е
Б	О	Е	С	П	С	Ы	Т
Е	С	Р	С	О	Т	Й	Т
Л	У	С	К	Р	В	У	Р
О	Д	И	И	Т	Е	Н	А
Р	А	Т	Й	А	Н	И	Н

После перестановки

Рисунок 6 – Таблицы, заполненные ключевым словом и текстом сообщения

В верхней строке левой таблицы записан ключ, а номера под буквами ключа определены в соответствии с естественным порядком соответствующих букв ключа в алфавите. Если бы в ключе встретились одинаковые буквы, они были бы пронумерованы слева направо. В правой таблице столбцы переставлены в соответствии с упорядоченными номерами букв ключа. При считывании содержимого правой таблицы по строкам и записи шифротекста группами по восемь букв получим шифрованное сообщение: ВГВУСРНЕ БОЕССПСЫТ ЕСРСОТЙТ ЛУСКРВУР ОДИИТЕНА РАТЙАНИН.

Для обеспечения дополнительной скрытности можно повторно зашифровать сообщение, которое уже было зашифровано. Такой метод шифрования называется *двойной перестановкой*. В случае двойной перестановки столбцов и строк таблицы перестановки определяются отдельно для столбцов и для строк. Сначала в таблицу записывается текст сообщения по столбцам, а потом поочередно переставляются столбцы, а затем строки. При расшифровании порядок перестановок должен быть обратным. Пример выполнения шифрования методом двойной перестановки показан на рисунке 7.

Если считывать шифротекст из правой таблицы построчно блоками по восемь букв, то получится следующее: СИОКУЙГС РИТСРТЕТ ВЕРБСЛОУ ДРВАНСТЕ НЙИЫЕУНВ АСРНАПОТ.

		А	Л	Г	О	Р	И	Т	М
		1	4	2	6	7	3	8	5
З	3	В	Б	Е	Л	О	Р	У	С
А	1	С	К	И	Й	Г	О	С	У
П	4	Д	А	Р	С	Т	В	Е	Н
Р	5	Н	Ы	Й	У	Н	И	В	Е
Е	2	Р	С	И	Т	Е	Т	Т	Р
Т	6	А	Н	С	П	О	Р	Т	А

До второй перестановки

		А	Г	И	Л	М	О	Р	Т
		1	2	3	4	5	6	7	8
А	1	С	И	О	К	У	Й	Г	С
Е	2	Р	И	Т	С	Р	Т	Е	Т
З	3	В	Е	Р	Б	С	Л	О	У
П	4	Д	Р	В	А	Н	С	Т	Е
Р	5	Н	Й	И	Ы	Е	У	Н	В
Т	6	А	С	Р	Н	А	П	О	Т

После второй перестановки

Рисунок 7 – Шифрование методом двойной перестановки

Ключом к шифру двойной перестановки служат два слова: АЛГОРИТМ и ЗАПРЕТ. Вторая перестановка осуществляется с таблицей из предыдущего примера.

Число вариантов двойной перестановки быстро возрастает при увеличении размера таблицы: для таблицы 3×3 – 36 вариантов, для таблицы 4×4 – 576 вариантов, для таблицы 5×5 – 14400 вариантов. Однако двойная перестановка не отличается высокой стойкостью и сравнительно просто "взламывается" при любом размере таблицы шифрования.

Шифры простой замены. При шифровании заменой (подстановкой) символы шифруемого текста заменяются символами того же или другого

алфавита с заранее установленным правилом замены. В шифре простой замены каждый символ исходного текста заменяется символами того же алфавита по одному правилу на всем протяжении текста. Часто шифры простой замены называют шифрами одноалфавитной подстановки.

Система шифрования Цезаря. Шифр Цезаря является частным случаем шифра простой замены (одноалфавитной подстановки). Свое название этот шифр получил по имени римского императора Гая Юлия Цезаря, который использовал этот шифр при переписке.

При шифровании исходного текста каждая буква заменялась на другую букву того же алфавита по следующему правилу. Заменяющая буква определялась путем смещения по алфавиту m от исходной буквы на k букв. При достижении конца алфавита выполнялся циклический переход к его началу. Цезарь использовал латинский алфавит $m = 26$ и шифр замены при смещении $k = 3$. Такой шифр замены можно задать таблицей подстановок, содержащей соответствующие пары букв открытого текста и шифротекста. Совокупность возможных подстановок для $k = 3$ показана в таблице 1.

Таблица 1 – **Одноалфавитные подстановки ($k = 3$, $m = 26$)**

A → D	E → H	I → L	M → P	Q → T	U → X	Y → B
B → E	F → I	J → M	N → Q	R → U	V → Y	Z → C
C → F	G → J	K → N	O → R	S → V	W → Z	
D → G	H → K	L → O	P → S	T → W	X → A	

Например, известное послание Цезаря VENI VIDI VICI (в переводе на русский означает "Пришел, увидел, победил") выглядело бы в зашифрованном виде так: YHQL YLGL YLFL.

Система шифрования Цезаря образует, по существу, семейство одноалфавитных подстановок для выбираемых значений ключа k, причем $0 \le k < m$. Достоинством системы шифрования Цезаря является простота шифрования и расшифрования. К недостаткам системы Цезаря можно отнести следующее:

– подстановки, выполняемые в соответствии с системой Цезаря, не маскируют частот появления различных букв исходного открытого текста;

– сохраняется алфавитный порядок в последовательности заменяющих букв; при изменении значения k изменяются только начальные позиции такой последовательности;

– число возможных ключей k мало;

– шифр Цезаря легко вскрывается на основе анализа частот появления букв в шифротексте.

Криптоаналитическая атака против системы одноалфавитной замены начинается с подсчета частот появления символов: определяется число появлений каждой буквы в шифротексте. Затем полученное распределение

частот букв в шифротексте сравнивается с распределением частот букв в алфавите исходных сообщений. Буква с наивысшей частотой появления в шифротексте заменяется на букву с наивысшей частотой появления в алфавите и т. д. Вероятность успешного вскрытия системы шифрования повышается с увеличением длины шифротекста. Вместе с тем идеи, заложенные в системе шифрования Цезаря, оказались весьма плодотворными, о чем свидетельствуют их многочисленные модификации.

Аффинная система подстановок Цезаря. В данном преобразовании буква, соответствующая числу t, заменяется на букву, соответствующую числовому значению $(at + b)$ по модулю m. Такое преобразование является взаимно однозначным отображением на алфавите тогда и только тогда, когда НОД (a, m) – наибольший общий делитель чисел a и m равен единице, т. е. если a и m – взаимно простые числа.

Например, возьмем $m = 26$, $a = 3$, $b = 5$. Тогда, очевидно, НОД (3,26) = 1, и мы получаем соответствие между числовыми кодами букв. Преобразуя числа в буквы английского языка, получаем соответствие для букв открытого текста (A) и шифротекста (B) (таблица 2).

Таблица 2 – **Соответствие между символами открытого текста и шифротекста**

A	t	$3t + 5$	B	A	t	$3t + 5$	B	A	t	$3t + 5$	B
A	0	5	F	J	9	6	G	S	18	7	H
B	1	8	I	K	10	9	J	T	19	10	K
C	2	11	L	L	11	12	M	U	20	13	N
D	3	14	O	M	12	15	P	V	21	16	Q
E	4	17	R	N	13	18	S	W	22	19	T
F	5	20	U	O	14	21	V	X	23	22	W
G	6	23	X	P	15	24	Y	Y	24	25	Z
H	7	0	A	Q	16	1	B	Z	25	2	C
I	8	3	D	R	17	4	E				

Исходное сообщение HOPE преобразуется в шифротекст AVYR.

Достоинством аффинной системы является удобное управление ключами: ключи шифрования и дешифрования представляются в компактной форме в виде пары чисел (a, b). Недостатки аффинной системы аналогичны недостаткам системы шифрования Цезаря. На практике аффинная система использовалась несколько веков назад.

Шифры сложной замены. Шифры сложной замены называют многоалфавитными, так как для шифрования каждого символа исходного сообщения применяют свой шифр простой замены. Многоалфавитная подстановка последовательно и циклически меняет используемые алфавиты. При r-алфавитной подстановке символ x_0 исходного сообщения заменяется символом y_0 из алфавита B_0, символ x_1 – символом y_1 из алфавита B_1 , и т. д.; символ x_{r-1} заменяется символом y_{r-1} из алфавита B_{r-1} , символ x_r заменяется символом y_r снова из алфавита B_0, и т. д. Общая схема многоалфавитной

подстановки для случая $r = 4$ показана на рисунке 8.

Эффект использования многоалфавитной подстановки заключается в том, что обеспечивается маскировка естественной статистики исходного языка, так как конкретный символ из исходного алфавита A может быть преобразован в несколько различных символов шифровальных алфавитов B_j. Степень обеспечиваемой защиты теоретически пропорциональна длине периода r в последовательности используемых алфавитов B_j.

Входной символ	x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
Алфавит подстановки	B_0	B_1	B_2	B_3	B_0	B_1	B_2	B_3	B_0	B_1

Рисунок 8 – Схема r-алфавитной подстановки для случая $r = 4$

Система шифрования Вижинера. Система Вижинера подобна такой системе шифрования Цезаря, у которой ключ подстановки меняется от буквы к букве. Этот шифр многоалфавитной замены можно описать таблицей шифрования, называемой таблицей (квадратом) Вижинера. На рисунках 9 и 10 показаны таблицы Вижинера для русского и английского алфавитов соответственно. Таблица Вижинера используется как для шифрования, так и для расшифрования. Она имеет два входа:

– верхнюю строку подчеркнутых символов, используемую для считывания очередной буквы исходного открытого текста;

– крайний левый столбец ключа.

Последовательность ключей обычно получают из числовых значений букв ключевого слова. При шифровании исходного сообщения его выписывают в строку, а под ним записывают ключевое слово (или фразу).

Если ключ оказался короче сообщения, то его циклически повторяют. В процессе шифрования находят в верхней строке таблицы очередную букву исходного текста и в левом столбце очередное значение ключа. Очередная буква шифротекста находится на пересечении столбца, определяемого шифруемой буквой, и строки, определяемой числовым значением ключа.

Рассмотрим пример получения шифротекста с помощью таблицы Вижинера. Пусть выбрано ключевое слово ТРАНСПОРТ. Необходимо зашифровать сообщение СЕГОДНЯ В ШЕСТЬ. Выпишем исходное сообщение в строку и запишем под ним ключевое слово с повторением. В третью строку будем выписывать буквы шифротекста, определяемые из таблицы Вижинера (рисунок 11).

Одноразовые шифры. Почти все применяемые на практике шифры характеризуются как условно надежные, поскольку они могут быть в принципе раскрыты при наличии неограниченных вычислительных возможностей. Абсолютно надежные шифры нельзя разрушить даже при использовании неограниченных вычислительных возможностей. Существует единственный такой шифр, применяемый на практике, – одноразовая система шифрования. Характерной особенностью одноразовой системы шифрования являет-

ся одноразовое использование ключевой последовательности.

Ключ	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я
0	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я
1	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а
2	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б
3	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в
4	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г
5	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д
6	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е
7	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж
8	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з
9	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и
10	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й
11	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к
12	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л
13	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м
14	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н
15	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о
16	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
17	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р
18	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с
19	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т
20	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у
21	х	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф
22	ц	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х
23	ч	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц
24	ш	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч
25	щ	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш
26	ь	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ
27	ы	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь
28	ъ	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы
29	э	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ
30	ю	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э
31	я	а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ь	ы	ъ	э	ю

Рисунок 9 – Таблица Вижинера для русского алфавита

Этот шифр абсолютно надежен, если набор ключей K_i действительно случаен и непредсказуем. Если криптоаналитик попытается использовать для заданного шифротекста все возможные наборы ключей и восстановить все возможные варианты исходного текста, то они все окажутся равновероятными. Не существует способа выбрать исходный текст, который был действительно послан. Теоретически доказано, что одноразовые системы являются недешифрируемыми системами, поскольку их шифротекст не содержит достаточной информации для восстановления открытого текста.

Ключ	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
0	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
1	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a
2	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b
3	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c
4	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d
5	f	g	h	I	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e
6	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f
7	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g
8	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h
9	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i
10	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j
11	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k
12	m	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l
13	n	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m
14	o	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n
15	p	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
16	q	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
17	r	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q
18	s	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r
19	t	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s
20	u	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t
21	v	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
22	w	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v
23	x	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w
24	y	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x
25	z	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y

Рисунок 10 – Таблица Вижинера для английского алфавита

Сообщение	С	Е	Г	О	Д	Н	Я	В	Ш	Е	С	Т	Ь
Ключ	Т	Р	А	Н	С	П	О	Р	Т	Т	Р	А	Н
Шифротекст	Г	Х	Г	Ы	Х	Ъ	Н	Т	Й	Ч	Б	Т	Ж

Рисунок 11 – Пример шифрования сообщения с помощью таблицы Вижинера

Возможности применения одноразовой системы ограничены чисто практическими аспектами. Существенным моментом является требование одноразового использования случайной ключевой последовательности. Ключевая последовательность с длиной, не меньшей длины сообщения, должна передаваться получателю сообщения заранее или отдельно по некоторому секретному каналу. Такое требование практически сложно осуществимо для современных систем обработки информации, где требуется шифровать многие миллионы символов, однако в обоснованных случаях построение систем с одноразовыми шифрами является наиболее целесообразным.

Исторически различают шифраторы с внешней и внутренней гаммами. В шифраторах с внешней гаммой в качестве ключа используется случайная последовательность однократного использования, длина которой равна длине шифруемого сообщения. В шифраторах с внутренней гаммой в качестве ключа применяется многократно используемая случайная последовательность длиной, много меньшей длины шифруемого текста, на основе которой формируется гамма шифра. Шифраторы с внутренней гаммой, т. е. обладающие свойством практической стойкости, в настоящее время являются преобладающими при построении систем шифрованной связи. Основным их достоинством является простота в управлении ключами, т. е. их заготовка, распределение, доставка и уничтожение. Данное преимущество позволяет на основе шифраторов с внутренней гаммой создавать системы шифрованной связи практически любых размеров, не ограничивая их географию и количество абонентов.

Современное развитие информационных технологий позволяет концентрировать значительное количество информации на физических носителях небольшого размера, что также обусловливает практическую применимость данного подхода.

Задача построения системы шифрованной связи на основе шифраторов с внешней гаммой может иметь несколько подходов к ее решению. Например, исходя из установленного граничного значения объема ключевого документа, определяются оптимальное количество абонентов системы и допустимая нагрузка. С другой стороны, можно, исходя из требуемого количества абонентов и нагрузки на них, рассчитать необходимый объем ключевого документа.

Шифрование методом гаммирования. Под *гаммированием* понимают процесс наложения по определенному закону гаммы шифра на открытые данные. *Гамма шифра* – это псевдослучайная последовательность, выработанная по заданному алгоритму для шифрования открытых данных и расшифрования принятых данных.

Процесс шифрования заключается в генерации гаммы шифра и наложении полученной гаммы на исходный открытый текст обратимым образом, например, с использованием операции сложения по модулю 2.

Перед шифрованием открытые данные разбивают на блоки одинаковой длины, обычно по 64 бита. Гамма шифра вырабатывается в виде последовательности аналогичной длины. Процесс расшифрования сводится к повторной генерации гаммы шифра и наложению этой гаммы на принятые данные.

Получаемый этим методом шифротекст достаточно труден для раскрытия, поскольку теперь ключ является переменным. По сути дела, гамма шифра должна изменяться случайным образом для каждого шифруемого блока. Если период гаммы превышает длину всего шифруемого текста и злоумышленнику неизвестна никакая часть исходного текста, то такой

шифр можно раскрыть только прямым перебором всех вариантов ключа. В этом случае криптостойкость шифра определяется длиной ключа.

1.2.2 Современные симметричные криптосистемы

Американский стандарт шифрования данных DES. Стандарт шифрования данных DES (Data Encryption Standard) опубликован в 1977 г. Национальным бюро стандартов США. Он предназначен для защиты от несанкционированного доступа к важной, но не секретной информации в государственных и коммерческих организациях США.

Основные достоинства алгоритма DES:

– используется только один ключ длиной 56 бит;

– относительная простота алгоритма обеспечивает высокую скорость обработки;

– достаточно высокая стойкость алгоритма.

Алгоритм DES основан на комбинировании методов подстановки и перестановки и состоит из чередующейся последовательности блоков перестановки и подстановки. DES осуществляет шифрование 64-битовых блоков данных с помощью 64-битового ключа, в котором значащими являются 56 бит (остальные 8 бит – проверочные биты для контроля на четность). Расшифрование в DES является операцией, обратной шифрованию, и выполняется путем повторения операций шифрования в обратной последовательности. Обобщенная схема процесса шифрования в алгоритме DES показана на рисунке 12.

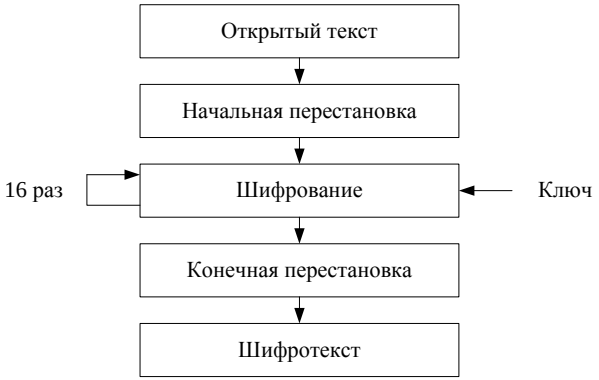

Рисунок 12 – Обобщенная схема шифрования в алгоритме DES

Процесс шифрования заключается в начальной перестановке битов 64-битового блока, шестнадцати циклах шифрования и, наконец, в конечной перестановке битов. Следует сразу отметить, что все приводимые таблицы являются стандартными и должны включаться в реализацию

алгоритма DES в неизменном виде. Все перестановки и коды в таблицах подобраны разработчиками таким образом, чтобы максимально затруднить процесс расшифровки путем подбора ключа. При описании алгоритма DES (рисунок 13) применены следующие обозначения:

L и R – последовательности битов [левая (left) и правая (right)];

LR – конкатенация последовательностей L и R, т. е. такая последовательность битов, длина которой равна сумме длин L и R; в последовательности LR биты последовательности R следуют за битами последовательности L;

$\oplus$ – операция побитового сложения по модулю два.

Пусть из файла исходного текста считан очередной 64-битовый блок T. Этот блок T преобразуется с помощью *матрицы начальной перестановки IP* (таблица 3). Биты входного блока T (64 бита) переставляются в соответствии с матрицей IP: бит 58 входного блока T становится битом 1, бит 50 – битом 2 и т. д. Полученная последовательность битов T_0 разделяется на две последовательности: L_0 – левые, или старшие биты и R_0 – правые, или младшие биты, каждая из которых содержит 32 бита. Затем выполняется итеративный процесс шифрования, состоящий из 16 шагов (циклов). Пусть T_i – результат i-й итерации: $T_i = L_iR_i$, где $L_i = t_1, t_2, ..., t_{32}$ (первые 32 бита); $R_i = t_{33}, t_{34}, ..., t_{64}$ (последние 32 бита). Тогда результат i-й итерации описывается следующими формулами: $L_i = R_{i-1}$, $i = 1, 2, ..., 16$; $R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$, $i = 1, 2, ..., 16$.

Функция f называется функцией шифрования. Ее аргументами являются последовательность R_{i-1}, получаемая на предыдущем шаге итерации, и 48-битовый ключ K_i, который является результатом преобразования 64-битового ключа шифра K. На последнем шаге итерации получают последовательности R_{16} и L_{16} (без перестановки местами), которые конкатенируются в 64-битовую последовательность $R_{16}\ L_{16}$. По окончании шифрования осуществляется восстановление позиций битов с помощью матрицы обратной перестановки IP^{-1} (таблица 4).

Процесс расшифрования данных является инверсным по отношению к процессу шифрования. Все действия должны быть выполнены в обратном порядке. Это означает, что расшифровываемые данные сначала представляются в соответствии с матрицей IP^{-1}, а затем над последовательностью битов $R_{16}L_{16}$ выполняются те же действия, что и в процессе шифрования, но в обратном порядке.

Итеративный процесс расшифрования может быть описан следующими формулами:

$$R_{i-1} = L_i,\ i = 1, 2, \ldots, 16;\ \ L_{i-1} = R_i \oplus f(L_i, K_i),\ i = 1, 2, ..., 16. \qquad (1)$$

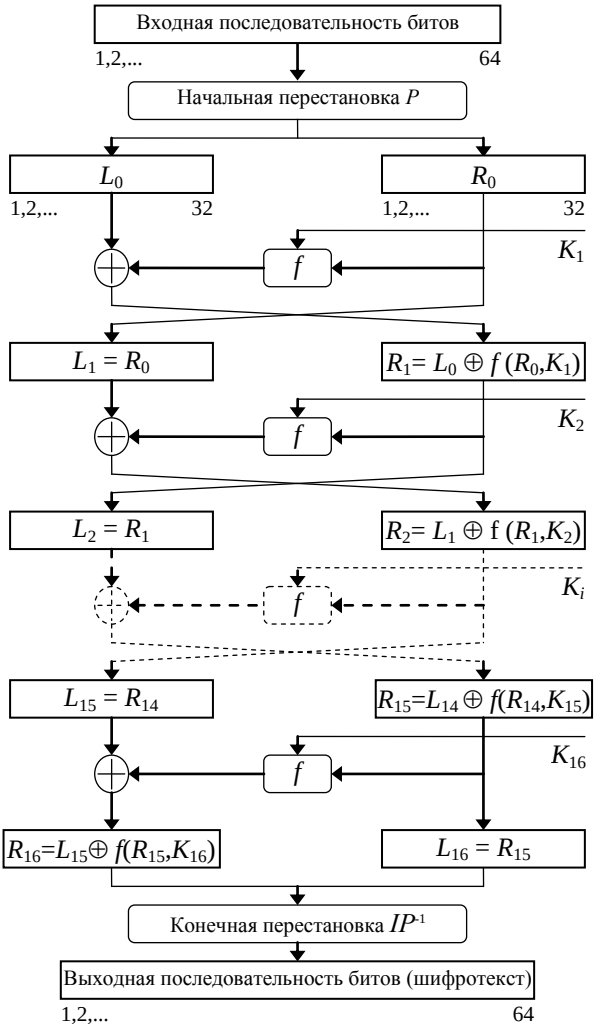

Рисунок 13 – Структура алгоритма DES

Таким образом, для процесса расшифрования с переставленным входным блоком $R_{16}L_{16}$ на первой итерации используется ключ K_{16}, на второй – K_{15} и т. д., на 16-й – K_1. На последнем шаге итерации будут получены последовательности L_0 и R_0, которые конкатенируются в 64-битовую последовательность L_0R_0. Затем в этой последовательности 64 бита переставляются в соответствии с матрицей *IP*. Результат такого преобразования – исходная последовательность битов (расшифрованное 64-битовое значение).

Таблица 3 – **Матрица начальной перестановки IP**

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

Таблица 4 – **Матрица обратной перестановки IP^{-1}**

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

Для вычисления значения функции f используются:

– функция E (расширение 32 бит до 48);

– функция S_1, S_2, ..., S_8 (преобразование 6-битового числа в 4-битовое);

– функция P (перестановка битов в 32-битовой последовательности).

Результат функции E (R_{i-1}) есть 48-битовое число. Функция расширения E, выполняющая расширение 32 бит до 48 (принимает блок из 32 бит и порождает блок из 48 бит), определяется таблицей 5.

Таблица 5 – **Функция расширения *E***

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

В соответствии с таблицей 5 первые три бита $E(R_{i-1})$ – это биты 32, 1 и 2, а последние – 31, 32 и 1. Полученный результат [обозначим его $E(R_{i-1})$] складывается по модулю два (операция XOR) с текущим значением ключа K_i и затем разбивается на восемь 6-битовых блоков B_1, B_2, ..., B_8. Далее каждый из этих блоков используется как номер элемента в функциях-матрицах S_1, S_2, ..., S_8, содержащих 4-битовые значения (таблица 6).

Таблица 6 – **Функции преобразования S_1, S_2, ..., S_8**

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	
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8	S_1
2	4	1	4	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	
1	3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5	S_2
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	
1	13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1	S_3
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	
1	13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9	S_4
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	
1	14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6	S_5
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	
1	10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8	S_6
2	9	14	15	5	2	8	12	3	7	0	4	10	1	13	1	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	
1	13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6	S_7
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	
1	1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2	S_8
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	

Выбор элемента в матрице S_j осуществляется следующим образом. Пусть на вход матрицы S_j поступает 6-битовый блок $B_j = b_1b_2b_3b_4b_5b_6$, тогда двухбитовое число b_1b_6 указывает номер строки матрицы, а четырехбитовое число $b_2b_3b_4b_5$ – номер столбца. Например, если на вход матрицы S_1 поступает 6-битовый блок $B_1 = 100110$, то 2-битовое число $b_1b_6 = 10_{(2)} = 2_{(10)}$ указывает строку с номером 2 матрицы S_1, а 4-битовое число $b_2b_3b_4b_5 = 0011_{(2)} = 3_{(10)}$ указывает столбец с номером 3 матрицы S_1. Это означает,

что в матрице S_1 блок B_1 = 100110 выбирает элемент на пересечении строки с номером 2 и столбца с номером 3, т. е. элемент $8_{(10)} = 1000_{(2)}$.

Совокупность 6-битовых блоков B_1, B_2, ..., B_8 обеспечивает выбор четырехбитового элемента в каждой из матриц S_1, S_2, ..., S_8. В результате получаем $S_1(B_1)$, $S_2(B_2)$, $S_3(B_3)$, ..., $S_8(B_8)$, т. е. 32-битовый блок (поскольку матрицы S_j содержат 4-битовые элементы). Этот 32-битовый блок преобразуется с помощью функции перестановки битов P (таблица 7).

Таблица 7 – **Функция *P* перестановки битов**

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

Таким образом, функция шифрования $f(R_{i-1}, K_i) = P(S_1(B_1), ..., S_8(B_8))$. На каждой итерации используется новое значение ключа K_i (длиной 56 бит), которое вычисляется из начального ключа K, представляющего собой 64-битовый блок с 8 битами контроля по четности, расположенными в позициях 8, 16, 24, 32, 40, 48, 56, 64. Для удаления контрольных битов и подготовки ключа к работе используется функция G первоначальной подготовки ключа (таблица 8). Эта таблица разделена на две части. Результат преобразования $G(K)$ разбивается на две половины C_0 и D_0 по 28 бит каждая. Первые четыре строки матрицы G определяют, как выбираются биты последовательности C_0 (первым битом C_0 будет бит 57 ключа шифра, затем бит 49 и т. д., а последними – биты 44 и 36 ключа). Следующие четыре строки матрицы G определяют, как выбираются биты последовательности D_0 (т. е. последовательность D_0 будет состоять из битов 63, 55, 47, ...,12, 4 ключа шифра).

Таблица 8 – **Функция *G* первоначальной подготовки ключа (переставленная выборка 1)**

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

Для генерации последовательностей C_0 и D_0 не используются биты 8, 16, 24, 32, 40, 48, 56 и 64 ключа шифра. Эти биты не влияют на шифрование и могут служить для других целей (например, для контроля по четности). Таким образом, в действительности ключ шифра является 56-битовым.

После определения C_0 и D_0 рекурсивно определяются C_i и D_i, где $i = 1, 2, \ldots, 16$. Для этого применяются операции циклического сдвига влево на один или два бита в зависимости от номера шага итерации, как показано в таблице 9.

Таблица 9 – **Таблица сдвигов s_i для вычисления ключа**

Номер итерации	Количество s_i сдвигов влево, бит	Номер итерации	Количество s_i сдвигов влево, бит
1	1	9	1
2	1	10	2
3	2	11	2
4	2	12	2
5	2	13	2
6	2	14	2
7	2	15	2
8	2	16	1

Операции сдвига выполняются для последовательностей C_i и D_i независимо. Например, последовательность C_3 получается посредством циклического сдвига влево на две позиции последовательности C_2, а последовательность D_3 – посредством сдвига влево на две позиции последовательности D_2; C_{16} и D_{16} получаются из C_{15} и D_{15} посредством сдвига влево на одну позицию. Ключ K_i, определяемый на каждом шаге итерации, есть результат выбора конкретных битов из 56-битовой последовательности C_iD_i и их перестановки. Другими словами, ключ $K_i = H(C_iD_i)$, где функция H определяется матрицей, завершающей обработку ключа (таблица 10). Первым битом ключа K_i будет 14-й бит последовательности C_iD_i, вторым – 17-й бит, 47-м – 29-й бит $C_i D_i$, а 48-м – 32-й бит $C_i D_i$.

Алгоритм DES вполне подходит как для шифрования, так и для аутентификации данных. Он позволяет непосредственно преобразовывать 64-битовый входной открытый текст в 64-битовый выходной шифрованный текст, однако реальные данные редко ограничиваются длиной в 64 разряда.

Чтобы воспользоваться алгоритмом DES для решения разнообразных криптографических задач, разработаны четыре рабочих режима:

– электронная кодовая книга ECB (Electronic Code Book);

– сцепление блоков шифра CBC (Cipher Block Chaining);

– обратная связь по шифротексту CFB (Cipher Feed Back);

– обратная связь по выходу OFB (Output Feed Back).

Таблица 10 – **Функция *H* завершающей обработки ключа (переставленная выборка 2)**

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

Режим "Электронная кодовая книга". Длинный файл разбивают на 64-битовые отрезки (блоки) по 8 байтов. Каждый из этих блоков шифруют независимо от использования одного и того же ключа шифрования (рисунок 14). Основное достоинство – простота реализации. Недостаток – относительно слабая устойчивость против квалифицированных криптоаналитиков. Из-за фиксированного характера шифрования при ограниченной длине блока в 64 бита возможно проведение криптоанализа "со словарем". Блок такого размера может повториться в сообщении вследствие большой избыточности в тексте на естественном языке. Это приводит к тому, что идентичные блоки открытого текста в сообщении будут представлены идентичными блоками шифротекста, что дает криптоаналитику некоторую информацию о содержании сообщения.

Режим "Сцепление блоков шифра". В этом режиме исходный файл M разбивается на 64-битовые блоки: $M = M_1, M_2, ..., M_n$. Первый блок M_1 складывается по модулю 2 с 64-битовым начальным вектором IV, который меняется ежедневно и держится в секрете (рисунок 15). Полученная сумма затем шифруется с использованием ключа DES, известного и отправителю, и получателю информации. Полученный 64-битовый шифр C_1 складывается по модулю 2 со вторым блоком текста, результат шифруется и получается второй 64-битовый шифр C_2 и т. д. Процедура повторяется до тех пор, пока не будут обработаны все блоки текста. Таким образом, для всех $i = 1 \ldots n$ (n – число блоков) результат шифрования C_i определяется следующим образом: $C_i = = \text{DES}\ (M_i \oplus C_{i-1})$, где $C_0 = \text{IV}$ – начальное значение шифра, равное начальному вектору (вектору инициализации). Последний 64-битовый блок шифротекста является функцией секретного ключа, начального вектора и каждого бита открытого текста независимо от его длины. Этот блок шифротекста называют кодом аутентификации сообщения (КАС). Код КАС может быть легко проверен получателем, владеющим секретным ключом и начальным вектором, путем повторения процедуры, выполненной отправителем. Посторонний, однако, не может осуществить генерацию КАС, который воспринялся бы получателем как подлинный, чтобы добавить его к ложному сообщению, либо отделить КАС от истинного сообщения для использования его с измененным или ложным сообщением.

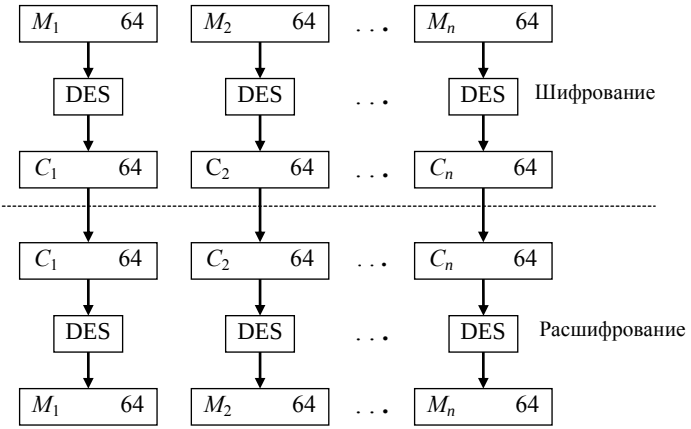

Рисунок 14 – Схема алгоритма DES в режиме электронной кодовой книги

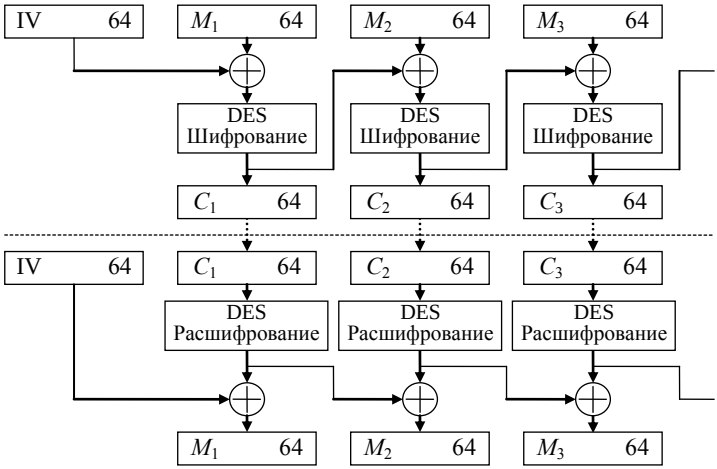

Рисунок 15 – Схема алгоритма DES в режиме сцепления блоков шифра

Достоинство данного режима в том, что он не позволяет накапливаться ошибкам при передаче. Блок M_i является функцией только C_{i-1} и C_i. Поэтому ошибка при передаче приведет к потере только двух блоков исходного текста.

Режим "Обратная связь по шифру". В этом режиме размер блока может отличаться от 64 бит (рисунок 16). Файл, подлежащий шифрованию (расшифрованию), считывается последовательными блоками длиной k битов (k = 1, …, 64). Входной блок (64-битовый регистр сдвига) вначале содержит вектор инициализации, выровненный по правому краю.

Предположим, что в результате разбиения на блоки мы получили n блоков длиной k битов каждый (остаток дописывается нулями или пробелами). Тогда для любого $i = 1 \ldots n$ блок шифротекста $C_i = M_i \oplus P_{i-1}$, где P_{i-1} обозначает k старших битов предыдущего зашифрованного блока. Обновление сдвигового регистра осуществляется путем удаления его старших k битов и записи C_i в регистр. Восстановление зашифрованных данных также выполняется достаточно просто: P_{i-1} и C_i вычисляются аналогичным образом и $M_i = C_i \oplus P_{i-1}$.

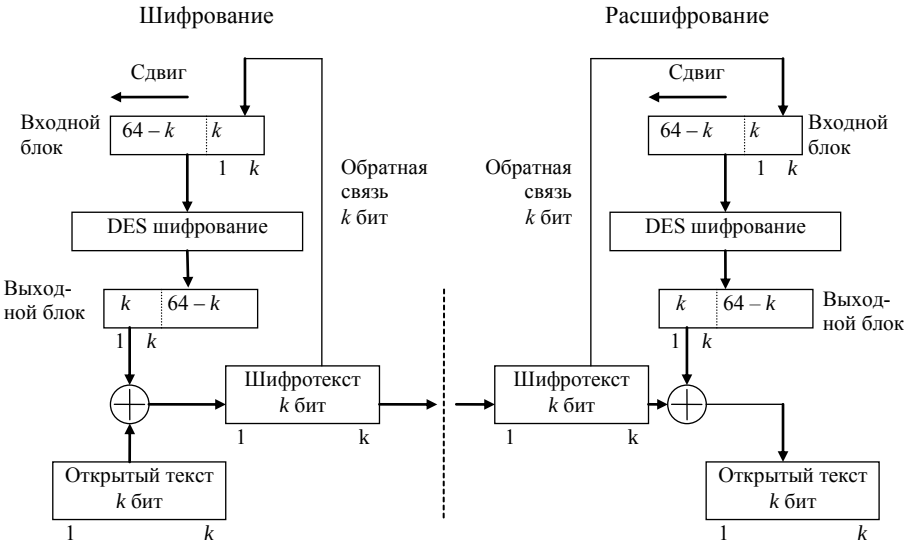

Рисунок 16 – Схема алгоритма DES в режиме обратной связи по шифротексту

Режим "Обратная связь по выходу". Этот режим тоже использует переменный размер блока и сдвиговый регистр, инициализируемый так же, как и в режиме CFB, а именно – входной блок вначале содержит вектор инициализации IV, выровненный по правому краю (рисунок 17).

При этом для каждого сеанса шифрования данных необходимо использовать новое начальное состояние регистра, которое должно пересылаться по каналу открытым текстом. Положим $M = M_1, M_2, ..., M_n$, для всех $i = 1, \ldots, n$, $C_i = M_i \oplus \oplus P_i$, где P_i – старшие k битов операции DES (C_{i-1}). Отличие от режима обратной связи по шифротексту состоит в методе обновления сдвигового регистра. Это осуществляется путем отбрасывания старших k битов и дописывания справа P_i.

Каждому из рассмотренных режимов (ECB, CBC, CFB, OFB) свойственны свои достоинства и недостатки, что обусловливает области их применения.

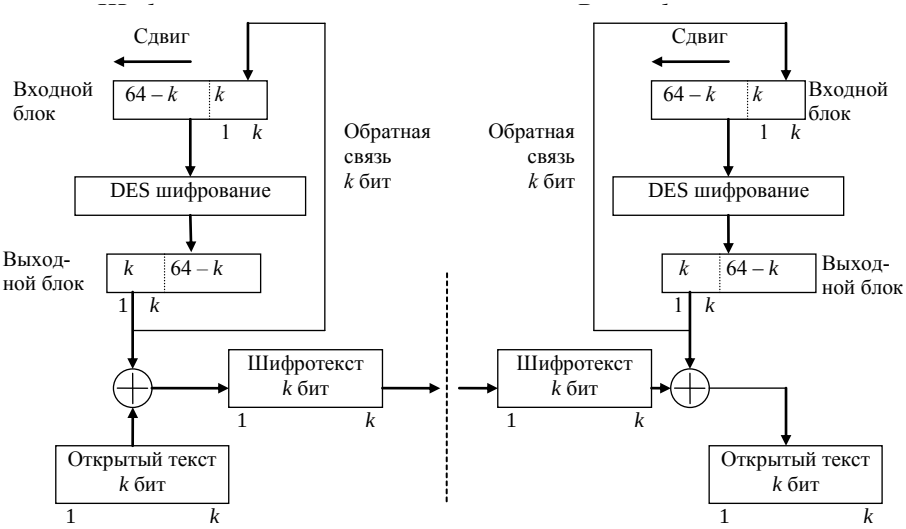

Рисунок 17 – Схема алгоритма DES в режиме обратной связи по выходу

Режим ECB хорошо подходит для шифрования ключей, режим CFB, как правило, предназначается для шифрования отдельных символов, а режим OFB нередко применяется для шифрования в спутниковых системах связи.

Режимы CBC и CFB пригодны для аутентификации данных. Они позволяют использовать алгоритм DES:

– для интерактивного шифрования при обмене данными между терминалом и главной ЭВМ;

– шифрования криптографического ключа в практике автоматизированного распространения ключей;

– шифрования файлов, почтовых отправлений, данных спутников и других практических задач.

С помощью алгоритма DES можно также шифровать файлы ЭВМ для их хранения.

Стандарт шифрования данных (ГОСТ 28147-89). Алгоритм криптографического преобразования данных для систем обработки информации в сетях ЭВМ, отдельных вычислительных комплексах и ЭВМ был разработан в СССР и опубликован в виде государственного стандарта ГОСТ 28147-89 в 1989 году. Алгоритм криптографического преобразования данных предна-

значен для аппаратной и программной реализации, удовлетворяет криптографическим требованиям и не накладывает ограничений на степень секретности защищаемой информации. Алгоритм шифрования данных представляет собой 64-битовый блочный алгоритм с 256-битовым ключом.

При описании алгоритма используются следующие обозначения:

L и R – последовательности битов;

LR – конкатенация последовательностей L и R, в которой биты последовательности R следуют за битами последовательности L;

$\oplus$ – операция побитового сложения по модулю 2;

$\boxplus$ – операция сложения по модулю 2^{32} двух 32-разрядных двоичных чисел;

$\boxplus'$ – операция сложения двух 32-разрядных чисел по модулю $2^{32} - 1$.

Алгоритм предусматривает четыре режима работы:

– шифрование данных в режиме простой замены;

– шифрование данных в режиме гаммирования;

– шифрование данных в режиме гаммирования с обратной связью;

– выработка имитовставки.

Режим простой замены. Для реализации алгоритма шифрования данных в режиме простой замены используется только часть блоков общей криптосистемы (рисунок 18).

Обозначения на схеме:

N_1, N_2 – 32-разрядные накопители;

CM_1 – 32-разрядный сумматор по модулю 2^{32} ($\boxplus$);

CM_2 – 32-разрядный сумматор по модулю 2 ($\oplus$);

R – 32-разрядный регистр циклического сдвига;

КЗУ – ключевое запоминающее устройство на 256 бит, состоящее из восьми 32-разрядных накопителей X_0, X_1, X_2, ..., X_7;

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

Открытые данные, которые подлежат зашифрованию, разбивают на 64-разрядные блоки T_o. Процедура шифрования 64-разрядного блока T_o в режиме простой замены включает 32 цикла (j = 1, …, 32). В ключевое запоминающее устройство вводят ключ K размером 256 бит в виде восьми 32-разрядных подключей (чисел) K_i: $K = K_7K_6K_5K_4K_3K_2K_1K_0$.

Последовательность битов блока T_o = ($a_1(0)$, $a_2(0)$, ..., $a_{31}(0)$, $a_{32}(0)$, $b_1(0)$, $b_2(0)$, ..., $b_{31}(0)$, $b_{32}(0)$) разбивают на две последовательности по 32 бита: $b(0)$ и $a(0)$, где $b(0)$ – левые, или старшие биты, $a(0)$ – правые, или младшие биты. Эти последовательности вводят в накопители N_1 и N_2 перед началом первого цикла шифрования.

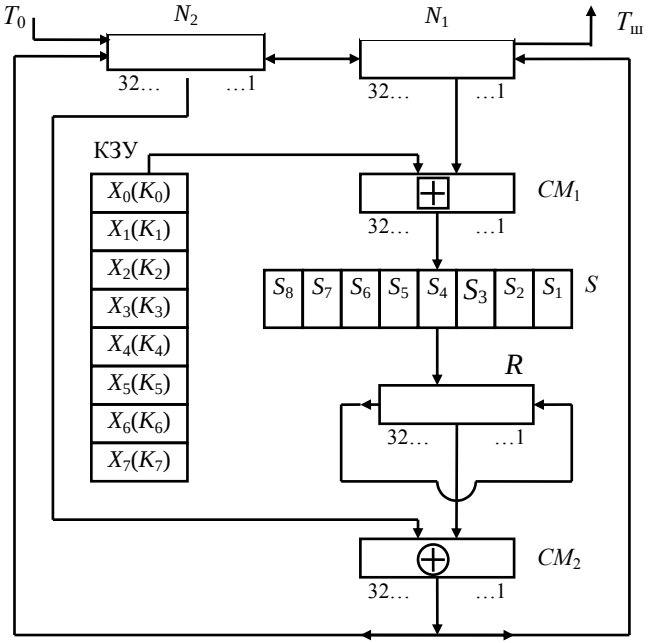

Рисунок 18 – Схема реализации режима простой замены

В результате начальное заполнение накопителя N_1

$$a(0) = (a_{32}(0),\ a_{31}(0),\ ...,\ a_2(0),\ a_1(0)),$$
$$32,\quad 31,\quad\quad 2,\quad 1 \qquad \leftarrow \text{номер разряда } N_1,$$

начальное заполнение накопителя N_2

$$b(0) = (b_{32}(0),\ b_{31}(0),\ ...,\ b_2(0),\ b_1(0)),$$
$$32,\quad 31,\quad\quad 2,\quad 1 \qquad \leftarrow \text{номер разряда } N_2.$$

Первый цикл (j = 1) процедуры шифрования 64-разрядного блока открытых данных можно описать уравнениями:

$$\begin{cases} a(1) = f(a(0) \boxplus K_0) \oplus b(0), \\ b(1) = a(0). \end{cases} \qquad (2)$$

где $a(1)$, $b(1)$ – заполнение соответственно N_1 и N_2 после 1-го цикла шифрования;

f – функция шифрования.

Аргументом функции f является сумма по модулю 2^{32} числа $a(0)$ (начального заполнения накопителя N_1) и числа K_0 – подключа, считывае-

мого из накопителя X_0 КЗУ. Каждое из этих чисел равно 32 битам. Функция f включает две операции над полученной 32-разрядной суммой ($a(0)$ $\boxplus$ K_0).

Первая операция называется *подстановкой* (*заменой*) и выполняется блоком подстановки S, который состоит из восьми узлов замены (S-блоков замены) S_1, S_2, ..., S_8 с памятью 64 бит каждый. Поступающий из CM_1 на блок подстановки S 32-разрядный вектор разбивают на восемь последовательно идущих 4-разрядных векторов, каждый из которых преобразуется в четырехразрядный вектор соответствующим узлом замены. Каждый узел замены можно представить в виде таблицы-перестановки шестнадцати четырехразрядных двоичных чисел в диапазоне 0000…1111. Входной вектор указывает адрес строки в таблице, а число в этой строке является выходным вектором. Затем четырехразрядные выходные векторы последовательно объединяют в 32-разрядный вектор. Узлы замены (таблицы-перестановки) представляют собой ключевые элементы, которые являются общими для сети ЭВМ и редко изменяются. Эти узлы замены должны сохраняться в секрете.

Вторая операция – *циклический сдвиг влево* (на 11 разрядов) 32-разрядного вектора, полученного с выхода блока подстановки S. Циклический сдвиг выполняется регистром сдвига R.

Далее результат работы функции шифрования f суммируют поразрядно по модулю 2 в сумматоре CM_2 с 32-разрядным начальным заполнением $b(0)$ накопителя N_2. Затем полученный на выходе CM_2 результат [значение $a(1)$] записывают в накопитель N_1, а старое значение N_1 [значение $a(0)$] переписывают в накопитель N_2 [значение $b(1) = a(0)$]. Первый цикл завершен.

Последующие циклы осуществляются аналогично, при этом во втором цикле из КЗУ считывают заполнение X_1 – подключ K_1, в третьем – подключ K_2 и т. д., в восьмом – подключ K_7. В циклах с 9-го по 16-й, а также с 17-го по 24-й подключи из КЗУ считываются в том же порядке: K_0, K_1, K_2, ..., K_6, K_7. В последних восьми циклах (с 25-го по 32-й) порядок считывания подключей из КЗУ обратный: K_7, K_6, ..., K_2, K_1, K_0.

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

Уравнения шифрования в режиме простой замены имеют вид:

$$\begin{cases} a(j) = f(a(j-1) \boxplus K_{j-1(\mathrm{mod}\,8)}) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \text{при } j = 1, \ldots, 24, \qquad (3)$$

$$\begin{cases} a(j) = f(a(j-1) \boxplus K_{32-j}) \oplus b(j-1) \\ b(j) = a(j-1) \end{cases} \quad \text{при } j = 25, \ldots, 31, \qquad (4)$$

$$\begin{cases} a(32) = a(31) \\ b(32) = f a(31) \boxplus K_0) \oplus b(31) \end{cases} \quad \text{при } j = 32, \qquad (5)$$

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

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

Остальные блоки открытых данных зашифровываются в режиме простой замены аналогично.

Криптосхема, реализующая алгоритм расшифрования в режиме простой замены, имеет тот же вид, что и при шифровании (см. рисунок 18). В КЗУ вводят 256 бит ключа, на котором осуществлялось шифрование. Зашифрованные данные, подлежащие расшифрованию, разбиты на блоки $T_ш$ по 64 бита в каждом. Ввод любого блока $T_ш = [a_1(32), a_2(32), ..., a_{32}(32), b_1(32), b_2(32), ..., b_{32}(32)]$ в накопители N_1 и N_2 производят так, чтобы начальное значение накопителя N_1 имело вид $a_{32}(32), a_{31}(32), ..., a_2(32), a_1(32)$, а начальное заполнение накопителя N_2 – вид $b_{32}(32), b_{31}(32), ..., b_2(32), b_1(32)$. Расшифрование осуществляется по тому же алгоритму, что и шифрование, с тем изменением, что заполнения накопителей $X_0, X_1, ..., X_7$ считываются из КЗУ в циклах расшифрования в следующем порядке:

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

Уравнения расшифрования имеют вид

$$\begin{cases} a(32-j) = f(a(32\ \ j+1) \boxplus K_{j-1}) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad \text{при } j = 1, \ldots, 8, \quad (6)$$

$$\begin{cases} a(32-j) = f(a(32\ \ j+1) \boxplus K_{32-j(\mathrm{mod}\,8)}) \oplus b(32-j+1) \\ b(32-j) = a(32-j+1) \end{cases} \quad \text{при } j = 9, \ldots, 31, \quad (7)$$

$$\begin{cases} a(0) = a(1) \\ b(0) = f(a(1) \boxplus K_0) \oplus b(1) \end{cases} \quad \text{при } j = 32. \qquad (8)$$

Полученные после 32 циклов работы заполнения накопителей N_1 и N_2 образуют блок открытых данных $T_o = (a_1(0), a_2(0), ..., a_{32}(0), b_1(0), b_2(0),..., b_{32}(0))$, соответствующий блоку зашифрованных данных $T_ш$. При этом состояния накопителей N_1 $[a_{32}(0), a_{31}(0), ..., a_2(0), a_1(0)]$, N_2 $[b_{32}(0), b_{31}(0), ..., b_2(0), b_1(0)]$. Аналогично расшифровываются остальные блоки зашифрованных данных.

Если алгоритм шифрования в режиме простой замены 64-битового блока $T_о$ обозначить через A, то $A(T_о) = A(a\,(0), b(0)) = (a\,(32), b(32)) = T_ш$.

Следует иметь в виду, что режим простой замены допустимо использовать для шифрования данных только в ограниченных случаях – при выработке ключа и шифровании его с обеспечением имитозащиты для передачи по каналам связи или для хранения в памяти ЭВМ.

Р е ж и м г а м м и р о в а н и я . Криптосхема, реализующая алгоритм шифрования в режиме гаммирования, показана на рисунке 19.

Открытые данные разбивают на 64-разрядные блоки $T_0^{(1)}$, $T_0^{(2)}$, ..., $T_0^{(i)}$, ..., $T_0^{(m)}$, где $T_0^{(i)}$ – i-й 64-разрядный блок открытых данных, $i = 1, \ldots, m$, где m определяется объемом шифруемых данных. Эти блоки поочередно зашифровываются в режиме гаммирования путем поразрядного сложения по модулю 2 в сумматоре $СМ_5$ с гаммой шифра $Г_ш$, которая вырабатывается блоками по 64 бита, т. е. $Г_ш = (Г_ш^{(1)}, Г_ш^{(2)}, ..., Г_ш^{(i)}, Г_ш^{(m)})$, где $Г_ш^{(i)}$ – i-й 64-разрядный блок, $i = 1, \ldots, m$. Число двоичных разрядов в блоке $T_о^{(m)}$ может быть меньше 64, при этом неиспользованная для шифрования часть гаммы шифра из блока $Г_ш^{(m)}$ отбрасывается. Уравнение шифрования данных в режиме гаммирования имеет вид $T_ш^{(i)} = T_0^{(i)} \oplus Г_ш^{(i)}$, где $Г_ш^{(i)} = A(Y_{i-1} \boxplus C_2, Z_{i-1} \boxplus ´C_1)$, $i = 1, \ldots, m$; $T_ш^{(i)}$ – i-й блок 64-разрядного блока зашифрованного текста; $A()$ – функция шифрования в режиме простой замены; C_1, C_2 – 32-разрядные двоичные константы; Y_i, Z_i – 32-разрядные двоичные последовательности. Величины Y_i, Z_i определяются итерационно по мере формирования гаммы $Г_ш$ следующим образом: $(Y_0, Z_0) = A(S)$, где S – синхропосылка (64-разрядная двоичная последовательность), $(Y_i, Z_i) = (Y_{i-1} \boxplus C_2, Z_{i-1} \boxplus C_1)$, $i = 1, \ldots, m$.

Рассмотрим реализацию процедуры шифрования в режиме гаммирования. В накопители N_6 и N_5 заранее записаны 32-разрядные двоичные константы C_1 и C_2, имеющие следующие значения (в шестнадцатеричной форме): $C_1 = 01010104_{(16)}$, $C_2 = 01010101_{(16)}$.

В КЗУ вводится 256 бит ключа; в накопители N_1 и N_2 – 64-разрядная двоичная последовательность (синхропосылка) $S = (S_1, S_2, ..., S_{64})$. Синхропосылка S является исходным заполнением накопителей N_1 и N_2 для последовательной выработки m блоков гаммы шифра. Исходное заполнение накопителя N_1 [S_{32}, S_{31}, ..., S_2, S_1], N_2 [S_{64}, S_{63}, ..., S_{34}, S_{33}]. Исходное заполнение N_1 и N_2 (синхропосылка S) зашифровывается в режиме простой замены. Результат шифрования $A(S) = (Y_0, Z_0)$ переписывается в 32-разрядные накопители N_3 и N_4 так, что заполнение N_1 переписывается в N_3, а заполнение N_2 – в N_4. Заполнение накопителя N_4 суммируется по модулю $(2^{32} – 1)$ в сумма-

торе CM_4 с 32-разрядной константой C_1 из накопителя N_6. Результат записывается в N_4. Заполнение накопителя N_3 суммируется по модулю 2^{32} в сумматоре CM_3 с 32-разрядной константой C_2 из накопителя N_5. Результат записывается в N_3. Заполнение N_3 переписывается в N_1, а заполнение N_4 – в N_2, при этом заполнения N_3, N_4 сохраняются. Заполнение накопителей N_1 и N_2 зашифровывается в режиме простой замены.

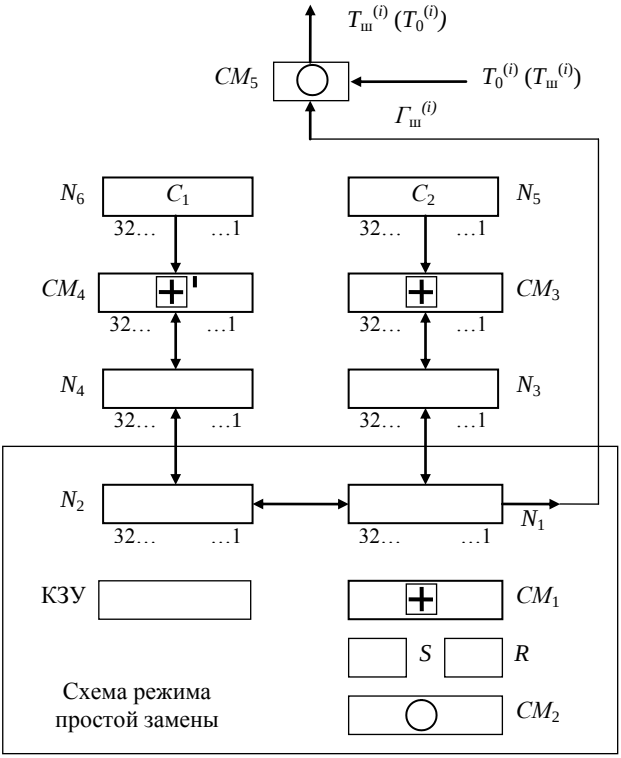

Рисунок 19 – Схема реализации режима гаммирования

Полученное в результате шифрования заполнение накопителей N_1, N_2 образует первый 64-разрядный блок гаммы шифра $Г_ш^{(1)} = (\gamma_1^{(1)}, \gamma_2^{(1)}, ..., \gamma_{63}^{(1)}, \gamma_{64}^{(1)})$, который суммируют поразрядно по модулю 2 в сумматоре CM_5 с первым 64-разрядным блоком открытых данных $T_0^{(1)} = (t_1^{(1)}, t_2^{(1)}, ..., t_{63}^{(1)}, t_{64}^{(1)})$. В результате суммирования по модулю 2 значений $Г_ш^{(1)}$ и $T_0^{(1)}$ получают первый 64-разрядный блок зашифрованных данных: $T_ш^{(1)} = Г_ш^{(1)} \oplus T_0^{(1)} =$

$= (\tau_1^{(1)}, \tau_2^{(1)}, ..., \tau_{63}^{(1)}, \tau_{64}^{(1)})$, где $\tau_i^{(1)} = t_i^{(1)} \oplus \gamma_i^{(1)}$, $i = 1, \ldots, 64$.

Для получения следующего 64-разрядного блока гаммы шифра $\Gamma_{ш}^{(2)}$ заполнение N_4 суммируется по модулю $(2^{32} - 1)$ в сумматоре CM_4 с константой C_1 из N_6. Результат записывается в N_4. Заполнение N_3 суммируется по модулю 2^{32} в сумматоре CM_3 с константой C_2 из N_5. Результат записывается в N_3. Новое заполнение N_3 переписывают в N_1, а новое заполнение N_4 – в N_2, при этом заполнения N_3 и N_4 сохраняют. Заполнения N_1, N_2 зашифровывают в режиме простой замены.

Полученное в результате шифрования заполнение накопителей N_1 и N_2 образует второй 64-разрядный блок гаммы шифра $\Gamma_{ш}^{(2)}$, который суммируется поразрядно по модулю два в сумматоре CM_5 со вторым блоком открытых данных $T_0^{(2)}$: $T_{ш}^{(2)} = \Gamma_{ш}^{(2)} \oplus T_0^{(2)}$.

Аналогично вырабатываются блоки гаммы шифра $\Gamma_{ш}^{(3)}$, $\Gamma_{ш}^{(4)}$, ..., $\Gamma_{ш}^{(m)}$ и зашифровываются блоки открытых данных $T_0^{(3)}$, $T_0^{(4)}$, ..., $T_0^{(m)}$. В канал связи или память ЭВМ передаются синхропосылка S и блоки зашифрованных данных $T_{ш}^{(1)}$, $T_{ш}^{(2)}$, ..., $T_{ш}^{(m)}$.

При расшифровании криптосхема имеет тот же вид, что и при шифровании (см. рисунок 19). Уравнение расшифрования:

$$T_0^{(i)} = T_{ш}^{(i)} \oplus \Gamma_{ш}^{(i)} = T_{ш}^{(i)} \oplus A(Y_{i-1} \boxplus C_2, Z_{i-1} \boxplus' C_1), \quad i = 1, \ldots, m.$$

Расшифрование данных возможно только при наличии синхропосылки, которая не является секретным элементом шифра и может храниться в памяти ЭВМ или передаваться по каналам связи вместе с зашифрованными данными.

При расшифровании в КЗУ вводят 256 бит ключа, с помощью которого осуществлялось шифрование данных $T_0^{(1)}$, $T_0^{(2)}$, ..., $T_0^{(m)}$. В накопители N_1 и N_2 вводится синхропосылка, и осуществляется процесс выработки m блоков гаммы шифра $\Gamma_{ш}^{(1)}$, $\Gamma_{ш}^{(2)}$, ..., $\Gamma_{ш}^{(m)}$. Блоки зашифрованных данных $T_{ш}^{(1)}$, $T_{ш}^{(2)}$, ..., $T_{ш}^{(m)}$ суммируются поразрядно по модулю 2 в сумматоре CM_5 с блоками гаммы шифра $\Gamma_{ш}^{(1)}$, ..., $\Gamma_{ш}^{(m)}$. В результате получаются блоки открытых данных $T_0^{(1)}$, $T_0^{(2)}$, ..., $T_0^{(m)}$, при этом $T_0^{(m)}$ может содержать меньше 64 разрядов.

Режим гаммирования с обратной связью. Криптосхема, реализующая алгоритм шифрования в режиме гаммирования с обратной связью, имеет вид, показанный на рисунке 20.

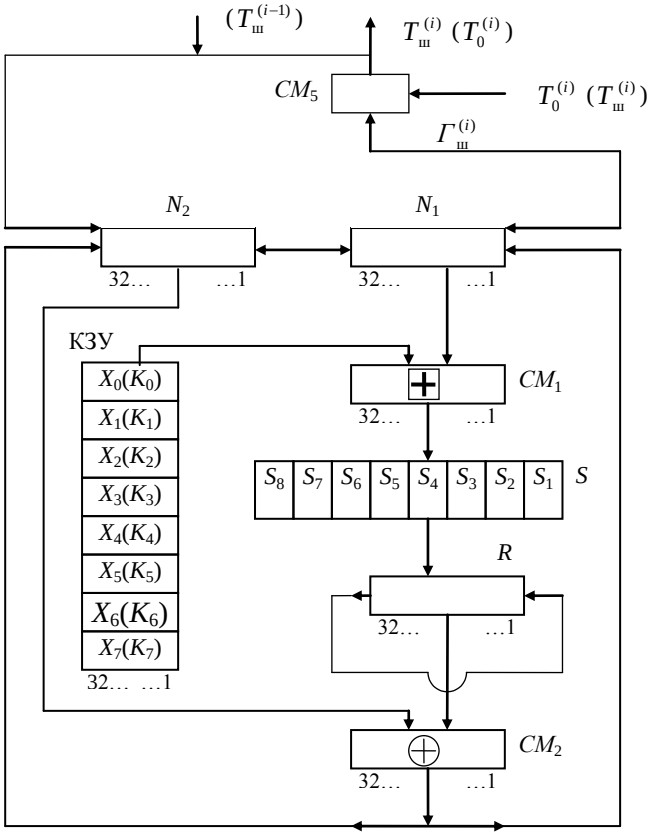

Рисунок 20 – Схема реализации режима гаммирования с обратной связью

Открытые данные, разбитые на 64-разрядные блоки $T_0^{(1)}$, $T_0^{(2)}$, ..., $T_0^{(m)}$, зашифровываются в режиме гаммирования с обратной связью путем поразрядного сложения по модулю 2 с гаммой шифра $Г_{ш}$, которая вырабатывается блоками по 64 бита: $Г_{ш} = (Г_{ш}^{(1)}, Г_{ш}^{(2)}, ..., Г_{ш}^{(m)})$. Число двоичных разрядов в блоке $T_{о}^{(m)}$ может быть меньше 64, при этом не использованная для шифрования часть гаммы шифра из блока $Г_{ш}^{(m)}$ отбрасывается. Уравнения шифрования в режиме гаммирования с обратной связью имеют вид: $T_{ш}^{(1)} = A(S) \oplus T_0^{(1)} = Г_{ш}^{(1)} \oplus T_0^{(1)}$, $T_{ш}^{(i)} = A(T_{ш}^{(i-1)}) \oplus T_0^{(i)} = Г_{ш}^{(i)} \oplus T_0^{(i)}$, $i = 2, \ldots, m$. Здесь $T_{ш}^{(i)}$ – i-й 64-разрядный блок зашифрованного текста; $A()$ – функция шифрования в режиме простой замены; m определяется объемом

открытых данных. Аргументом функции $A()$ на первом шаге итеративного алгоритма является 64-разрядная синхропосылка S, а на всех последующих шагах – предыдущий блок зашифрованных данных $T_{ш}^{(i-1)}$.

Процедура шифрования данных в режиме гаммирования с обратной связью реализуется следующим образом. В КЗУ вводятся 256 бит ключа. В накопители N_1 и N_2 вводится синхропосылка $S = (S_1, S_2, ..., S_{64})$ из 64 бит. Исходное заполнение накопителей N_1 и N_2 зашифровывается в режиме простой замены. Полученное в результате шифрования заполнение накопителей N_1 и N_2 образует первый 64-разрядный блок гаммы шифра $Г_{ш}^{(1)} = A(S)$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 с первым 64-разрядным блоком открытых данных $T_0^{(1)} = (t_1^{(1)}, t_2^{(1)}, \ldots, t_{63}^{(1)}, t_{64}^{(1)})$. В результате получают первый 64-разрядный блок зашифрованных данных $T_{ш}^{(1)} = Г_{ш}^{(1)} \oplus T_0^{(1)}$, где $T_{ш}^{(1)} = (\tau_1^{(1)}, \tau_2^{(1)}, ..., \tau_{63}^{(1)}, \tau_{64}^{(1)})$. Блок зашифрованных данных $T_{ш}^{(1)}$ одновременно является также исходным состоянием накопителей N_1, N_2 для выработки второго блока гаммы шифра $Г_{ш}^{(2)}$, и поэтому по обратной связи $T_{ш}^{(1)}$ записывается в указанные накопители N_1 и N_2. Заполнение накопителей N_1 [$\tau_{32}^{(1)}, \tau_{31}^{(1)}, ..., \tau_2^{(1)}, \tau_1^{(1)}$], N_2 [$\tau_{64}^{(1)}, \tau_{63}^{(1)}, ..., \tau_{34}^{(1)}, \tau_{33}^{(1)}$]. Заполнение накопителей N_1 и N_2 зашифровывается в режиме простой замены. Полученное в результате шифрования заполнение накопителей N_1 и N_2 образует второй 64-разрядный блок гаммы шифра $Г_{ш}^{(2)}$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 со вторым блоком открытых данных $T_0^{(2)}$: $Г_{ш}^{(2)} \oplus T_0^{(2)} = T_{ш}^{(2)}$.

Выработка последующих блоков гаммы шифра $Г_{ш}^{(i)}$ и шифрование соответствующих блоков открытых данных $T_0^{(i)}$ ($i = 3, \ldots, m$) производятся аналогично.

Если длина последнего m-го блока открытых данных $T_0^{(m)}$ меньше 64 разрядов, то из $Г_{ш}^{(m)}$ используется только соответствующее число разрядов гаммы шифра, остальные разряды отбрасываются.

В канал связи или память ЭВМ передаются синхропосылка S и блоки зашифрованных данных $T_{ш}^{(1)}, T_{ш}^{(2)}, ..., T_{ш}^{(m)}$.

При расшифровании криптосхема имеет тот же вид, что и при шифровании (см. рисунок 20). Уравнения расшифрования: $T_0^{(1)} = A(S) \oplus T_{ш}^{(1)} =$

$= \Gamma_{ш}^{(1)} \oplus T_{ш}^{(1)}$, $T_0^{(i)} = \Gamma_{ш}^{(i)} \oplus T_{ш}^{(i)} = A\,(T_{ш}^{(i-1)}) \oplus T_{ш}^{(i)}$, $i = 2, \ldots, m$.

Реализация процедуры расшифрования зашифрованных данных в режиме гаммирования с обратной связью происходит следующим образом. В КЗУ вводят 256 бит того же ключа, на котором осуществлялось шифрование открытых блоков $T_0^{(1)}$, $T_0^{(2)}$, ..., $T_0^{(m)}$. В накопители N_1 и N_2 вводится синхропосылка S. Исходное заполнение накопителей N_1 и N_2 (синхропосылка S) зашифровывается в режиме простой замены. Полученные в результате шифрования заполнение N_1 и N_2 образуют первый блок гаммы шифра $\Gamma_{ш}^{(1)} = A(S)$, который суммируется поразрядно по модулю 2 в сумматоре CM_5 с блоком зашифрованных данных $T_{ш}^{(1)}$. В результате получается первый блок открытых данных $T_0^{(1)} = \Gamma_{ш}^{(1)} \oplus T_{ш}^{(1)}$.

Блок зашифрованных данных $T_{ш}^{(1)}$ является исходным заполнением накопителей N_1 и N_2 для выработки второго блока гаммы шифра $\Gamma_{ш}^{(2)}$: $\Gamma_{ш}^{(2)} = A(T_{ш}^{(1)})$. Полученное заполнение накопителей N_1 и N_2 зашифровывается в режиме простой замены. Образованный в результате шифрования блок $\Gamma_{ш}^{(2)}$ суммируется поразрядно по модулю два в сумматоре CM_5 со вторым блоком зашифрованных данных $T_{ш}^{(2)}$. В результате получают второй блок открытых данных. Аналогично в N_1, N_2 последовательно записывают блоки зашифрованных данных $T_{ш}^{(2)}$, $T_{ш}^{(3)}$, ..., $T_{ш}^{(m)}$, из которых в режиме простой замены вырабатываются блоки гаммы шифра $\Gamma_{ш}^{(3)}$, $\Gamma_{ш}^{(4)}$, ..., $\Gamma_{ш}^{(m)}$. Блоки гаммы шифра суммируются поразрядно по модулю 2 в сумматоре CM_5 с блоками зашифрованных данных $T_{ш}^{(3)}$, $T_{ш}^{(4)}$, ..., $T_{ш}^{(m)}$. В результате получают блоки открытых данных $T_0^{(3)}$, $T_0^{(4)}$, ..., $T_0^{(m)}$, при этом последний блок открытых данных $T_0^{(m)}$ может содержать меньше 64 разрядов.

Р е ж и м в ы р а б о т к и и м и т о в с т а в к и. Имитовставка – это блок из P бит, который вырабатывают по определенному правилу из открытых данных с использованием ключа и затем добавляют к зашифрованным данным для обеспечения их имитозащиты. Имитозащита – это защита системы шифрованной связи от навязывания ложных данных.

В стандарте ГОСТ 28147-89 определяется процесс выработки имитовставки, который единообразен для любого из режимов шифрования данных. Имитовставка $И_p$ вырабатывается из блоков открытых данных либо

перед шифрованием всего сообщения, либо параллельно с шифрованием по блокам. Первые блоки открытых данных, которые участвуют в выработке имитовставки, могут содержать служебную информацию (например, адресную часть, время, синхропосылку) и не зашифровываются. Значение параметра P (число двоичных разрядов в имитовставке) определяется криптографическими требованиями с учетом того, что вероятность навязывания ложных помех равна $1/2^p$.

Для выработки имитовставки открытые данные представляют в виде последовательности 64-разрядных блоков $T_0^{(i)}$, $i = 1, \ldots, m$. Первый блок открытых данных $T_0^{(1)}$ подвергают преобразованию $A()$, соответствующему первым 16 циклам алгоритма шифрования в режиме простой замены. В качестве ключа для выработки имитовставки используют ключ длиной 256 бит, по которому шифруют данные. Полученное после 16 циклов 64-разрядное число $A(T_0^{(1)})$ суммируют по модулю 2 со вторым блоком открытых данных $T_0^{(2)}$. Результат суммирования $[A(T_0^{(1)}) \oplus T_0^{(2)}]$ снова подвергают преобразованию $A()$. Полученное 64-разрядное число $A[A(T_0^{(1)}) \oplus T_0^{(2)}]$ суммируют по модулю 2 с третьим блоком $T_0^{(3)}$ и снова подвергают преобразованию $A()$, получая 64-разрядное число $A[A(A(T_0^{(1)}) \oplus T_0^{(2)}) \oplus T_0^{(3)}]$, и т. д.

Последний блок $T_0^{(m)}$ (при необходимости дополненный нулями до полного 64-разрядного блока) суммируют по модулю 2 с результатом вычислений на шаге $(m - 1)$, после чего зашифровывают в режиме простой замены, используя преобразование $A()$. Из полученного 64-разрядного числа выбирают отрезок $И_p$ (имитовставку) длиной P бит: $И_p = [a^{(m)}{}_{32-p+1}(16), a^{(m)}{}_{32-p+2}(16), \ldots, a^{(m)}{}_{32}(16)]$, где $a_i^{(m)}$ – i-й бит 64-разрядного числа, полученного после 16-го цикла последнего преобразования $A()$, $32 - p + 1 \le i \le 32$. Имитовставка $И_p$ передается по каналу связи или в память ЭВМ в конце зашифрованных данных, т. е. $T_{ш}^{(1)}, T_{ш}^{(2)}, \ldots, T_{ш}^{(m)}, И_p$.

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

ных $T_0^{(1)}$, $T_0^{(2)}$, ..., $T_0^{(m)}$ считают ложными.

1.3 Асимметричные криптосистемы

Криптосистема шифрования данных RSA. Алгоритм RSA предложили в 1978 г. три автора: Р. Райвест (Rivest), А. Шамир (Shamir) и А. Адлеман (Adleman). Алгоритм получил свое название по первым буквам фамилий его авторов.

Надежность алгоритма основывается на трудности факторизации больших чисел в произведение простых множителей. В криптосистеме RSA открытый ключ $K_в$, секретный ключ $k_в$, сообщение M и криптограмма C принадлежат множеству целых чисел $Z_N = \{0, 1, 2, ..., N-1\}$, где N – модуль $N = PQ$. Здесь P и Q – случайные большие простые числа. Для обеспечения максимальной безопасности выбирают P и Q равной длины и хранят в секрете. Множество Z_N с операциями сложения и умножения по модулю N образует арифметику по модулю N. Открытый ключ $K_в$ выбирают случайным образом так, чтобы выполнялись условия: $1 < K_в \le \varphi(N)$, НОД($K_в$, $\varphi(N)$) = 1, $\varphi(N) = (P-1)(Q-1)$, где $\varphi(N)$ – функция Эйлера. Второе из указанных выше условий означает, что открытый ключ $K_в$ и значение функции Эйлера $\varphi(N)$ должны быть взаимно простыми. Далее, используя расширенный алгоритм Евклида, вычисляют секретный ключ $k_в$, такой, что

$$k_в K_в \equiv 1 \ [\text{mod } \varphi(N)] \text{ или } k_в = K_в^{-1} \ [\text{mod}(P-1)(Q-1)]. \qquad (9)$$

Это можно осуществить, так как при известной паре простых чисел (P, Q) можно легко найти $\varphi(N)$. Заметим, что $k_в$ и N должны быть взаимно простыми. Открытый ключ $K_в$ используют для шифрования данных, а секретный ключ $k_в$ – для расшифрования.

Преобразование шифрования определяет криптограмму C через пару «открытый ключ $K_в$, сообщение M» в соответствии со следующей формулой:

$$C = E_{K_в}(M) = E_в(M) = M^{K_в} \ (\text{mod } N). \qquad (10)$$

В качестве алгоритма быстрого вычисления значения C используют ряд последовательных возведений в квадрат целого M и умножений на M с приведением по модулю N.

Обращение функции $C = M^{K_в} \ (\text{mod } N)$, т. е. определение значения M по известным значениям C, $K_в$ и N, практически не осуществимо при $N \approx 2^{512}$. Однако обратную задачу, т. е. задачу расшифрования криптограммы C, можно решить, используя пару «секретный ключ $k_в$, криптограмма C» по следующей формуле:

$$M = D_{K_в}(C) = D_в(C) = C^{k_в} \pmod N. \qquad (11)$$

Подставляя в это уравнение значение C, получаем

$$\left(M^{K_в}\right)^{k_в} = M \pmod N \text{ или } M^{K_в k_в} = M \pmod N. \qquad (12)$$

Сопоставляя это выражение с формулой расшифрования, получаем $K_в k_в = n\,\varphi(N) + 1$ или, что то же самое, $K_в k_в \equiv 1[\mathrm{mod}\ \varphi(N)]$.

Следовательно, если криптограмму $C = M^{K_в} \pmod N$ возвести в степень $k_в$, то в результате восстанавливается исходный открытый текст M, так как

$$\left(M^{K_в}\right)^{k_в} = M^{K_в k_в} = M^{n\varphi(N)+1} \equiv M \pmod N. \qquad (13)$$

Таким образом, получатель, который создает криптосистему, защищает два параметра: 1) секретный ключ $k_в$ и 2) пару чисел (P, Q), произведение которых дает значение модуля N. С другой стороны, отправителю известны значения модуля N и открытого ключа $K_в$. Противнику известны лишь значения $K_в$ и N. Если бы он смог разложить число N на множители P и Q, то он узнал бы тройку чисел $\{P, Q, K_в\}$, вычислил бы значение функции Эйлера $\varphi(N) = (P-1)(Q-1)$ и определил значение секретного ключа $k_в$.

Однако, как уже отмечалось, разложение очень большого N на множители вычислительно не осуществимо (при условии, что длины выбранных P и Q составляют не менее 100 десятичных знаков).

Предположим, что пользователь A хочет передать пользователю B сообщение в зашифрованном виде, используя криптосистему RSA. В таком случае пользователь A выступает в роли отправителя сообщения, а пользователь B – в роли получателя. Криптосистему RSA должен сформировать получатель сообщения, т. е. пользователь B. Рассмотрим последовательность действий пользователей B и A при передаче сообщения «ГБВ» и при использовании небольших чисел для простоты вычислений, хотя на практике применяются очень большие числа.

1 Пользователь B выбирает два произвольных простых числа P и Q: $P = 3$ и $Q = 11$.

2 Пользователь B вычисляет значение модуля $N = PQ$: $N = 3 \cdot 11 = 33$.

3 Пользователь B вычисляет значение функции Эйлера $\varphi(N) = (P-1)(Q-1)$ и выбирает случайным образом значение открытого ключа $K_в$ с учетом выполнения условий: $1 < K_в \le \varphi(N)$, НОД($K_в$, $\varphi(N)$) = 1.

$$\varphi(N) = \varphi(33) = (P-1)(Q-1) = 2 \cdot 10 = 20;\ K_в = 7.$$

4 Пользователь B вычисляет значение секретного ключа $k_в$, используя

расширенный алгоритм Евклида при решении $k_в \equiv K_в^{-1} \pmod{\varphi(N)}$. $k_в \equiv 7^{-1} \pmod{20}$. Решение дает $k_в = 3$.

5 Пользователь *B* пересылает пользователю *A* пару чисел (N, $K_в$) по незащищенному каналу.

6 Пользователь *A* разбивает исходный открытый текст *M* на блоки, каждый из которых может быть представлен в виде числа $M_i = 0, 1, 2, ..., N - 1$. Для этого он представляет шифруемое сообщение «ГБВ» как последовательность целых чисел в диапазоне от 0 до 32. Например, пусть буква Б представляется как число 1, буква В – как число 2, буква Г – как число 3. Тогда сообщение «ГБВ» можно представить как последовательность чисел 312, т. е. $M_1 = 3$, $M_2 = 1$, $M_3 = 2$.

7 Пользователь *A* шифрует текст, представленный в виде последовательности чисел M_i, по формуле $C_i = M_i^{K_в} \pmod{N}$:

$$C_1 = 3^7 \pmod{33} = 2187 \pmod{33} = 9,$$
$$C_2 = 1^7 \pmod{33} = 1 \pmod{33} = 1,$$
$$C_3 = 2^7 \pmod{33} = 128 \pmod{33} = 29,$$

и отправляет криптограмму $C_1, C_2, C_3, ..., C_i$ пользователю *B*. $C_1, C_2, C_3 = 9, 1, 29$.

8 Пользователь *B* расшифровывает принятую криптограмму $C_1, C_2, C_3, ..., C_i$, используя секретный ключ $k_в$, по формуле $M_i = C_i^{k_в} \pmod{N}$:

$$M_1 = 9^3 \pmod{33} = 729 \pmod{33} = 3,$$
$$M_2 = 1^3 \pmod{33} = 1 \pmod{33} = 1,$$
$$M_3 = 29^3 \pmod{33} = 24389 \pmod{33} = 2.$$

В результате будет получена последовательность чисел M_i, которые представляют собой исходное сообщение *M*. Таким образом, восстановлено исходное сообщение: «ГБВ».

Чтобы алгоритм RSA имел практическую ценность, необходимо иметь возможность без существенных затрат генерировать большие простые числа, уметь оперативно вычислять значения ключей $K_в$ и $k_в$.

Криптосистемы RSA реализуются как аппаратным, так и программным путем. Для аппаратной реализации операций шифрования и расшифрования RSA разработаны специальные процессоры. Эти процессоры, реализованные на сверхбольших интегральных схемах (СБИС), позволяют выполнять операции RSA, связанные с возведением больших чисел в колоссально большую степень по модулю *N*, за относительно короткое время. Аппаратная реализация RSA примерно в 1000 раз медленнее аппаратной реализации симметричного криптоалгоритма DES. Программная реализация RSA примерно в 100 раз медленнее программной реализации DES. С развитием тех-

нологии эти оценки могут несколько изменяться, но асимметричная криптосистема RSA никогда не достигнет быстродействия симметричных криптосистем. Малое быстродействие криптосистем RSA ограничивает область их применения, но не уменьшает их ценность.

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

Для того чтобы генерировать пару ключей (открытый ключ – секретный ключ), сначала выбирают некоторое большое простое число P и большое целое число G, причем $G < P$. Числа P и G могут быть распространены среди группы пользователей. Затем выбирают ключ X – случайное целое число, причем $X < P$. Число X является секретным ключом и должно храниться в секрете. Далее вычисляют $Y = G^X \bmod P$. Число Y является открытым ключом.

Для того чтобы зашифровать сообщение M, выбирают случайное целое число $1 < K < P - 1$, такое, что числа K и $(P - 1)$ являются взаимно простыми. Затем вычисляют числа $a = G^K \bmod P$, $b = Y^K M \bmod P$. Пара чисел (a, b) является шифротекстом. Длина шифротекста вдвое больше длины исходного открытого текста M.

Для того чтобы расшифровать шифротекст (a, b), вычисляют

$$M = b/a^X \bmod P. \qquad (14)$$

Для примера выберем $P = 11$, $G = 2$, секретный ключ $X = 8$.

Вычисляем $Y = G^X \bmod P = 2^8 \bmod 11 = 256 \bmod 11 = 3$. Открытый ключ $Y = 3$.

Пусть сообщение $M = 5$. Выберем некоторое случайное число $K = 9$. Убедимся, что НОД $(K, P - 1) = 1$. НОД $(9, 10) = 1$. Вычисляем пару чисел a и b: $a = G^K \bmod P = 2^9 \bmod 11 = 512 \bmod 11 = 6$; $b = Y^K M \bmod P = 3^9 \cdot 5 \bmod 11 = 19683 \cdot 5 \bmod 11 = 9$. Получим шифротекст $(a, b) = (6, 9)$.

Выполним расшифрование этого шифротекста. Вычисляем сообщение M, используя секретный ключ X: $M = b/a^X \bmod P = 9/6^8 \bmod 11$. Выражение $M = 9/6^8 \bmod 11$ можно представить в виде $6^8 \cdot M \equiv 9 \bmod 11$ или $1679616 \times M \equiv 9 \bmod 11$. В результате находим $M = 5$.

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

Комбинированный метод шифрования. Главным достоинством криптосистем с открытым ключом является их потенциально высокая безопасность: нет необходимости ни передавать, ни сообщать кому бы то ни было значения секретных ключей, ни убеждаться в их подлинности. В симметричных криптосистемах существует опасность раскрытия секретного ключа во время передачи. Однако алгоритмы, лежащие в основе криптосистем с

открытым ключом, имеют следующие недостатки:

– генерация новых секретных и открытых ключей основана на генерации новых больших простых чисел, а проверка простоты чисел занимает много процессорного времени;

– процедуры шифрования и расшифрования, связанные с возведением в степень многозначного числа, достаточно громоздки.

Именно поэтому быстродействие криптосистем с открытым ключом обычно в сотни и более раз меньше быстродействия симметричных криптосистем с секретным ключом.

Комбинированный (гибридный) метод шифрования позволяет сочетать преимущества высокой секретности, присущие асимметричным криптосистемам с открытым ключом, с преимуществами высокой скорости работы, присущими симметричным криптосистемам с секретным ключом. При таком подходе криптосистема с открытым ключом применяется для шифрования, передачи и последующего расшифрования только секретного ключа симметричной криптосистемы. А симметричная криптосистема применяется для шифрования и передачи исходного открытого текста. В результате криптосистема с открытым ключом не заменяет симметричную криптосистему с секретным ключом, а лишь дополняет ее, позволяя повысить в целом защищенность передаваемой информации. Такой подход иногда называют схемой электронного цифрового конверта.

Если пользователь A хочет передать зашифрованное комбинированным методом сообщение M пользователю B, то порядок его действий будет таков:

1 Создать (например, сгенерировать случайным образом) симметричный ключ, называемый в этом методе сеансовым ключом $K_с$.

2 Зашифровать сообщение M на сеансовом ключе $K_с$.

3 Зашифровать сеансовый ключ $K_с$ на открытом ключе $K_в$ пользователя B.

4 Передать по открытому каналу связи в адрес пользователя B зашифрованное сообщение вместе с зашифрованным сеансовым ключом.

Действия пользователя B при получении зашифрованного сообщения и зашифрованного сеансового ключа должны быть обратными:

5 Расшифровать на своем секретном ключе $k_в$ сеансовый ключ $K_с$.

6 С помощью полученного сеансового ключа $K_с$ расшифровать и прочитать сообщение M.

При использовании комбинированного метода шифрования можно быть уверенным в том, что только пользователь В сможет правильно расшифровать ключ $K_с$ и прочитать сообщение M.

Комбинированный метод шифрования является наиболее рациональным, объединяя в себе высокое быстродействие симметричного шифрования и высокую криптостойкость, гарантируемую системами с открытым ключом.

1.4 Электронная цифровая подпись

При обмене электронными документами по сети связи возникает проблема аутентификации автора документа и самого документа, т. е. установления подлинности автора и отсутствия изменений в полученном документе. В обычной (бумажной) информатике эти проблемы решаются за счет того, что информация в документе и рукописная подпись автора жестко связаны с физическим носителем (бумагой). В электронных документах на машинных носителях такой связи нет.

Электронная цифровая подпись (ЭЦП) используется для аутентификации текстов, передаваемых по телекоммуникационным каналам. Функционально она аналогична обычной рукописной подписи и обладает ее основными достоинствами:

– удостоверяет, что подписанный текст исходит от лица, поставившего подпись;

– не дает самому этому лицу возможности отказаться от обязательств, связанных с подписанным текстом;

– гарантирует целостность подписанного текста.

Цифровая подпись представляет собой относительно небольшое количество дополнительной цифровой информации, передаваемой вместе с подписываемым текстом. Система ЭЦП включает две процедуры: 1) постановки подписи; 2) проверки подписи. В процедуре постановки подписи используется секретный ключ отправителя сообщения, в процедуре проверки подписи – открытый ключ отправителя.

При формировании ЭЦП отправитель прежде всего вычисляет хэш-функцию $h(M)$ подписываемого текста M. Вычисленное значение хэш-функции $h(M)$ представляет собой один короткий блок информации m, характеризующий весь текст M в целом. Хэш-функция предназначена для сжатия подписываемого документа M до нескольких десятков или сотен бит. Хэш-функция $h(M)$ принимает в качестве аргумента сообщение (документ) M произвольной длины и возвращает хэш-значение $h(M) = m$ фиксированной длины. Обычно хэшированная информация является сжатым двоичным представлением основного сообщения произвольной длины. Значение хэш-функции $h(M)$ сложным образом зависит от документа M и не позволяет восстановить сам документ M.

Затем число m шифруется секретным ключом отправителя. Получаемая при этом пара чисел представляет собой ЭЦП для данного текста M.

При проверке ЭЦП получатель сообщения снова вычисляет хэш-функцию $m = h(M)$ принятого по каналу сообщения M, после чего при помощи открытого ключа отправителя проверяет, соответствует ли полученная подпись вычисленному значению m хэш-функции.

Функция $h(M)$ – является хэш-функцией, если она удовлетворяет следующим условиям:

– исходный текст может быть произвольной длины;

– само значение $h(M)$ имеет фиксированную длину;

– значение функции $h(M)$ легко вычисляется для любого аргумента;

– восстановить аргумент по значению с вычислительной точки зрения – практически невозможно;

– функция $h(M)$ – однозначна.

Из определения следует, что для любой хэш-функции есть тексты-близнецы – имеющие одинаковое значение хэш-функции, так как мощность множества аргументов неограниченно больше мощности множества значений. Такой факт получил название «эффект дня рождения».

Наиболее известные из хэш-функций – MD2, MD4, MD5 и SHA.

Три алгоритма серии MD разработаны Ривестом в 1989-м, 90-м и 91-м годах соответственно. Все они преобразуют текст произвольной длины в 128-битную сигнатуру.

Алгоритм MD2 предполагает:

– дополнение текста до длины, кратной 128 битам;

– вычисление 16-битной контрольной суммы (старшие разряды отбрасываются);

– добавление контрольной суммы к тексту;

– повторное вычисление контрольной суммы.

Алгоритм MD4 предусматривает:

– дополнение текста до длины, равной 448 бит по модулю 512;

– добавление длины текста в 64-битном представлении;

– использование процедуры Damgard-Merkle с 512-битными блоками (в отличие от хэш-функции этот класс преобразований предполагает вычисление для аргументов фиксированной длины также фиксированных по длине значений), причем каждый блок участвует в трех разных циклах.

В алгоритме MD4 довольно быстро были найдены «дыры», поэтому он был заменен алгоритмом MD5, в котором каждый блок участвует не в трех, а в четырех различных циклах.

Алгоритм SHA (Secure Hash Algorithm) разработан NIST (National Institute of Standard and Technology) и повторяет идеи серии MD. В SHA используются тексты более 2^{64} бит, которые закрываются сигнатурой длиной 160 бит.

Принципиальным моментом в системе ЭЦП является невозможность подделки ЭЦП пользователя без знания его секретного ключа подписывания. В качестве подписываемого документа может быть использован любой файл. Подписанный файл создается из неподписанного путем добавления в него одной или более электронных подписей. Каждая подпись содержит следующую информацию:

– дату подписи;

– срок окончания действия ключа данной подписи;

– информацию о лице, подписавшем файл (Ф.И.О., должность, краткое наименование фирмы);

– идентификатор подписавшего (имя открытого ключа);

– собственно цифровую подпись.

Технология применения системы ЭЦП предполагает наличие сети абонентов, посылающих друг другу подписанные электронные документы. Для каждого абонента генерируется пара ключей: секретный и открытый. Секретный ключ хранится абонентом в тайне и используется им для формирования ЭЦП. Открытый ключ известен всем другим пользователям и предназначен для проверки ЭЦП получателем подписанного электронного документа. Иначе говоря, открытый ключ является необходимым инструментом, позволяющим проверить подлинность электронного документа и автора подписи. Открытый ключ не позволяет вычислить секретный ключ.

Для генерации пары ключей (секретного и открытого) в алгоритмах ЭЦП, как и в асимметричных системах шифрования, используются разные математические схемы, основанные на применении однонаправленных функций. Эти схемы разделяются на две группы. В основе такого разделения лежат известные сложные вычислительные задачи:

– факторизации (разложения на множители) больших целых чисел;

– дискретного логарифмирования.

Алгоритм электронной цифровой подписи RSA. Первой и наиболее известной во всем мире конкретной системой ЭЦП стала система RSA. Согласно этому алгоритму, сначала необходимо вычислить пару ключей (секретный ключ и открытый ключ). Для этого отправитель (автор) электронных документов вычисляет два больших простых числа – P и Q, затем находит их произведение $N = PQ$ и значение функции $\varphi(N) = (P - 1)(Q - 1)$. Далее отправитель вычисляет число E из условий: $E \le \varphi(N)$, НОД$[E, \varphi(N)] = 1$ и число D из условий: $D < N$, $ED \equiv 1 \ [\text{mod } \varphi(N)]$. Пара чисел (E, N) является открытым ключом. Эту пару чисел автор передает партнерам по переписке для проверки его цифровых подписей. Число D сохраняется автором как секретный ключ для подписывания.

Допустим, что отправитель хочет подписать сообщение M перед его отправкой. Сначала сообщение M (блок информации, файл, таблица) сжимают с помощью хэш-функции $h(M)$ в целое число m: $m = h(M)$. Затем вычисляют цифровую подпись S под электронным документом M, используя хэш-значение m и секретный ключ D: $S = m^D \ (\text{mod } N)$.

Пара (M, S) передается партнеру-получателю как электронный документ M, подписанный цифровой подписью S, причем подпись S сформирована обладателем секретного ключа D.

После приема пары (M, S) получатель вычисляет хэш-значение сообще-

ния M двумя разными способами. Прежде всего он восстанавливает хэш-значение m', применяя криптографическое преобразование подписи S с использованием открытого ключа E: $m' = S^E \pmod N$. Кроме того, он находит результат хэширования принятого сообщения M с помощью такой же хэш-функции $h(M)$: $m = h(M)$.

Если соблюдается равенство вычисленных значений, т. е. $S^E \pmod N = h(M)$, то получатель признает пару (M, S) подлинной. Доказано, что только обладатель секретного ключа D может сформировать цифровую подпись S по документу M, а определить секретное число D по открытому числу E не легче, чем разложить модуль N на множители. Кроме того, можно строго математически доказать, что результат проверки цифровой подписи S будет положительным только в том случае, если при вычислении S был использован секретный ключ D, соответствующий открытому ключу E. Поэтому открытый ключ E иногда называют "идентификатором" подписавшего.

К недостаткам алгоритма электронной цифровой подписи RSA можно отнести следующее:

1 При вычислении модуля N, ключей E и D для системы цифровой подписи RSA необходимо проверять большое количество дополнительных условий, что сделать практически трудно. Невыполнение любого из этих условий делает возможным фальсификацию цифровой подписи со стороны того, кто обнаружит такое невыполнение. При подписании важных документов нельзя допускать такую возможность даже теоретически.

2 Для обеспечения криптостойкости цифровой подписи RSA по отношению к попыткам фальсификации на уровне, например, национального стандарта США шифрования информации (алгоритм DES), т. е. 10^{18}, необходимо использовать при вычислениях N, D и E целые числа не менее 2^{512} (или около 10^{154}) каждое, что требует больших вычислительных затрат, превышающих на 20–30 % вычислительные затраты других алгоритмов цифровой подписи при сохранении того же уровня криптостойкости.

3 Цифровая подпись RSA уязвима к так называемой мультипликативной атаке. Иначе говоря, алгоритм цифровой подписи RSA позволяет противнику без знания секретного ключа D сформировать подписи под теми документами, у которых результат хэширования можно вычислить как произведение результатов хэширования уже подписанных документов.

Алгоритм электронной цифровой подписи Эль Гамаля (EGSA). Название EGSA происходит от слов El Gamal Signature Algorithm (алгоритм цифровой подписи Эль Гамаля). Идея EGSA основана на том, что для обоснования практической невозможности фальсификации цифровой подписи может быть использована более сложная вычислительная задача, чем разложение на множители большого целого числа, – задача дискретного логарифмирования. Кроме того, Эль Гамалю удалось избежать явной слабости алгоритма цифровой подписи RSA, связанной с возможностью подделки

цифровой подписи под некоторыми сообщениями без определения секретного ключа.

Для того чтобы генерировать пару ключей (открытый ключ – секретный ключ), сначала выбирают некоторое большое простое целое число P и большое целое число G, причем $G < P$. Отправитель и получатель подписанного документа используют при вычислениях одинаковые большие целые числа P ($\sim 10^{308}$ или $\sim 2^{1024}$) и G ($\sim 10^{154}$ или $\sim 2^{512}$), которые не являются секретными.

Отправитель выбирает случайное целое число X, $1 < X \le (P - 1)$ и вычисляет $Y = G^X \bmod P$. Число Y является открытым ключом, используемым для проверки подписи отправителя. Число Y открыто передается всем потенциальным получателям документов. Число X является секретным ключом отправителя для подписывания документов и должно храниться в секрете.

Для того чтобы подписать сообщение M, сначала отправитель хэширует его с помощью хэш-функции $h()$ в целое число m: $m = h(M)$, $1 < m < (P - 1)$ и генерирует случайное целое число K, $1 < K < (P - 1)$ такое, что K и $(P - 1)$ являются взаимно простыми. Затем отправитель вычисляет целое число a: $a = G^K \bmod P$ и, применяя расширенный алгоритм Евклида, вычисляет с помощью секретного ключа X целое число b из уравнения $m = Xa + Kb \pmod{(P - 1)}$.

Пара чисел (a, b) образует цифровую подпись S, проставляемую под документом M. Тройка чисел (M, a, b) передается получателю, в то время как пара чисел (X, K) держится в секрете.

После приема подписанного сообщения (M, a, b) получатель должен проверить, соответствует ли подпись $S = (a, b)$ сообщению M. Для этого получатель сначала вычисляет по принятому сообщению M число $m = h(M)$, т. е. хэширует принятое сообщение.

Затем получатель вычисляет значение $A = Y^a a^b \pmod P$ и признает сообщение M подлинным, если $A = G^m \pmod P$. Иначе говоря, получатель проверяет справедливость соотношения $Y^a a^b \pmod P = G^m \pmod P$.

Можно строго математически доказать, что последнее равенство будет выполняться тогда, и только тогда, когда подпись $S = (a, b)$ под документом M получена с помощью именно того секретного ключа X, из которого был получен открытый ключ Y. Таким образом, можно надежно удостовериться, что отправителем сообщения M был обладатель именно данного секретного ключа X, не раскрывая при этом сам ключ, и что отправитель подписал именно этот конкретный документ M.

Выполнение каждой подписи по методу Эль Гамаля требует нового значения K, причем это значение должно выбираться случайным образом. Если нарушитель раскроет когда-либо значение K, повторно используемое отправителем, то он сможет раскрыть секретный ключ X отправителя.

Для примера выберем числа $P = 11$, $G = 2$ и секретный ключ $X = 8$. Вычислим значение открытого ключа: $Y = G^X \bmod P = Y = 2^8 \bmod 11 = 3$.

Предположим, что исходное сообщение M характеризуется хэш-значением $m = 5$. Для того чтобы вычислить цифровую подпись для сообщения M, имеющего хэш-значение $m = 5$, сначала выберем случайное целое число $K = 9$. Убедимся, что числа K и $(P - 1)$ являются взаимно простыми. Действительно, НОД (9, 10) = 1.

Далее вычисляем элементы a и b подписи: $a = G^K \bmod P = 2^9 \bmod 11 = 6$, элемент b определяем, используя расширенный алгоритм Евклида:

$$m = Xa + Kb \ [\bmod (P - 1)]. \qquad (15)$$

При $m = 5$, $a = 6$, $X = 8$, $K = 9$, $P = 11$ получаем $5 = (6\times8 + 9\times b) (\bmod 10)$ или $9\times b \equiv -43 \pmod{10}$. Решив данное линейное уравнение, получим: $b = 3$. Цифровая подпись представляет собой пару: $a = 6$, $b = 3$.

Далее отправитель передает подписанное сообщение. Приняв подписанное сообщение и открытый ключ $Y = 3$, получатель вычисляет хэш-значение для сообщения M: $m = 5$, а затем вычисляет два числа:

1) $Y^a a^b \pmod P = 3^6\times6^3 \pmod{11} = 10 \pmod{11}$;

2) $G^m \pmod P = 2^5 \pmod{11} = 10 \pmod{11}$.

Так как эти два целых числа равны, принятое получателем сообщение признается подлинным.

Схема Эль Гамаля является характерным примером подхода, который допускает пересылку сообщения M в открытой форме вместе с присоединенным аутентификатором (a, b). В таких случаях процедура установления подлинности принятого сообщения состоит в проверке соответствия аутентификатора сообщению.

Схема цифровой подписи Эль Гамаля имеет ряд преимуществ по сравнению со схемой цифровой подписи RSA:

1 При заданном уровне стойкости алгоритма цифровой подписи целые числа, участвующие в вычислениях, имеют запись на 25 % короче, что уменьшает сложность вычислений почти в два раза и позволяет заметно сократить объем используемой памяти.

2 При выборе модуля P достаточно проверить, что это число является простым и что у числа $(P - 1)$ имеется большой простой множитель (т. е. всего два достаточно просто проверяемых условия).

3 Процедура формирования подписи по схеме Эль Гамаля не позволяет вычислять цифровые подписи под новыми сообщениями без знания секретного ключа (как в RSA).

Однако алгоритм цифровой подписи Эль Гамаля имеет и некоторые недостатки по сравнению со схемой подписи RSA. В частности, длина цифровой подписи получается в 1,5 раза больше, что, в свою очередь, увеличивает время ее вычисления.

Российский стандарт электронной цифровой подписи. В российском алгоритме цифровой подписи, определяемом стандартом ГОСТ Р 34.10-94,

используются следующие параметры:

p – большое простое число длиной от 509 до 512 либо от 1020 до 1024 бит;

q – простой сомножитель числа (p – 1), имеющий длину от 254 до 256 бит;

a – любое число, меньшее (p – 1), причем такое, что $a^q \bmod p = 1$;

x – некоторое число, меньшее q;

$y = a^x \bmod p$.

Кроме того, этот алгоритм использует однонаправленную хэш-функцию $h(M)$. Стандарт ГОСТ Р 34.11-94 определяет хэш-функцию, основанную на использовании стандартного симметричного алгоритма ГОСТ 28147-89.

Первые три параметра p, q и a являются открытыми и могут быть общими для всех пользователей сети. Число x является секретным ключом. Число y является открытым ключом.

Чтобы подписать некоторое сообщение m, а затем проверить подпись, выполняются следующие шаги:

1 Пользователь A генерирует случайное число k, причем $k < q$.

2 Пользователь A вычисляет значения $r = (a^k \bmod p) \bmod q$, $s = \{xr + k [h(m)]\} \bmod q$.

Если $h(m) \bmod q = 0$, то значение $h(m) \bmod q$ принимают равным единице. Если $r = 0$, то выбирают другое значение k и начинают снова.

Цифровая подпись представляет собой два числа: $r \bmod 2^{256}$ и $s \bmod 2^{256}$.

Пользователь A отправляет эти числа пользователю B.

3 Пользователь B проверяет полученную подпись, вычисляя последовательно величины $v = h(m)^{q-2} \bmod q$, $z_1 = (sv) \bmod q$, $z_2 = ((q - r)v) \bmod q$, $u = (a^{z_1} \cdot y^{z_2} \bmod p) \bmod q$.

Если $u = r$, то подпись считается верной.

Следует также отметить, что в данном стандарте ЭЦП параметр q имеет длину 256 бит. Западных криптографов вполне устраивает q длиной примерно 160 бит. Различие в значениях параметра q является отражением стремления разработчиков российского стандарта к получению более безопасной подписи.

1.5 Управление ключами

Как бы ни была сложна и надежна сама криптосистема, она основана на использовании ключей. Если для обеспечения конфиденциального обмена информацией между двумя пользователями процесс обмена ключами тривиален, то в системе, где количество пользователей составляет десятки и сотни управление ключами, – это серьезная проблема.

Под ключевой информацией понимается совокупность всех действующих в системе ключей. Если не обеспечено достаточно надежное

управление ключевой информацией, то, завладев ею, злоумышленник получает неограниченный доступ ко всей информации.

Управление ключами – информационный процесс, включающий в себя три элемента:

– генерацию ключей;

– накопление ключей;

– распределение ключей.

Рассмотрим, как они должны быть реализованы для того, чтобы обеспечить безопасность ключевой информации.

Генерация ключей. В реальных системах используются специальные аппаратные и программные методы генерации случайных ключей. Как правило используют датчики случайных чисел. Однако степень случайности их генерации должна быть достаточно высокой. Идеальными генераторами являются устройства на основе "натуральных" случайных процессов. Например, генерация ключей на основе белого радиошума. Другим случайным математическим объектом являются десятичные знаки иррациональных чисел, например π или e, которые вычисляются с помощью стандартных математических методов.

В системах со средними требованиями защищенности вполне приемлемы программные генераторы ключей, которые вычисляют случайные числа как сложную функцию от текущего времени и (или) числа, введенного пользователем.

Накопление ключей. Под накоплением ключей понимается организация их хранения, учета и удаления.

Поскольку ключ является самым привлекательным для злоумышленника объектом, открывающим ему путь к конфиденциальной информации, то вопросам накопления ключей следует уделять особое внимание.

Секретные ключи никогда не должны записываться в явном виде на носителе, который может быть считан или скопирован.

В достаточно сложной системе один пользователь может работать с большим объемом ключевой информации, и иногда даже возникает необходимость организации минибаз данных по ключевой информации. Такие базы данных отвечают за принятие, хранение, учет и удаление используемых ключей.

Каждая информация об используемых ключах должна храниться в зашифрованном виде. Ключи, зашифровывающие ключевую информацию называются мастер-ключами. Желательно, чтобы мастер-ключи каждый пользователь знал наизусть и не хранил их вообще на каких-либо материальных носителях.

Очень важным условием безопасности информации является периодическое обновление ключевой информации в системе. При этом переназначаться должны как обычные ключи, так и мастер-ключи. В особо ответственных

системах обновление ключевой информации необходимо производить ежедневно.

Вопрос обновления ключевой информации связан и с третьим элементом управления ключами – распределением ключей.

Распределение ключей. Распределение ключей – самый ответственный процесс в управлении ключами. К нему предъявляются два требования:

– оперативность и точность распределения;

– скрытность распределяемых ключей.

В последнее время заметен сдвиг в сторону использования криптосистем с открытым ключом, в которых проблема распределения ключей отпадает. Тем не менее распределение ключевой информации в системе требует новых эффективных решений.

Распределение ключей между пользователями реализуются двумя разными подходами:

1 *Путем создания одного или нескольких центров распределения ключей.* Недостаток такого подхода состоит в том, что в центре распределения известно, кому и какие ключи назначены, и это позволяет читать все сообщения, циркулирующие в системе. Возможные злоупотребления существенно влияют на защиту.

2 *Прямой обмен ключами* между пользователями системы. В этом случае проблема состоит в том, чтобы надежно удостоверить подлинность субъектов.

В обоих случаях должна быть гарантирована подлинность сеанса связи. Это можно обеспечить двумя способами:

1 *Механизм запроса-ответа*, который состоит в следующем. Если пользователь А желает быть уверенным, что сообщения, которые он получает от пользователя В, не являются ложными, он включает в посылаемое для В сообщение непредсказуемый элемент (запрос). При ответе пользователь В должен выполнить некоторую операцию над этим элементом (например, добавить 1). Это невозможно осуществить заранее, так как не известно, какое случайное число придет в запросе. После получения ответа с результатами действий пользователь А может быть уверен, что сеанс является подлинным. Недостатком этого метода является возможность установления, хотя и сложной, закономерности между запросом и ответом.

2 *Механизм отметки времени.* Он подразумевает фиксацию времени для каждого сообщения. В этом случае каждый пользователь системы может знать, насколько “старым” является пришедшее сообщение.

В обоих случаях следует использовать шифрование, чтобы быть уверенным, что ответ послан не злоумышленником и штемпель отметки времени не изменен.

При использовании отметок времени встает проблема допустимого временно́го интервала задержки для подтверждения подлинности сеанса. Ведь

сообщение с отметкой времени в принципе не может быть передано мгновенно. Кроме этого, компьютерные часы получателя и отправителя не могут быть абсолютно синхронизированы.

Для обмена ключами можно использовать криптосистемы с открытым ключом, используя тот же алгоритм RSA.

Но весьма эффективным оказался алгоритм Диффи-Хелмана, позволяющий двум пользователям без посредников обменяться ключом, который может быть использован затем для симметричного шифрования.

Алгоритм Диффи-Хеллмана. Диффи и Хелман предложили для создания криптографических систем с открытым ключом функцию дикретного возведения в степень.

Необратимость преобразования в этом случае обеспечивается тем, что достаточно легко вычислить показательную функцию в конечном поле Галуа, состоящим из p элементов (p – либо простое число, либо простое в любой степени). Вычисление же логарифмов в таких полях – значительно более трудоемкая операция.

Для обмена информацией первый пользователь выбирает случайное число x_1, равновероятное из целых чисел от 1 до $p - 1$. Это число он держит в секрете, а другому пользователю посылает число $y_1 = \alpha^{x_1} \bmod p$, где α – фиксированный элемент поля Галуа $GF(p)$, который вместе с p заранее распространяется между пользователями.

Аналогично поступает и второй пользователь, генерируя x_2 и вычислив y_2, отправляя его первому пользователю. В результате этого они оба могут вычислить общий секретный ключ $k_{12} = \alpha^{x_1 x_2} \bmod p$.

Для того, чтобы вычислить k_{12}, первый пользователь возводит y_2 в степень x_1 и находит остаток от деления на p. То же делает и второй пользователь, только используя y_1 и x_2. Таким образом, у обоих пользователей оказывается общий ключ k_{12}, который можно использовать для шифрования информации обычными алгоритмами. В отличие от алгоритма RSA, данный алгоритм не позволяет шифровать собственно информацию.

Не зная x_1 и x_2, злоумышленник может попытаться вычислить k_{12}, зная только перехваченные y_1 и y_2. Эквивалентность этой проблемы проблеме вычисления дискретного логарифма есть главный и открытый вопрос в системах с открытым ключом. Простого решения до настоящего времени не найдено. Так, если для прямого преобразования 1000-битных простых чисел требуется 2000 операций, то для обратного преобразования (вычисления логарифма в поле Галуа) – потребуется около 10^{30} операций.

Для примера выберем $p = 43$. α можно определить как первый простой множителель числа $(p - 1)$. Первым простым множителем числа $(43 - 1) = 42$ является 3. Между двумя пользователями распределяется пара чисел (43, 3). Пользователь A придумывает свой секретный ключ, допустим, число

8 и посылает пользователю B число $y_1 = 3^8 \pmod{43} = 25$. Пользователь B придумывает свой секретный ключ, допустим, число 37 и посылает пользователю A число $y_2 = 3^{37} \pmod{43} = 20$. Оба пользователя вычисляют общий секретный ключ k_{12}: пользователь A – $k_{12} = 20^8 \pmod{43} = 9$; пользователь B – $k_{12} = 25^{37} \pmod{43} = 9$.

Как видно, при всей простоте алгоритма Диффи-Хелмана, вторым его недостатком по сравнению с системой RSA является отсутствие гарантированной нижней оценки трудоемкости раскрытия ключа.

Кроме того, хотя описанный алгоритм позволяет обойти проблему скрытой передачи ключа, необходимость аутентификации остается. Без дополнительных средств, один из пользователей не может быть уверен, что он обменялся ключами именно с тем пользователем, который ему нужен.

2 ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1 Изучить краткие сведения из теории.

2 Для решения задач из пп. 4–13 по последней цифре шифра из таблицы 11 необходимо выбрать сообщения, которые будут подвергаться шифрованию.

Таблица 11 – **Исходные сообщения**

Параметр	Последняя цифра шифра				
	1	2	3	4	5
M_1	За объектами альфа ведется постоянное наблюдение ничего не предпринимать				
M_2	lagra manu	licitum sit	ite missia est	in vino veritas	dura lex sed lex
M_3	$A0B1C13F3FA6D4B6_{(16)}$				
M_4	3 0 7 9 11 2	7 12 1 0 3 2	9 1 3 17 2 6	13 3 2 7 1 4	6 2 12 9 1 3
m	5	7	9	11	13
Параметр	Последняя цифра шифра				
	6	7	8	9	0
M_1	Сектор очищен искомый объект не обнаружен срочно переходите к шестой схеме				
M_2	jus humanum	memento mori	vini vidi vici	cogito ergo sum	citius altius
M_3	$49F3BAD29A1B0F67_{(16)}$				
M_4	7 3 8 9 10 2	19 9 8 1 0 6	9 3 0 8 1 12	7 8 9 15 1 5	6 1 0 16 5 9
m	15	17	19	21	23

3 Для решения задач из пп. 5, 6, 9–11, 13 и15 по предпоследней цифре шифра из таблицы 12 необходимо выбрать ключи шифрования.

4 Для решения задач из пп. 7, 8, 12–16 по первой цифре шифра из таблицы 13 необходимо выбрать ключи шифрования.

5 Зашифровать сообщение M_1 на русском языке с помощью шифрующей таблицы размером 8 на 8:

– без использования перестановок;

– с одиночной перестановкой по ключу K_1 (рисунок 21);

– с двойной перестановкой по ключам K_2 и K_3 (рисунок 22).

6 Зашифровать сообщение M_2 на латинском языке с помощью:

– системы шифрования Цезаря при смещении на k символов;

– аффинной системы подстановок Цезаря с параметрами a, b;

– таблицы Вижинера для английского алфавита и ключа K_4.

7 Зашифровать сообщение M_1 на русском языке с помощью таблицы Вижинера для русского алфавита и ключа K_1.

Таблица 12 – **Ключи шифрования**

Параметр	Предпоследняя цифра шифра									
	1	2	3	4	5	6	7	8	9	0
K_1	музыкант		экспонат		организм		механизм		грузовик	
K_2	разность		покрытие		компания		выдержка		верность	
K_3	бумеранг		интервал		родитель		нотариус		кастрюля	
K_4	sine mora		pro forma		rideamus		sapienti		divideet	
K_5	ABCDCF27B967D5F1$_{(16)}$					F51FD8A4FA3890BA$_{(16)}$				
X_1	9	7	5	12	8	9	7	8	13	7
X_2	2	11	9	6	13	11	12	11	6	9

8 Зашифровать бинарное сообщение M_3 длиной 64 бит с использованием симметричной схемы шифрования DES и ключа K_5.

9 Зашифровать сообщение M_4 с использованием асимметричной схемы шифрования:

– RSA с параметрами P_1, Q_1.

– Эль Гамаля с параметрами P_2, G_1, X_1, K_6.

Таблица 13 – **Параметры алгоритмов шифрования**

Параметр	Первая цифра шифра									
	1	2	3	4	5	6	7	8	9	0
k	3	4	5	6	7	8	9	10	11	12
a	1	2	3	1	2	3	1	2	3	4
b	9	8	7	6	5	4	3	2	1	0
P_1	4	5	7	9	11	13	15	16	12	6
Q_1	13	16	12	10	8	7	8	3	5	11
P_2	14	15	16	17	18	19	20	21	22	23
G_1	2	4	6	8	10	12	11	9	7	5
K_6	12	9	7	15	10	11	9	11	8	13
P_3	6	12	16	15	13	11	9	7	5	4
Q_2	13	7	5	7	8	9	7	11	13	11
P_4	23	22	21	20	19	18	17	16	15	14
G_2	5	7	9	11	12	10	8	6	4	2
K_7	9	11	13	7	7	5	11	2	3	4
p	45	39	41	37	49	57	61	33	48	53

x_1	13	31	15	5	9	31	42	5	30	14
x_2	27	9	21	22	13	12	10	23	16	22

10 Создать ЭЦП для хешированного сообщения m с использованием алгоритма:

– RSA с параметрами P_3, Q_2.

– Эль Гамаля с параметрами P_4, G_2, X_2, K_7.

Символы ключа K_1							

Рисунок 21 – Пустая матрица для одиночной перестановки

Символы ключа K_3	Символы ключа K_2							

Рисунок 22 – Пустая матрица для двойной перестановки

11 Создать общий ключ для двух пользователей с использованием алгоритма Диффи-Хелмана с параметрами p, x_1, x_2.