Министерство образования Республики Беларусь

Учреждение образования
«Полоцкий государственный университет»

Р. П. Богуш
А. В. Курилович

ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИИ

Учебно-методический комплекс
для слушателей ИПК специальности 1-40 01 73
«Программное обеспечение информационных систем»

Новополоцк
ПГУ
2009

УДК 621.391.1(075.8)
ББК 32.811.4я73
Б73

Рекомендовано к изданию методической комиссией
факультета информационных технологий
в качестве учебно-методического комплекса
(протокол № 3 от 25.03.2009)

РЕЦЕНЗЕНТЫ:

канд. техн. наук, доц., докторант каф. защиты информации УО «БГУИР»
Т. В. БОРБОТЬКО;
канд. техн. наук, доц., проректор по информатизации УО «ПГУ»
Д. О. ГЛУХОВ

Богуш, Р. П.

Б73 Основы защиты информации : учеб.-метод. комплекс для слушателей ИПК спец. 1-40 01 73 «Программное обеспечение информационных систем» / Р. П. Богуш, А. В. Курилович. – Новополоцк : ПГУ, 2009. – 96 с.
ISBN 978-985-418-819-5.

Содержит курс лекций, рассмотрены задачи информационной безопасности, угрозы, элементы теории сложности и теории чисел в приложении к компьютерной защите информации, методы и алгоритмы криптографического кодирования информации. Представлены методические указания к выполнению семинарских занятий. Предлагается система оценки знаний слушателей ИПК.

УДК 621.391.1(075.8)
ББК 32.811.4я73

ISBN 978-985-418-819-5

ВВЕДЕНИЕ В КУРС «ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИИ»

Цель и задачи дисциплины

Целью дисциплины «Основы защиты информации» является формирование у слушателей курсов переподготовки знаний и умений о задачах защиты информации, основных информационных угрозах и методах криптографического кодирования информации.

В результате изучения дисциплины слушатели должны

знать:

- задачи информационной безопасности;
- современные угрозы информационной безопасности;
- криптографические алгоритмы и системы;

уметь:

- использовать криптографические алгоритмы для защиты информации от несанкционированного доступа;

иметь представление:

- о сложности проблем и алгоритмов;
- о направлениях, перспективах и проблемах развития информационной безопасности.

Структура дисциплины

Согласно учебному плану специальности 1-40 01 73 «Программное обеспечение информационных систем» курс «Основы защиты информации» изучается слушателями на 1 курсе (1 семестр), рассчитан на 16 часов лекций и 4 часа семинарских занятий. Ниже представлено распределение курса по видам аудиторных занятий по разделам и темам.

Лекционный курс

Наименование разделов и тем лекций	Кол-во часов
Раздел 1. Основные понятия информационной безопасности	2
Раздел 2. Элементы теории сложности. Элементы теории чисел	2
Раздел 3. Методы криптографического кодирования информации	
3.1. Криптографическая защита компьютерной информации	2
3.2. Шифры перестановки. Шифры простой замены	2
Раздел 4. Современные симметричные криптосистемы	
4.1. Американский стандарт шифрования DES	2
4.2. Стандарт шифрования ГОСТ 28147-89	2
Раздел 5. Современные асимметричные криптосистемы	
5.1. Построения систем с открытым ключом. Алгоритм RSA	2
5.2. Криптосистема Эль-Гамаля. Алгоритм Рабина. Комбинированный метод шифрования	2

Семинарские занятия

Наименование семинарских занятий	Кол-во часов
1. Симметричные системы шифрования DES и ГОСТ 28147-89	2
2. Ассиметричные системы шифрования	2

Оценка знаний слушателей курсов переподготовки

Для оценки работы и знаний слушателей курсов переподготовки в рамках курса «Основы защиты информации» используется накопительная система. Результирующая оценка выставляется по сумме баллов, которые слушатель набирает в течение всего учебного семестра, а также в результате выходного итогового контроля – зачета.

Распределение баллов по видам занятий

Вид занятий	Форма оценки учебной активности слушателя курсов переподготовки	Максимальное количество баллов по каждой форме оценки	Максимальное количество баллов по каждому виду занятий
Семинарские занятия	Устные ответы на вопросы	50	220
	Решение задач и (или) выполнение тестовых заданий	60	
Зачет	Ответы на вопросы	400	400

Дополнительные баллы предусматриваются за подготовку докладов (до 150 баллов).

Итоговый зачет по данной дисциплине выставляется в случае, если слушатель курсов переподготовки набрал 310 баллов.

Для этого необходимо, например:

- по результатам семинарских занятий набрать 100 баллов;
- подготовить доклад – 110 баллов;
- получить при сдаче зачета 100 баллов.

Модуль 1

ОСНОВНЫЕ ПОНЯТИЯ ИНФОРМАЦИОННОЙ БЕЗОПАСНОСТИ

Цель модуля – изучение слушателями курсов переподготовки задач защиты информации и основных информационных угроз.

В результате изучения модуля слушатели должны

знать:

- задачи информационной безопасности;
- современные угрозы информационной безопасности;

иметь представление

- о государственной политике в информационной сфере.

1.1. Введение в информационную безопасность

Проблема защиты информации путем ее преобразования, исключающего доступ к ней посторонних лиц, является актуальной с давних времен. Бурное развитие криптографические системы получили в годы первой и второй мировых войн. Начиная с послевоенного времени и по нынешний день появление вычислительных средств ускорило разработку и совершенствование криптографических методов.

Защита информации – это комплекс мероприятий, направленных на обеспечение информационной безопасности.

Словосочетание «информационная безопасность» в разных контекстах может иметь различный смысл:

1. В Доктрине информационной безопасности Республики Беларусь термин «информационная безопасность» используется в широком смысле, как состояние защищенности информационной среды общества, обеспечивающее ее формирование, использование и развитие в интересах граждан, организаций, государства.
2. Термин «информационная безопасность» может использоваться в узком смысле. В этом случае под информационной безопасностью понимают защищенность информации и поддерживающей инфраструктуры от случайных или преднамеренных воздействий естественного или искусственного характера, которые могут нанести неприемлемый ущерб субъектам информационных отношений, в т.ч. владельцам и пользователям информации и поддерживающей инфраструктуры.

Таким образом, правильный с методологической точки зрения подход к проблемам информационной безопасности начинается с выявления субъектов информационных отношений и интересов этих субъектов, связанных с использованием информационных систем (ИС). Угрозы инфор-

мационной безопасности – это оборотная сторона использования информационных технологий.

Из этого положения очевидно:

1. Трактовка проблем, связанных с информационной безопасностью, для разных категорий субъектов может существенно различаться. Для пояснения достаточно сопоставить режимные государственные организации и учебные институты. В первом случае «пусть лучше все сломается, чем враг узнает хоть один секретный бит», во втором – «да нет у нас никаких секретов, лишь бы все работало».

2. Информационная безопасность не сводится исключительно к защите от несанкционированного доступа к информации, это принципиально более широкое понятие. Субъект информационных отношений может пострадать (понести убытки и/или получить моральный ущерб) не только от несанкционированного доступа, но и от поломки системы, вызвавшей перерыв в работе. Более того, для многих открытых организаций (например, учебных) собственно защита от несанкционированного доступа к информации стоит по важности отнюдь не на первом месте.

Согласно определению информационной безопасности, она зависит не только от компьютеров, но и от поддерживающей инфраструктуры, к которой можно отнести системы электро-, водо- и теплоснабжения, кондиционеры, средства коммуникаций и, конечно, обслуживающий персонал. Эта инфраструктура имеет самостоятельную ценность, но нас интересует лишь то, как она влияет на выполнение информационной системой предписанных ей функций.

Застраховаться от всех видов ущерба невозможно, тем более невозможно сделать это экономически целесообразным способом, когда стоимость защитных средств и мероприятий не превышает размер ожидаемого ущерба. Значит, с чем-то приходится мириться и защищаться следует только от того, с чем смириться никак нельзя. Иногда таким недопустимым ущербом является нанесение вреда здоровью людей или состоянию окружающей среды, но чаще порог неприемлемости имеет материальное (денежное) выражение, а целью защиты информации становится уменьшение размеров ущерба до допустимых значений.

1.2. Угрозы информационной безопасности

Основные угрозы информационной безопасности можно сформулировать следующим образом [4].

Угроза (опасность) – это потенциальная возможность определенным образом нарушить информационную безопасность. Попытка реализации угрозы

называется атакой, а тот, кто предпринимает такую попытку, – злоумышленником. Потенциальные злоумышленники называются источниками угрозы.

Чаще всего угроза является следствием наличия уязвимых мест в защите информационных систем (таких, например, как возможность доступа посторонних лиц к критически важному оборудованию или ошибки в программном обеспечении). Промежуток времени от момента, когда появляется возможность использовать слабое место, и до момента, когда пробел ликвидируется, называется окном опасности, ассоциированным с данным уязвимым местом. Пока существует окно опасности, возможны успешные атаки на ИС. Если речь идет об ошибках в программном обеспечении, то окно опасности «открывается» с появлением средств использования ошибки и ликвидируется при наложении заплат, ее исправляющих. Новые уязвимые места и средства их использования появляются постоянно; это значит, во-первых, что почти всегда существуют окна опасности и, во-вторых, что отслеживание таких окон должно производиться постоянно, а выпуск и наложение заплат – как можно более оперативно.

Некоторые угрозы нельзя считать следствием каких-то ошибок или просчетов; они существуют в силу самой природы современных ИС. Например, угроза отключения электричества или выхода его параметров за допустимые границы существует в силу зависимости аппаратного обеспечения ИС от качественного электропитания.

Иметь представление о возможных угрозах, а также об уязвимых местах, которые эти угрозы обычно эксплуатируют, необходимо для того, чтобы выбирать наиболее экономичные средства обеспечения безопасности. Классификация, называемая STRIDE – по первым буквам английских названий категорий, представлена в [4].

Подмена сетевых объектов (*Spoofing identity*). Атаки подобного типа позволяют взломщику выдавать себя за другого пользователя или подменять настоящий сервер подложным. Пример подмены личности пользователя – использование ворованных аутентификационных данных (имени пользователя и пароля) для атаки на систему. Типичный пример подобной бреши «из жизни» – применение ненадежных методов аутентификации, например, некоторых видов HTTP-аутентификации: базовой (Basic authentication) и аутентификации на основе хеша (Digest authentication). Подменой серверов считаются подлог DNS-сервера (DNS spoofing) и модификация записей кэша DNS (DNS cache poisoning).

Модификация данных (*Tampering with data*). Атаки этого типа предусматривают злонамеренную порчу данных. Примеры: несанкционированные изменения постоянных данных (например, хранящихся в базе данных),

а также информации, пересылаемой между компьютерами через открытую сеть (например, Интернет).

Отказ от авторства (*Repudiation*). Контрагент отказывается от совершенного им действия (или бездействия), пользуясь тем, что у другой стороны нет никакого способа доказать обратное. Например, в системе, где не ведется аудит, пользователь может выполнить запрещенную операцию и отказаться от ее «авторства», а администратору не удастся ничего доказать. Невозможность отрицания авторства (nonrepudiation) – это способность системы противостоять такой опасности. Купив товар через Интернет, пользователь должен расписаться в его получении. Продавец впоследствии сможет использовать подписанную квитанцию как доказательство того, что пользователь действительно получил товар. Понятно, что поддержка невозможности отрицания авторства жизненно важна для приложений электронной коммерции.

Разглашение информации (*Information disclosure*). Подразумевается раскрытие информации лицам, доступ к которой им запрещен, например, прочтение пользователем файла, доступ к которому ему не предоставлялся, а также способность злоумышленника считывать данные при передаче между компьютерами. Примером может быть перехват пароля пользователя при передаче по сети.

Отказ в обслуживании (*Denial of service*). В атаках такого типа взломщик пытается лишить доступа к сервису правомочных пользователей, например, сделав Web-сервер временно недоступным или непригодным для работы.

Повышение привилегий (*Elevation of privilege*). В данном случае непривилегированный пользователь получает привилегированный доступ, позволяющий ему «взломать» или даже уничтожить систему. К повышению привилегий относятся и случаи, когда злоумышленник удачно проникает через защитные средства системы и становится частью защищенной и доверенной подсистемы. Пример: в уязвимой системе злоумышленник может поместить на диск файл, который выполняется автоматически при входе пользователя в систему, и дождаться входа в систему полномочного пользователя. Если это администратор, зловредный код будет выполняться от его имени.

Некоторые типы опасностей взаимосвязаны. Если реквизиты пользователей недостаточно надежно защищены, то нередко разглашение информации влечет за собой подмену объектов. И, конечно же, повышение привилегий приводит к значительно худшим последствиям: если кто-то получит полномочия администратора или учетной записи root на атакуемом компьютере, все потенциальные угрозы остальных категорий становятся реальностью. И наоборот, иногда подмены объектов достаточно для достижения цели, и

злоумышленнику не нужно даже повышать привилегии. Так, получив контроль над SMTP-сервером, злоумышленник сможет замечательно повеселиться, разослав электронное письмо от имени генерального директора с объявлением лишнего выходного дня за отлично выполненную работу.

1.3. Основные задачи информационной безопасности

Основные задачи информационной безопасности следующие [4].

Секретность (*privacy, confidentiality, secrecy*). Практически у каждого человека или организации найдутся документы, которые ни в коем случае не должны стать всеобщим достоянием, будь то личные медицинские данные, информация о финансовых операциях или государственная тайна. Пока для хранения используются неэлектронные средства, секретность обеспечивается административными методами (хранение в сейфах, транспортировка в сопровождении охраны и т.д.). Но когда информация обрабатывается на компьютерах и передается по открытым каналам связи, административные методы оказываются бессильны и на помощь приходят методы информационной безопасности. Задача обеспечения секретности фактически сводится к тому, чтобы сделать возможным хранение и передачу данных в таком виде, чтобы противник, даже получив доступ к носителю или среде передачи, не смог получить сами защищенные данные.

Целостность (*data integrity*). В процессе обработки и передачи по каналам связи данные могут быть искажены как случайно, так и преднамеренно. Также информация может быть изменена прямо на носителе, где она хранится. Проверка целостности просто необходима в ситуациях, когда интерпретация неправильных данных может привести к очень серьезным последствиям, например, при возникновении ошибки в сумме банковского перевода или значении скорости самолета, заходящего на посадку. Обеспечение целостности (контроль целостности) заключается в том, чтобы позволить либо утверждать, что данные не были модифицированы при хранении и передаче, либо определить факт искажения данных, т.е. никакое изменение данных не должно пройти незамеченным.

Идентификация (*identification*) необходима для того, чтобы отождествить пользователя с некоторым уникальным идентификатором. После этого ответственность за все действия, при выполнении которых предъявлялся данный идентификатор, возлагается на пользователя, за которым этот идентификатор закреплен.

Аутентификация (*data origin, authentication*) является необходимым дополнением к идентификации и предназначена для подтверждения истинности (аутентичности) пользователя, предъявившего идентификатор.

Неанонимный пользователь должен получить возможность работать только после успешной аутентификации.

Уполномочивание (*authorization*) сводится к тому, что ни один пользователь не должен получить доступ к системе без успешного выполнения идентификации и последующей аутентификации и ни один пользователь не должен получить доступ к ресурсам, если он не уполномочен на такие действия специальным разрешением.

Контроль доступа (*access control*) – комплексное понятие, обозначающее совокупность методов и средств, предназначенных для ограничения доступа к ресурсам. Доступ должны иметь только уполномоченные пользователи, и попытки доступа должны протоколироваться.

Право собственности (*ownership*) – используется для того, чтобы предоставить пользователю законное право на использование некоторого ресурса и, при желании, возможность передачи этого ресурса в собственность другому пользователю. Право собственности обычно является составной частью системы контроля доступа.

Сертификация (*certification*) – процесс подтверждения некоторого факта стороной, которой пользователь доверяет. Чаще всего сертификация используется для удостоверения принадлежности открытого ключа конкретному пользователю или компании, т.к. эффективное использование инфраструктуры открытых ключей возможно лишь при наличии системы сертификации. Организации, занимающиеся выдачей сертификатов, называются удостоверяющими центрами.

Подпись (*signature*) позволяет получателю документа доказать, что данный документ был подписан именно отправителем. При этом подпись не может быть перенесена на другой документ, отправитель не может отказаться от своей подписи, любое изменение документа приведет к нарушению подписи, и любой пользователь, при желании, может самостоятельно убедиться в подлинности подписи.

Неотказуемость (*non-repudiation*) – свойство схемы информационного обмена, при котором существует доказательство, которое получатель сообщения способен предъявить третьей стороне, чтобы та смогла независимо проверить, кто является отправителем сообщения. То есть отправитель сообщения не имеет возможности отказаться от авторства, т.к. существуют математические доказательства того, что никто кроме него не способен создать такое сообщение.

Расписка в получении (*receipt*) передается от получателя к отправителю и может впоследствии быть использована отправителем для доказательства того, что переданная информация была доставлена получателю не позже определенного момента, указанного в расписке.

Датирование (*time stamping*) часто применяется совместно с подписью и позволяет зафиксировать момент подписания документа. Это может быть полезно, например, для доказательства первенства, если один документ был подписан несколькими пользователями, каждый из которых утверждает, что именно он является автором документа. Кроме того, датирование широко используется в сертификатах, которые имеют ограниченный срок действия. Если действительный сертификат был использован для подписи, а затем соответствующей службой сертифицирующего центра была проставлена метка времени, то такая подпись должна признаваться правильной и после выхода сертификата из употребления. Если же отметка времени отсутствует, то после истечения срока действия сертификата подпись не может быть признана корректной.

Аннулирование (*annul*) используется для отмены действия сертификатов, полномочий или подписей. Если какой-либо участник информационного обмена или принадлежащие ему ключи и сертификаты оказались скомпрометированы, необходимо предотвратить доступ этого пользователя к ресурсам и отказать в доверии соответствующим сертификатам, т.к. ими могли воспользоваться злоумышленники. Также процедура аннулирования может быть использована в отношении удостоверяющего центра.

Свидетельствование (witnessing) – удостоверение (подтверждение) факта создания или существования информации некоторой стороной, не являющейся создателем.

Анонимность (*anonymity*) – довольно редко вспоминаемая задача. Правительствам и корпорациям невыгодно, чтобы пользователь мог остаться анонимным при совершении каких-либо действий в информационном пространстве. Возможно, по этой причине проекты по обеспечению анонимности носят единичный характер и, как правило, долго не живут. Да и средства коммуникации в подавляющем большинстве позволяют определить маршрут передачи того или иного сообщения, а значит, вычислить отправителя.

Все рассмотренные задачи информационной безопасности сформулированы исходя из потребностей существующего информационного мира. Может быть, в будущем часть задач потеряет свою актуальность или появятся новые задачи, нуждающиеся в решении.

Вопросы и задания для самопроверки

1.1. Что понимают под защитой информации?
1.2. Поясните термин «информационная безопасность».
1.3. Что понимают под угрозой информационной безопасности?
1.4. Что определяет термин «окно опасности»?
1.5. Приведите классификацию угроз согласно STRIDE.
1.6. Назовите основные задачи информационной безопасности.

Модуль 2

ЭЛЕМЕНТЫ ТЕОРИИ СЛОЖНОСТИ. ЭЛЕМЕНТЫ ТЕОРИИ ЧИСЕЛ

Цель модуля – изучение слушателями курсов переподготовки аспектов теории сложности и теории чисел в приложении к задаче криптографической защиты информации.

В результате изучения модуля слушатели переподготовки должны

знать:

- классы сложности алгоритмов и проблем;
- основы модульной арифметики;

уметь:

- оценивать вычислительную сложность алгоритмов;

иметь представление:

- о классической и вероятностной машинах Тьюринга;
- о теоретико-сложностном подходе к определению криптостойкости систем.

2.1. Введение в теорию сложности

Теория сложности обеспечивает методологию анализа вычислительной сложности различных криптографических методов и алгоритмов. Она сравнивает криптографические методы и алгоритмы и определяет их безопасность. Теория информации сообщает о том, что все криптографические алгоритмы (кроме одноразовых блокнотов) могут быть взломаны. Теория сложности сообщает, за какое время алгоритмы могут быть взломаны [6].

С точки зрения классической математики проблемы в криптографии являются тривиальными в том смысле, что могут быть решены за конечное число попыток. Однако сведение к конечному числу случаев не имеет особого смысла, если число самих случаев практически не реализуемо, т.е. если система не способна расшифровать некоторое сообщение в разумных временных рамках.

Предположим теперь, что противник атакует криптосистему. Ему известен открытый ключ K_1, но неизвестен соответствующий секретный ключ K_2. Противник перехватил криптограмму d и пытается найти сообщение m, где $d = E_{K_1}(m)$. Поскольку алгоритм шифрования общеизвестен, противник может просто последовательно перебрать все возможные сообщения длины n, вычислить для каждого такого сообщения m_i криптограм-

му $d_i = E_{K_1}(m_i)$ и сравнить d_i с d. То сообщение, для которого $d_i = d$, будет искомым открытым текстом. Если повезет, то открытый текст будет найден достаточно быстро. В худшем случае перебор будет выполнен за время порядка $2^n T(n)$, где $T(n)$ – время, требуемое для вычисления функции E_{K_1} от сообщений длины n. Если сообщения имеют длину порядка 1 000 битов, то такой перебор неосуществим на практике ни на каких самых мощных компьютерах.

Рассмотрен лишь один из возможных способов атаки на криптосистему и простейший алгоритм поиска открытого текста, называемый обычно алгоритмом полного перебора. Используется также и другое название – «метод грубой силы». Другой простейший алгоритм поиска открытого текста – угадывание. Этот очевидный алгоритм требует небольших вычислений, но срабатывает с пренебрежимо малой вероятностью (при больших длинах текстов). На самом деле противник может пытаться атаковать криптосистему различными способами и использовать различные, более изощренные алгоритмы поиска открытого текста. Естественно считать криптосистему стойкой, если любой такой алгоритм требует практически неосуществимого объёма вычислений или срабатывает с пренебрежимо малой вероятностью. (При этом противник может использовать не только детерминированные, но и вероятностные алгоритмы.) Это и есть теоретико-сложностной подход к определению стойкости. Для его реализации в отношении того или иного типа криптографических схем необходимо выполнить следующее [6]:

- дать формальное определение схемы данного типа;
- дать формальное определение стойкости схемы;
- доказать стойкость конкретной конструкции схемы данного типа.

Здесь сразу же возникает ряд проблем.

Во-первых, в криптографических схемах используются фиксированные значения параметров. Например, криптосистемы разрабатываются для ключей длины в 256 или 512 байтов. Для применения же теоретико-сложностного подхода необходимо, чтобы задача, вычислительную сложность которой предполагается использовать, была массовой. Поэтому в теоретической криптографии рассматриваются математические модели криптографических схем. Эти модели зависят от некоторого параметра, называемого параметром безопасности, который может принимать сколь угодно большие значения (обычно для простоты предполагается, что параметр безопасности может пробегать весь натуральный ряд).

Во-вторых, определение стойкости криптографической схемы зависит от той задачи, которая стоит перед противником, и от того, какая информа-

ция о схеме ему доступна. Поэтому стойкость схем приходится определять и исследовать отдельно для каждого предположения о противнике.

В-третьих, необходимо уточнить, какой объём вычислений можно считать «практически неосуществимым». Из сказанного выше следует, что эта величина не может быть просто константой, она должна быть представлена функцией от растущего параметра безопасности. В соответствии с тезисом Эдмондса алгоритм считается эффективным, если время его выполнения ограничено некоторым полиномом от длины входного слова (в нашем случае – параметра безопасности). В противном случае говорят, что вычисления по данному алгоритму практически неосуществимы. Сами криптографические схемы должны быть эффективными, т.е. все вычисления, предписанные той или иной схемой, должны выполняться за полиномиальное время.

В-четвёртых, необходимо определить, какую вероятность можно считать пренебрежимо малой.

Итак, при наличии всех указанных определений проблема обоснования стойкости криптографической схемы свелась к доказательству отсутствия полиномиального алгоритма, который решает задачу, стоящую перед противником. Но здесь возникает еще одно весьма серьезное препятствие: современное состояние теории сложности вычислений не позволяет доказывать сверхполиномиальные нижние оценки сложности для конкретных задач рассматриваемого класса. Из этого следует, что на данный момент стойкость криптографических схем может быть установлена лишь с привлечением каких-либо недоказанных предположений. Поэтому основное направление исследований состоит в поиске наиболее слабых достаточных условий (в идеале – необходимых и достаточных) для существования стойких схем каждого из типов.

В основном рассматриваются предположения двух типов – общие (или теоретико-сложностные) и теоретико-числовые, т.е. предположения о сложности конкретных теоретико-числовых задач. Все эти предположения в литературе обычно называются криптографическими.

2.2. Сложность алгоритмов

Сложность алгоритма определяется вычислительными мощностями, необходимыми для его выполнения. Вычислительная сложность алгоритма часто измеряется двумя параметрами [6]: T (временная сложность) и S (пространственная сложность, или требования к памяти). И T, и S обычно представляются в виде функций от n, где n – это размер входных данных.

(Существуют и другие способы измерения сложности: количество случайных бит, ширина канала связи, объем данных и т.п.)

Обычно вычислительная сложность алгоритма выражается с помощью нотации *O*, т.е. описывается порядком величины вычислительной сложности. Это просто член разложения функции сложности, быстрее всего растущий с ростом *n*, все члены низшего порядка игнорируются. Например, если временная сложность данного алгоритма равна $4n^2 + 7n + 12$, то вычислительная сложность порядка – n^2, записываемая как $O(n^2)$. Временная сложность, измеренная таким образом, не зависит от реализации. Не нужно знать ни точное время выполнения различных инструкций, ни число битов, используемых для представления различных переменных, ни даже скорость процессора. Один компьютер может быть на 50 % быстрее другого, а у третьего – шина данных может быть в два раза шире, но сложность алгоритма, оцененная по порядку величины, не изменится. При работе с алгоритмами сложными, всем прочим можно пренебречь (с точностью до постоянного множителя) в сравнении со сложностью по порядку величины.

Нотация позволяет увидеть, как объем входных данных влияет на требования ко времени и объему памяти. Например, если $T = O(n)$, то удвоение входных данных удвоит время выполнения алгоритма. Если $T = O(2^n)$, то добавление одного бита к входным данным удвоит время выполнения алгоритма.

Обычно алгоритмы классифицируются в соответствии с их временной или пространственной сложностью. Алгоритм называют постоянным, если его сложность не зависит от *n*: $O(1)$. Алгоритм является линейным, если его временная сложность $O(n)$. Алгоритмы могут быть квадратичными, кубическими и т.д. Все эти алгоритмы – полиномиальны, их сложность – $O(n^m)$, где *m* – константа. Алгоритмы с полиномиальной временной сложностью называются алгоритмами с полиномиальным временем Алгоритмы, сложность которых равна $O\left(t^{f(n)}\right)$, где *t* – константа, большая, чем 1, а $f(n)$ – некоторая полиномиальная функция от *n*, называются экспоненциальными.

Подмножество экспоненциальных алгоритмов, сложность которых равна $O\left(c^{f(n)}\right)$, где *c* – константа, а $f(n)$ возрастает быстрее, чем постоянная, но медленнее, чем линейная функция, называется суперполиномиальным.

В идеале криптограф хотел бы утверждать, что алгоритм, лучший для взлома спроектированного алгоритма шифрования, обладает экспо-

ненциальной временной сложностью. На практике же самые сильные утверждения, которые могут быть сделаны при текущем состоянии теории вычислительной сложности, имеют форму «все известные алгоритмы вскрытия данной криптосистемы обладают суперполиномиальной временной сложностью». То есть, известные нам алгоритмы вскрытия обладают суперполиномиальной временной сложностью, но пока невозможно доказать, что не может быть открыт алгоритм вскрытия с полиномиальной временной сложностью.

С ростом n временная сложность алгоритмов может стать настолько огромной, что это повлияет на практическую реализуемость алгоритма. В табл. 2.1 [6] показано время выполнения для различных классов алгоритмов при n, равном 1 млн.

Таблица 2.1

Время выполнения для различных классов алгоритмов

Класс	Сложность	Количество операций $n = 10^6$	Время при 10^6 операций в секунду
Постоянные	$O(1)$	1	1 мкс
Линейные	$O(n)$	10^6	1 с
Квадратичные	$O(n^2)$	10^{12}	11,6 дня
Кубические	$O(n^2)$	10^{18}	32 000 лет
Экспоненциальные	$O(2^n)$	10^{301030}	

При условии, что единицей времени является микросекунда, компьютер может выполнить постоянный алгоритм за микросекунду, линейный – за секунду, а квадратичный – за 11,6 дня. Выполнение кубического алгоритма потребует 32 тыс. лет, что в принципе реализуемо, компьютер, в конце концов, получил бы решение. Выполнение экспоненциального алгоритма тщетно независимо от экстраполяции роста мощи компьютеров или параллельной обработки.

Временная сложность такого вскрытия грубой силой пропорциональна количеству возможных ключей, которое экспоненциально зависит от длины ключа. Если n – длина ключа, то сложность вскрытия грубой силой равна $O(2^n)$.

2.3. Сложность проблем

Теория сложности также классифицирует и сложность самих проблем, а не только сложность конкретных алгоритмов решения проблемы [6]. Теория рассматривает минимальное время и объем памяти, необходимые для решения самого трудного варианта проблемы на теоретическом компьютере, известном как машина Тьюринга. В обычных машинах Тьюринга (их называют детерминированными, чтобы отличить от вероятностных) новое состояние, в которое машина переходит на очередном шаге, полностью определяется текущим состоянием и тем символом, который обозревает головка на ленте. В вероятностных машинах новое состояние может зависеть еще и от случайной величины, которая принимает значения 0 и 1 с вероятностью 1/2 каждое. Альтернативно можно считать, что вероятностная машина Тьюринга имеет дополнительную случайную ленту, на которой записана бесконечная двоичная случайная строка. Случайная лента может читаться только в одном направлении и переход в новое состояние может зависеть от символа, обозреваемого на этой ленте.

Проблемы, которые можно решить с помощью алгоритмов с полиномиальным временем, называются решаемыми, потому что для разумных входных данных обычно могут быть решены за разумное время. (Точное определение «разумности» зависит от конкретных обстоятельств.) Проблемы, которые невозможно решить за полиномиальное время, называются нерешаемыми, потому что вычисление их решений быстро становится невозможным. Нерешаемые проблемы иногда называют трудными. Проблемы, которые могут быть решены только с помощью суперполиномиальных алгоритмов, вычислительно нерешаемы даже при относительно малых значениях n. Что еще хуже, Алан Тьюринг доказал, что некоторые проблемы принципиально неразрешимы. Даже отвлекаясь от временной сложности алгоритма, невозможно создать алгоритм решения этих проблем.

Проблемы можно разбить на классы в соответствии со сложностью их решения. Самые важные классы и их предполагаемые соотношения показаны на рис. 2.1 [6]. (Следует отметить, что лишь малая часть этих утверждений может быть доказана математически.)

Находящийся в самом низу класс P состоит из всех проблем, которые можно решить за полиномиальное время. Проблема называется (вычислительно) труднорешаемой, если она не принадлежит классу P. Легкорешаемые проблемы (т.е. проблемы из P) образуют в P несколько подклассов с очевидными определениями: задачи с линейной, квадратичной, кубичной и другой временной сложностью. Неформально понятие легкой проблемы означает, что степени многочлена малы, т.е. в указанных выше пределах.

Класс NP – из всех проблем, которые можно решить за полиномиальное время только на недетермированной машине Тьюринга: вариант обычной машины Тьюринга, которая может делать предположения. Машина предполагает решение проблемы – либо «удачно угадывая», либо перебирая все предположения параллельно – и проверяет свое предположение за полиномиальное время. Проблемы из данного класса обладают тем свойством, что легкорешаемой оказывается проверка: будет удачно угаданное решение проблемы верным или нет. Относительно разложение на множители чисел неизвестно, лежит ли эта проблема в классе P, хотя она из NP: достаточно угадать сомножители и проверить догадку, вычислив их произведение.

Важность NP в криптографии состоит в следующем: многие симметричные алгоритмы и алгоритмы с открытыми ключами могут быть взломаны за недетерминированное полиномиальное время. Для данного шифротекста *C*, криптоаналитик просто угадывает открытый текст, *X*, и ключ, *k*, и за полиномиальное время выполняет алгоритм шифрования со входами *X* и *k* и проверяет, равен ли результат *C*. Это имеет важное теоретическое значение, потому что устанавливает верхнюю границу сложности криптоанализа этих алгоритмов. На практике, конечно же, это выполняемый за полиномиальное время детерминированный алгоритм, который и ищет криптоаналитик. Более того, этот аргумент неприменим ко всем классам шифров, конкретно, он не применим для одноразовых блокнотов – для любого *C* существует множество пар *X*, *k*, дающих *C* при выполнении алгоритма шифрования, но большинство этих *X* представляют собой бессмысленные открытые тексты. Для криптографической стойкости необходимо существенно более сильное предположение, чем $P \neq NP$. Классы сложности показаны на рис. 2.1 [6].

Класс NP включает класс P, т.к. любая проблема, решаемая за полиномиальное время на детерминированной машине Тьюринга, будет также решена за полиномиальное время на недетерминированной машине Тьюринга, просто пропускается этап предположения. Если все NP проблемы решаются за полиномиальное время на детерминированной машине, то P =NP. Хотя кажется очевидным, что некоторые NP проблемы намного сложнее других (вскрытие алгоритма шифрования грубой силой против шифрования произвольного блока шифротекста), никогда не было доказано, что $P \neq NP$ (или P = NP). Однако большинство людей, работающих над теорией сложности, убеждены, что эти классы неравны. Можно доказать, что конкретные NP-проблемы настолько же трудны, как и любая проблема этого класса.

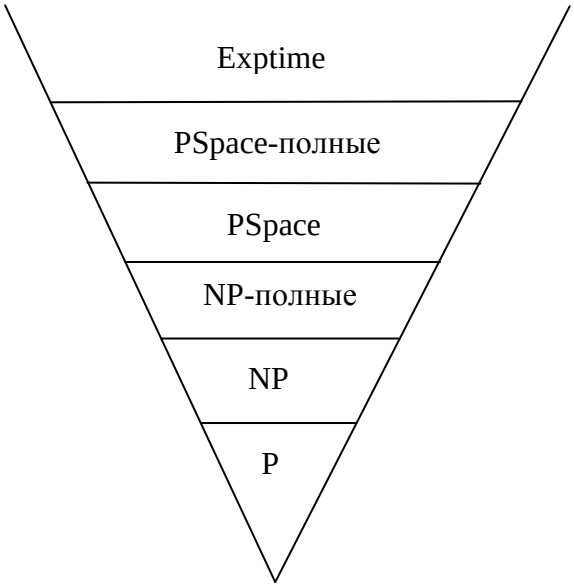

Рис. 2.1. **Классы сложности**

Стивен Кук (Steven Cook) доказал, что проблема выполнимости (Satisfiability problem, существует ли способ присвоить правильные значения входящим в него переменным так, чтобы все выражение стало истиной?) является NP-полной. Это означает, что, если проблема выполнимости решается за полиномиальное время, то P = NP. Наоборот, если может быть доказано, что для любой проблемы класса NP не существует детерминированного алгоритма с полиномиальным временем решения, доказательство покажет, что и для проблемы выполнимости не существует детерминированною алгоритма с полиномиальным временем решения. В NP нет проблемы труднее, чем проблема выполнимости.

Пространственная сложность определяется аналогично. Когда вход машины Тьюринга имеет длину n, то в исходной ситуации занято n ячеек ленты. В процессе вычислений могут понадобиться новые ячейки, их число и дает пространственную сложность. Полиномиальные ограничения могут быть рассмотрены и в этой ситуации. Это приводит к следующим классам. Класс PSPACE – проблемы класса PSPACE могут быть решены в полиномиальном пространстве, но не обязательно за полиномиальное время. PSPACE включает NP, но ряд проблем PSPACE кажутся сложнее, чем NP. Конечно, и это пока недоказуемо, существует класс проблем, так называемых PSPACE-полных, обладающих следующим свойством: если любая из них является NP-проблемой, то PSPACE = NP, и если любая из них является P-проблемой, то PSPACE = P.

Проблемы класса EXPTIME решаются за экспоненциальное время. Может быть действительно доказано, что EXPTIME-полные проблемы не

могут быть решены за детерминированное полиномиальное время. Также показано, что P ≠ EXPTIME.

NP-полные проблемы. Майкл Кэри (Michael Carey) и Дэвид Джонсон (David Johnson) составили список более чем 300 NP-полных проблем. Вот некоторые:

- Проблема путешествующего коммивояжера. Путешествующему коммивояжеру нужно посетить различные города, используя только один бак с горючим (существует максимальное расстояние, которое он может проехать). Существует ли маршрут, позволяющий ему посетить каждый город только один раз, используя этот единственный бак с горючим?
- Проблема тройного брака. В комнате n мужчин, n женщин и n чиновников. Есть список разрешенных браков, записи которого состоят из одного мужчины, одной женщины и одного регистрирующего чиновника. Дан этот список троек. Возможно ли построить n браков так, чтобы любой либо сочетался браком только с одним человеком, либо регистрировал только один брак?

2.4. Модульная арифметика

Целое число a делит другое число b, символически $a|b$, если и только если $b = da$ для некоторого целого числа d. В этом случае a называется делителем или множителем b. Пусть a – целое число, большее 1. Тогда a – простое число, если его единственными положительными делителями будут 1 и само a, в противном случае a называется составным. Любое целое $n > 1$ может быть представлено единственным образом с точностью до порядка сомножителей как произведение простых. Существенный с точки зрения криптографии факт состоит в том, что не известно никакого эффективного алгоритма разложения чисел на множители, хотя, с другой стороны, не было получено и никакой тривиальной нижней оценки временной сложности разложения. Никаких эффективных методов не известно даже в таком простом случае, когда необходимо восстановить два простых числа p и q из их произведения $n = pq$.

Наибольший общий делитель a и b, обозначение – НОД(a,b) или просто (a,b), есть наибольшее целое, делящее одновременно и a, и b. В эквивалентной форме (a,b) есть то единственное натуральное число, которое делит a и b и делится на любое целое, делящее и a, и b. Аналогично, наименьшее общее кратное, НОК(a,b), есть наименьшее натуральное число, делящееся и на a, и на b.

Наибольший общий делитель может быть вычислен с помощью алгоритма Евклида. Он состоит из следующей цепочки равенств:

$$
\begin{aligned}
&a = bq_1 + r_1, && 0<r_1 < b;\\
&b = r_1q_2 + r_2, && 0<r_2 < r_1;\\
&r_1 = r_2q_3 + r_3, && 0<r_3 < r_2;\\
&\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\\
&r_{k-2} = r_{k-1}q_k + r_k, && 0<r_k < r_{k-1};\\
&r_{k-1} = r_kq_{k+1}.
\end{aligned}
$$

Остановка гарантируется, поскольку остатки от деления r_i образуют строго убывающую последовательность натуральных чисел. Из этой цепочки немедленно получаем, что r_k есть общий делитель a и b и более того, что любой общий делитель a и b делит, в свою очередь, r_k. Таким образом, $r_k = (a,b)$.

Оценим теперь временную сложность этого алгоритма. Легко видеть, что алгоритм, выполняющий обычное деление, работает за квадратичное время. Итоговая оценка была бы все еще экспоненциальной, если бы выполнялось только $r_{i+1} < r_i$. Легко видеть, что $r_{i+2} < r_i / 2$ для всех i. Это дает верхнюю оценку $2\log_2 a$ для числа равенств. Таким образом, временная сложность в целом кубическая.

Считывая цепочку равенств снизу вверх, найдем суммарно за кубическое время целые числа x и y, такие, что

$$(a,b) = xa + yb.$$

Два целых a и b взаимно просты, если $(a,b) = 1$.

Функция Эйлера $\varphi(b)$, $n>1$, определяется как число неотрицательных $a<n$, таких, что a и n взаимно просты. Имеем $\varphi(1) = 1$ и $\varphi(p^b) = p^b - p^{b-1}$, где p – простое и $b \geq 1$. Легко также видеть, что $\varphi(mn) = \varphi(m)\varphi(n)$, если m и n взаимно просты. Опираясь на эти факты, можно вычислять $\varphi(n)$ для любого n. Это вычисление будет эффективным, если знать разложение n.

Говорят, что a сравнимо с b по модулю m,

$$a \equiv b (\mathrm{mod}\, m),$$

если m делит разность $a-b$. Число m называют модулем. Предполагается, что $m \geq 2$. Для любого целого x, в точности одно из чисел 0, 1,..., $m-1$ сравнимо с x по модулю m. Так, определенное число называется наименьшим неотрицательным остатком x по модулю m и обозначается $(x, \mathrm{mod}\, m)$.

Обозначив через $[x]$ целую часть x, т.е. наибольшее целое $\le x$, можно записать:

$$(x, \bmod m) = x - \left[\frac{x}{m}\right] \cdot m.$$

Если a и m взаимно просты, то тогда существуют x и y, такие, что $1 = xa + ym$. Отсюда $xa \equiv 1 (\bmod m)$. Целое число x называют обратным к a по модулю m и обозначают $a^{-1} (\bmod m)$. Обратное число определяется однозначно, если считать сравнимые числа равными. Сложность нахождения обратного числа примерно такая же, как и у алгоритма Евклида. Отсюда следует, что также и сравнение

$$az \equiv b (\bmod m), (a, m) = 1$$

может быть решено за кубическое время. Для нахождения z сперва вычисляем $a^{-1} (\bmod m)$ и умножаем его на b.

Если $(a, m) = 1$, то согласно теореме Эйлера

$$a^{\varphi(m)} \equiv 1 (\bmod m).$$

Если m – простое число, не делящее a, этот результат принимает вид

$$a^{m-1} \equiv 1 (\bmod m)$$

и называется малой теоремой Ферма.

Если модули m_i попарно взаимно просты, то система сравнений

$$x \equiv a_i (\bmod m_i),\ i = 1, \ldots,\ k,$$

имеет решение x, единственное с точностью до сравнений по модулю $M = m_1 \ldots m_k$. Этот результат известен как китайская теорема об остатках.

Вопросы и задания для самопроверки

2.1. Сформулируйте алгоритм полного перебора для поиска открытого текста.

2.2. Решение каких задач предполагает теоретико-сложностной подход определения криптостойкости систем?

2.3. Какими параметрами может оцениваться вычислительная сложность алгоритма?

2.4. Что такое нотация и для чего она используется?

2.5. Приведите классификацию алгоритмов в соответствии с их временной или пространственной сложностью.

2.6. Какой величиной характеризуется сложность вскрытия грубой силой, если длина ключа равна m?

2.7. Поясните принцип работы машины Тьюринга.

2.8. Какие проблемы называются решаемыми?

2.9. Приведите классификацию проблем.

2.10. В чем заключается важность NP-проблем в криптографии?

2.11. Приведите примеры NP-полных проблем.

2.12. Что понимают под НОД и НОК?

2.13. Сформулируйте алгоритм Евклида для вычисления НОД.

2.14. Дайте определение функции Эйлера.

2.15. В каком случае числа сравнимы по модулю? Приведите примеры.

2.16. Временная сложность алгоритма равна $4n^3 + 7n + 12n^4 + 44n$, определить вычислительную сложность:

а) $O(n^2)$; б) $O(n^3)$; в) $O(n)$; г) $O(n^4)$.

2.17. Если $T = O(2^n)$, то какая процедура удвоит время выполнения алгоритма:

а) увеличение входных данных в два раза;
б) увеличение входных данных в 1,5 раза;
в) добавление двух бит к входным данным;
г) добавление одного бита к входным данным.

2.18. Если сложность определяется величиной $O(n^m)$, то алгоритмы называются:

а) постоянными;
б) линейными;
в) полиномиальными;
г) экспоненциальными.

2.19. Сложность сперэкспоненциальных алгоритмов оценивается величиной:

а) $O\left(c^{f(n)}\right)$; б) $O(n^m)$; в) $O(n)$; г) $O(1)$.

2.20. Какой временной сложностью в идеале, должен обладать алгоритм, лучший для взлома спроектированной системы шифрования:

а) постоянной;
б) линейной;
в) полиномиальной;
г) экспоненциальной.

2.21. Из какого рода проблем состоит класс NP:

а) из всех проблем, которые можно решить за линейное время только на недетермированной машине Тьюринга;
б) из всех проблем, которые можно решить за полиномиальное время только на детермированной машине Тьюринга;
в) из всех проблем, которые можно решить за полиномиальное время только на недетермированной машине Тьюринга;
г) из всех проблем, которые можно решить за минимально возможное время только на недетермированной машине Тьюринга.

2.22. Какое выражение справедливо?

а) $\mathrm{P} \neq \mathrm{NP}$; б) $\mathrm{P} = \mathrm{NP}$; в) $\mathrm{P} \in \mathrm{NP}$; г) $\mathrm{P} \notin \mathrm{NP}$.

2.23. Определить вычислительную сложность алгоритма умножения вектора на квадратную матрицу.

2.24. Определить вычислительную сложность умножения прямоугольных матриц.

2.25. По алгоритму Евклида найти НОД чисел: 1065 и 45; 6066 и 478.

Модуль 3

МЕТОДЫ КРИПТОГРАФИЧЕСКОГО КОДИРОВАНИЯ ИНФОРМАЦИИ

Цель модуля – изучение слушателями курсов переподготовки традиционных методов шифрования и получение навыков по их использованию для криптографической защиты информации.

В результате изучения модуля слушатели переподготовки должны

знать:

- требования, предъявляемые к криптосистемам;
- методы шифрования с использованием шифров перестановки;
- алгоритмы шифрования с использование простой и сложной замены;
- технологию шифрования гаммированием и методы получения гаммы;

уметь

- использовать шифры перестановки, простой и сложной замены и гаммирования для криптографического кодирования;

иметь представление

- о надежности шифрования информации классическими методами.

3.1. Криптографическая защита компьютерной информации

3.1.1. Основные методы защиты компьютерной информации

Проблемой защиты информации путем ее преобразования занимается криптология (kryptos – тайный, logos – наука). Криптология разделяется на два направления – криптографию и криптоанализ [1]. Цели этих направлений прямо противоположны.

Криптография – прикладная наука, она использует самые последние достижения фундаментальных наук и, в первую очередь, математики [1]. С другой стороны, все конкретные задачи криптографии существенно зависят от уровня развития техники и технологии, применяемых средств связи и способов передачи информации.

Криптоанализ – наука (и практика её применения) о методах и способах вскрытия шифров [1]. Сферой интересов криптоанализа является исследование возможности расшифровывания информации без знания ключей.

Современная криптография включает в себя четыре раздела [1]:

- Симметричные криптосистемы.
- Криптосистемы с открытым ключом.
- Системы электронной подписи.
- Управление ключами.

Основные направления использования криптографических методов – передача конфиденциальной информации по каналам связи (например, электронная почта), установление подлинности передаваемых сообщений, хранение информации (документов баз данных) на носителях в зашифрованном виде.

Существует три возможности криптозащиты данных [1]:

- создание абсолютно надёжного, недоступного для других канала связи между абонентами. При современном уровне развития науки и техники сделать такой канал между удаленными абонентами для неоднократной передачи больших объёмов информации практически нереально;
- использование общедоступного канала связи, но сокрытие самого факта передачи информации. Разработкой средств и методов сокрытия факта передачи сообщения занимается стенография;
- использование общедоступного канала связи, но передача нужной информации в таком преобразованном виде, что восстановить её может только адресат. Разработкой методов преобразования (шифрования) информации с целью её защиты от незаконных пользователей занимается криптография. Такие методы и способы преобразования информации называются шифрами.

В качестве информации, подлежащей шифрованию и дешифрованию, рассмотрим тексты, построенные на некотором алфавите. Под этими терминами понимается следующее: алфавит – конечное множество используемых для кодирования информации знаков, текст – упорядоченный набор из элементов алфавита.

В качестве примеров алфавитов, используемых в современных ИС, можно привести: алфавит Z33 – 32 буквы русского алфавита и пробел; алфавит Z256 – символы, входящие в стандартные коды ASCII и КОИ-8; бинарный алфавит Z2 = {0,1}; восьмеричный алфавит или шестнадцатеричный алфавит.

Процесс кодирования сообщения называется шифрованием (или зашифровкой), а процесс декодирования – расшифровыванием (или расшифровкой). Само кодированное сообщение называется шифрованным (или просто шифровкой), а применяемый метод – шифром.

Основное требование к шифру состоит в том, чтобы расшифровка (и, может быть, зашифровка) была возможна только при наличии санкции, т.е. некоторой дополнительной информации (или устройства), которая называ-

ется ключом шифра. Процесс декодирования шифровки без ключа называется дешифрованием (или дешифрацией, или просто раскрытием шифра).

Ключ – информация, необходимая для беспрепятственного шифрования и дешифрования текстов [1].

Криптографическая система представляет собой семейство T преобразований открытого текста. Члены этого семейства индексируются, или обозначаются символом k; параметр k является ключом. Пространство ключей K' – это набор возможных значений ключа. Обычно ключ представляет собой последовательный ряд букв алфавита. Криптосистемы разделяются на симметричные и с открытым ключом.

В симметричных криптосистемах и для шифрования, и для дешифрования используется один и тот же ключ.

Термины распределение ключей и управление ключами относятся к процессам системы обработки информации, содержанием которых является составление и распределение ключей между пользователями.

Электронной (цифровой) подписью называется присоединяемое к тексту его криптографическое преобразование, которое позволяет при получении текста другим пользователем проверить авторство и подлинность сообщения [1].

Например, пусть пользователю A необходимо подписать сообщение x. Он, зная секрет K, находит такое y, что $F_K(y) = x$, и вместе с сообщением x посылает y пользователю B в качестве своей цифровой подписи. Пользователь B хранит y в качестве доказательства того, что A подписал сообщение x.

Сообщение, подписанное цифровой подписью, можно представлять себе как пару (x, y), где x – сообщение, y – решение уравнения.

$F_K(y) = x$, $F_K : X \to Y$ – функция с секретом, известная всем взаимодействующим абонентам. Из определения функции F_K очевидны следующие полезные свойства цифровой подписи:

1) подписать сообщение x, т.е. решить уравнение $F_K(y) = x$ может только абонент – обладатель данного секрета K; другими словами, подделать подпись невозможно;

2) проверить подлинность подписи может любой абонент, знающий открытый ключ, т.е. саму функцию F_K;

3) при возникновении споров отказаться от подписи невозможно в силу её подлинности;

4) подписанные сообщения (x, y) можно, не опасаясь ущерба, пересылать по любым каналам связи.

Криптостойкостью называется характеристика шифра, определяющая его стойкость к дешифрованию без знания ключа (т.е. криптоанализу). Имеется несколько показателей криптостойкости, среди которых:

- количество всех возможных ключей;
- среднее время, необходимое для криптоанализа.

Преобразование T_k определяется соответствующим алгоритмом и значением параметра k. Эффективность шифрования с целью защиты информации зависит от сохранения тайны ключа и криптостойкости шифра.

Таким образом, криптография представляет собой совокупность методов кодирования данных, направленных на то, чтобы сделать эти данные бесполезными для противника. Такое кодирование позволяет решить две главные проблемы защиты данных: проблему секретности – лишение противника возможности извлечь информацию из канала передачи и проблему имитостойкости – лишение противника возможности ввести ложную информацию в канал передачи или изменить сообщение так, чтобы изменился его смысл.

3.1.2. Основные модели криптосистем

Проблемы секретности и имитостойкости тесно связаны между собой, поэтому методы решения одной из них часто применимы для решения другой [5]. Из двух названных проблем секретность рассматривается первой, как наиболее исследованная на протяжении веков. На рис. 3.1 представлена модель канала передачи данных, обеспечивающая секретность благодаря криптографическому кодированию.

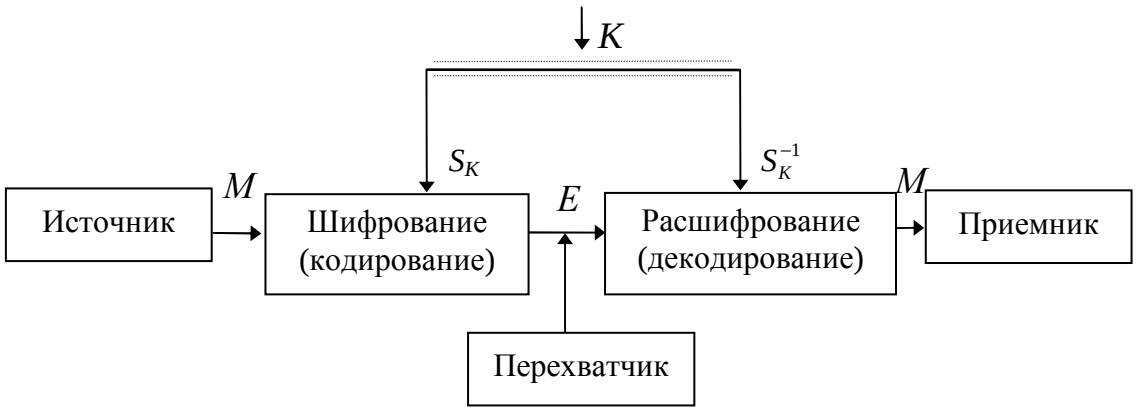

Рис. 3.1. **Модель канала передачи с криптографическим кодированием**

Источник информации генерирует открытый текст или незашифрованное сообщение M, которое должно быть передано соответствующему получателю по незащищенному каналу, за которым следит перехватчик. Для того чтобы перехватчик не смог распознать сообщение M, отправитель

шифрует (кодирует) его с помощью обратимого кодирования (преобразования) S_K и получает криптограмму или шифрованный текст $E = S_K(M)$, который отправляет получателю.

Законный получатель, приняв криптограмму E, декодирует (расшифровывает) ее с помощью обратного преобразования S_K^{-1} и получает исходное сообщение в виде открытого сообщения M: $S_K(E) = S_K^{-1}(S_K(M)) = M$.

Преобразование S_K выбирается из семейства криптографических преобразований, называемых криптографической системой или общей системой.

Параметр, выбираемый в качестве отдельного преобразования, называется криптографическим ключом или просто ключом. Общая система – это набор инструкций (часть аппаратуры или программа ЭВМ), с помощью которых можно закодировать открытый текст и декодировать шифрованный текст различными способами, один из которых выбирается с помощью конкретного ключа. Формально, криптографическая система – это однопараметрическое семейство обратимых преобразований кодирования S_K: $M \rightarrow E$ из пространства M сообщений открытых данных в пространство E (закодированных шифрованных сообщений). Причем ключ K_i выбирается из конечного множества K, называемого пространством ключей.

Обычно общая система, являющаяся семейством преобразований, рассматривается как общедоступная система. С одной стороны, то, что открытая для всех часть называется общей системой, отражает очень важное правило техники криптографического кодирования: защищенность системы не должна зависеть от секретности чего-либо такого, что нельзя быстро изменить в случае утечки секретной информации. Обычно общая система является некоторой совокупностью аппаратуры, которую можно изменить только со значительными затратами времени и средств, тогда как ключ является легко и просто изменяемым объектом. Криптографическая система подобна кодовому замку – структура замка известна любому, кто его приобрел, однако конкретная используемая комбинация не известна и может быть изменена всякий раз, когда есть подозрение, что она стала известна постороннему лицу. Даже если противник знает множество всех возможных комбинаций, он может все же оказаться не в состоянии определить, какая из них правильна.

Поскольку вся секретность сосредоточена в секретности ключа, то его надо передавать отправителю и получателю по защищенному каналу распространения ключей. На рис. 3.1 этот канал показан экранированной линией. В системе используются одинаковые секретные ключи в блоке шифрования и блоке расшифровывания.

На рис. 3.2 представлена модель канала передачи данных, обеспечивающая секретность с использованием асимметричного криптографиче-

ского кодирования. В этой криптосистеме один из ключей является открытым, а другой – секретным.

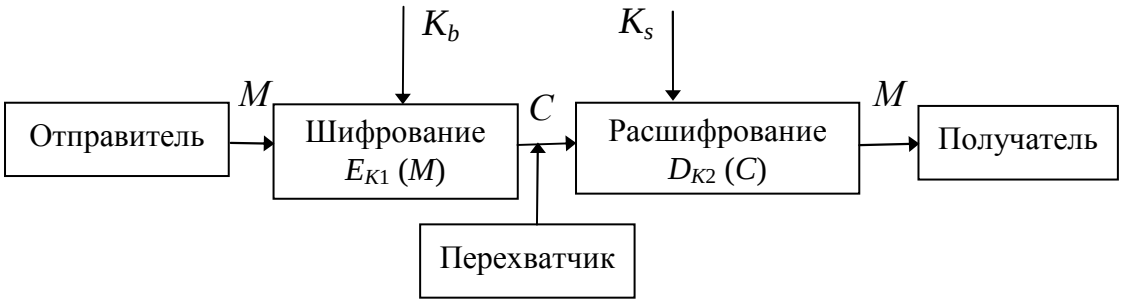

Рис. 3.2. **Обобщенная схема асимметричной криптосистемы**

В асимметричной криптосистеме для зашифровывания данных используется один ключ, а для расшифровывания – другой (отсюда и название – асимметричная) [5]. Первый ключ является открытым и может быть опубликован для использования всеми пользователями системы, которые зашифровывают данные. Расшифровывание данных с помощью открытого ключа невозможно.

Для расшифровывания данных получатель зашифрованной информации использует второй ключ, который является секретным. Разумеется, ключ расшифровывания не может быть определен из ключа зашифровывания.

В асимметричных системах нет необходимости в защищенном канале для передачи ключей, т.к. открытый ключ передается по незащищенному каналу.

Любая попытка со стороны перехватчика расшифровать криптограмму E для получения открытого текста M или зашифровать свой собственный текст M' для получения приемлемой криптограммы E' без получения ключа из канала распространения ключей называется криптоанализом. Если криптоанализ невозможен и криптоаналитик не может вывести M из E или E' из M' без предварительного получения ключа, то такая криптографическая система называется криптостойкой.

3.1.3. Требования к криптосистемам

Процесс криптографического закрытия данных может осуществляться как программно, так и аппаратно. Аппаратная реализация отличается существенно большей стоимостью, однако ей присущи и преимущества: высокая производительность, простота, защищенность и т.д. Программная реализация более практична, допускает известную гибкость в использовании.

Для современных криптографических систем защиты информации сформулированы следующие общепринятые требования [6]:

- зашифрованное сообщение должно поддаваться чтению только при наличии ключа;
- число операций, необходимых для определения использованного ключа шифрования по фрагменту шифрованного сообщения и соответствующего ему открытого текста, должно быть не меньше общего числа возможных ключей;
- число операций, необходимых для расшифровывания информации путем перебора всевозможных ключей должно иметь строгую нижнюю оценку и выходить за пределы возможностей современных компьютеров (с учетом возможности использования сетевых вычислений);
- знание алгоритма шифрования не должно влиять на надежность защиты;
- незначительное изменение ключа должно приводить к существенному изменению вида зашифрованного сообщения даже при использовании одного и того же ключа;
- структурные элементы алгоритма шифрования должны быть неизменными;
- дополнительные биты, вводимые в сообщение в процессе шифрования, должен быть полностью и надежно скрыты в шифрованном тексте;
- длина шифрованного текста должна быть равной длине исходного текста;
- не должно быть простых и легко устанавливаемых зависимостей между ключами, последовательно используемыми в процессе шифрования;
- любой ключ из множества возможных должен обеспечивать надежную защиту информации;
- алгоритм должен допускать как программную, так и аппаратную реализацию, при этом изменение длины ключа не должно вести к качественному ухудшению алгоритма шифрования.

К. Шеннон определил точную математическую модель понятия безопасности криптосистемы. Смысл работы криптоаналитика состоит в определении ключа K, открытого текста P или и того, и другого. Однако его может устроить и некоторая вероятностная информация о P: является ли этот открытый текст оцифрованным звуком, немецким текстом, данными электронных таблиц или еще чем-нибудь.

В реальном криптоанализе у криптоаналитика есть некоторая вероятностная информация о P еще до начала работы. Он, скорее всего, знает язык открытого текста. Этот язык обладает определенной, связанной с ним

избыточностью. Если это сообщения для друга, оно, возможно, начинается словом «Привет». Целью криптоаналитика является изменение вероятностей, связанных с каждым возможным открытым текстом. В конце концов, из всех возможных открытых текстов будет выбран один конкретный (или, но крайней мере, весьма вероятный).

Существуют криптосистемы, достигающие совершенной безопасности. Такой является криптосистема, в которой шифротекст не дает никакой информации об открытом тексте (кроме, возможно, его длины). К. Шеннон теоретически показал, что такое возможно только, если число возможных ключей также велико, как и число возможных сообщений. Другими словами, ключ должен быть не короче самого сообщения и не может использоваться повторно. Это означает, что единственной системой, которая достигает идеальной безопасности, может быть только криптосистема с одноразовым блокнотом. За исключением идеально безопасных систем шифротекст неизбежно даст определенную информацию о соответствующем шифротексте. Хороший криптографический алгоритм сохраняет минимум этой информации, хороший криптоаналитик пользуется этой информацией для определения открытого текста. Криптоаналитики используют естественную избыточность языка для уменьшения числа возможных открытых текстов. Чем избыточнее язык, тем легче его криптоанализировать. По этой причине многие криптографические реализации перед шифрованием используют программы сжатия для уменьшения размера текста. Сжатие уменьшает избыточность сообщения вместе с объемом работы, необходимым для его шифрования и дешифрования. В общем случае, чем больше энтропия, тем тяжелее взломать криптосистему.

Двумя основными методами маскировки избыточности открытого текста сообщения, согласно К. Шеннону, служат путаница и диффузия [6]. Путаница маскирует связь между открытым текстом и шифротекстом. Она затрудняет попытки найти в шифротексте избыточность и статистические закономерности. Простейшим путем создания путаницы является подстановка. В простом подстановочном шифре, например, шифре Цезаря, все одинаковые буквы открытого текста заменяются другими одинаковыми буквами шифротекста. Современные подстановочные шифры являются более сложными: длинный блок открытого текста заменяется блоком шифротекста, и способ замены меняется с каждым битом открытого текста или ключа. Такого типа подстановки обычно недостаточно – сложный алгоритм немецкой Энигмы был взломан в ходе второй мировой войны.

Диффузия рассеивает избыточность открытого текста, распространяя её по всему шифротексту. Простейший способ создания диффузии – транспози-

ция (также называемая перестановкой). Простой перестановочный шифр только переставляет буквы открытого текста. Современные шифры также выполняют такую перестановку, но они используют и другие формы диффузии, которые позволяют разбросать части сообщения по всему сообщению.

Потоковые шифры используют только путаницу, хотя ряд схем с обратной связью добавляют диффузию. Блочные алгоритмы применяют и путаницу, и диффузию. Как правило, диффузию саму по себе несложно взломать (хотя шифры с двойной перестановкой оказываются поустойчивее, чем другие некомпьютерные системы).

3.2. Шифры перестановки. Шифры простой замены

3.2.1. Шифры перестановки

При шифровании перестановкой символы шифруемого текста переставляются по определенному правилу в пределах блока этого текста [5].

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

УЛИЦА БЛОХИНА ДОМ ВОСЕМЬ КВАРТИРА ДЕВЯТЬ

записывается в таблицу поочередно по столбцам. Результат заполнения таблицы из 5 строк и 7 столбцов показан на рис. 3.3. После заполнения таблицы текстом сообщения по столбцам для формирования шифротекста считывают содержимое таблицы по строкам. Если шифротекст записывать группами по пять букв, получается такое шифрованное сообщение:

УБНВЬТЕЛЛАОКИВИОДСВРЯЦХОЕААТАИММРДЬ.

У	Б	Н	В	Ь	Т	Е
Л	Л	А	О	К	И	В
И	О	Д	С	В	Р	Я
Ц	Х	О	Е	А	А	Т
А	И	М	М	Р	Д	Ь

Рис. 3.3. **Заполнение сообщения в таблице из 5 строк и 7 столбцов**

Естественно, отправитель и получатель сообщения должны заранее условиться об общем ключе в виде размера таблицы. Следует заметить, что объединение букв шифротекста в 5-буквенные группы не входит в ключ шифра и осуществляется для удобства записи несмыслового текста. При расшифровывании 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 КАФЕДРА, а текст сообщения возьмем из предыдущего примера. На рис. 3.4 показаны две таблицы, заполненные текстом сообщения и ключевым словом, при этом левая таблица соответствует заполнению до перестановки, а правая таблица – заполнению после перестановки.

Ключ →

К	А	Ф	Е	Д	Р	А
5	1	7	4	3	6	2
У	Б	Н	В	Ь	Т	Е
Л	Л	А	О	К	И	В
И	О	Д	С	В	Р	Я
Ц	Х	О	Е	А	А	Т
А	И	М	М	Р	Д	Ь

До перестановки

А	А	Д	Е	К	Р	Ф
1	2	3	4	5	6	7
Б	Е	Ь	В	У	Т	Н
Л	В	К	О	Л	И	А
О	Я	В	С	И	Р	Д
Х	Т	А	Е	Ц	А	О
И	Ь	Р	М	А	Д	М

После перестановки

Рис. 3.4. **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 пять букв получим шифрованное сообщение:

БЕЬВУТН ЛВКОЛИА ОЯВСИРД ХТАЕЦАО ИЬРМАДМ

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 отдельно для строк. Сначала в таблицу записывается текст сообщения, а потом поочередно переставляются столбцы, а затем строки.

При расшифровывании порядок перестановок должен быть обратным. Пример выполнения шифрования методом двойной перестановки показан на рис. 3.5. Если считывать шифротекст из правой таблицы построчно блоками по четыре буквы, то получится следующее:

ОБАЛ ЬЯПТ ЦЛУИ АИХН.

Ключом к шифру двойной перестановки служит последовательность номеров столбцов и номеров строк исходной таблицы (в нашем примере последовательности 3241 и 2314 соответственно).

Исходная таблица

	3	2	4	1
2	У	Л	И	Ц
3	А	Б	Л	О
1	Х	И	Н	А
4	П	Я	Т	Ь

Перестановка столбцов

	1	2	3	4
3	Ц	Л	У	И
1	О	Б	А	Л
4	А	И	Х	Н
2	Ь	Я	П	Т

Перестановка строк

	1	2	3	4
1	О	Б	А	Л
2	Ь	Я	П	Т
3	Ц	Л	У	И
4	А	И	Х	Н

Рис. 3.5. **Шифрование методом двойной перестановки**

Число вариантов двойной перестановки быстро возрастает при увеличении размера таблицы: для таблицы 3×3 – 36 вариантов, 4×4 – 576, 5×5 – 14 400.

Однако двойная перестановка не отличается высокой стойкостью и сравнительно просто «взламывается» при любом размере таблицы шифрования.

3.2.2. Шифры простой замены

Подстановка Цезаря

Подстановка Цезаря является самым простым вариантом подстановки. Она относится к группе моноалфавитных подстановок. Этот шифр реализует следующее преобразование открытого текста: каждая буква открытого текста заменяется третьей после неё буквой в алфавите, который считается написанным по кругу, т.е. после буквы «я» следует буква «а». Отметим, что Цезарь заменял букву третьей после неё буквой, но можно заменять и какой-нибудь другой. Главное, чтобы тот, кому посылается сообщение, знал эту величину сдвига. Класс шифров, к которым относится шифр Цезаря, называется шифрами замены.

Подмножество $C_m = \{C_k: 0 \le k < m\}$ симметрической группы $\mathrm{SYM}(Z_m)$, содержащее m подстановок: $C_k: j \rightarrow (j + k) \pmod m$, $0 \le k < m$, называется подстановкой Цезаря.

Умножение коммутативно, $C_kC_j = C_jC_k = C_{j+k}$, C – идентичная подстановка, а обратной к C_k является $C_k^{-1} = C_{m-k}$, где $0 \le k < m$. Семейство подстановок Цезаря названо по имени римского императора Гая Юлия Цезаря, который поручал Марку Туллию Цицерону составлять послания с использованием 50-буквенного алфавита и подстановки C_3.

Подстановка определяется по таблице замещения, содержащей пары соответствующих букв «исходный текст – шифрованный текст». Для C_3 подстановки приведены ниже. Стрелка ($\rightarrow$) означает, что буква исходного текста (слева) шифруется при помощи C_3 в букву шифрованного текста (справа).

Таким образом, система Цезаря является моноалфавитной подстановкой, преобразующая n-грамму исходного текста $(x_0, x_1, ..., x_{n-1})$ в n-грамму шифрованного текста $(y_0, y_1, ..., y_{n-1})$ в соответствии с правилом

$$y_i = C_k(x_i),\ 0 \le i < n.$$

А→г	Й→м	Т→х	Ъ→э
Б→д	К→н	У→ц	Ы→ю
В→е	Л→о	Ф→ч	Ь→я
Г→ж	М→п	Х→ш	Э→ _
Д→з	Н→р	Ц→щ	Ю→а
Е→и	О→с	Ч→ъ	Я→б
Ж→й	П→т	Ш→ы	_ →в
З→к	Р→у	Щ→ь	
И→л	С→ф	Ъ→э	

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

Например, известное послание Цезаря VENI VIDI VICI («Пришел, увидел, победил») выглядело бы в зашифрованном виде так:

YHQL YLGL YLFL.

Достоинством системы шифрования Цезаря является простота шифрования и расшифровывания. К недостаткам системы Цезаря можно отнести следующее [5]:

- 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, например, в английском языке. Буква с наивысшей частотой появления в шифротексте заменяется на букву с наивысшей частотой появления в английском языке 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 многочисленные модификации.

Аффинная система подстановок Цезаря

В системе шифрования Цезаря использовались только аддитивные свойства множества чисел. Однако символы можно также умножать по модулю m. Применяя одновременно операции сложения и умножения по модулю m над буквами алфавита, можно получить систему подстановок, которую называют аффинной системой подстановок Цезаря.

В данном преобразовании буква, соответствующая числу t, заменяется на букву, соответствующую числовому значению $(at+b)$ по модулю m. Следует заметить, что данное преобразование является взаимно однозначным отображением тогда и только тогда, когда НОД (a, m) – наибольший общий делитель чисел a и m равен 1, т.е. если a и m – взаимно простые числа.

Пример. Пусть $m = 26$, $a = 5$, $b = 7$. Тогда, очевидно, НОД (5, 26) = 1, и мы получаем следующее соответствие между числовыми кодами букв:

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
$5t + 7$	7	12	17	22	1	6	11	16	21	0	5	10	15	20	25	4	9	14	19	24	3	8	13	18	23	2

Преобразуя числа в буквы английского языка, получаем следующее соответствие для букв открытого текста и шифротекста:

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

Исходное сообщение CLOSE преобразуется в шифротекст RKZTB.

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. Аффинная система использовалась на практике несколько веков назад, а сегодня ее применение ограничивается большей частью иллюстрациями основных криптологических положений.

3.2.3. Одноразовые системы

Шифры сложной замены называют многоалфавитными, т.к. для шифрования каждого символа исходного сообщения применяют свой шифр простой замены. Многоалфавитная подстановка последовательно и циклически меняет используемые алфавиты.

Многоалфавитная подстановка определяется ключом, содержащим не менее двух различных подстановок. В начале рассмотрим многоалфавитные системы подстановок с нулевым начальным смещением.

Система одноразового использования преобразует исходный текст

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

в шифрованный текст

$$Y = (Y_0, Y_1, ..., Y_{n-1})$$

при помощи подстановки

$$Y_i = C_{k_i}(X_i) = (K_i + X_i) \pmod m \ (i = 0...n-1).$$

Для такой системы подстановки используют также термин «одноразовая лента» и «одноразовый блокнот». Пространство ключей K' системы одноразовой подстановки является вектором рангов $(K_0, K_1, ..., K_{n-1})$ и содержит m^n точек.

Эффект использования многоалфавитной подстановки заключается в том, что обеспечивается маскировка естественной статистики исходного языка, т.к. 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.

В классическом понимании одноразовый блокнот является большой неповторяющейся последовательностью символов ключа, распределенных случайным образом, написанных на кусочках бумаги и приклеенных к листу блокнота. Он был изобретен в 1917 году. Первоначально это была одноразовая лента для телетайпов. Отправитель использовал каждый символ ключа блокнота для шифрования только одного символа открытого текста. Шифрование представляет собой сложение по модулю 26 (для английского алфавита) символа открытого текста и символа ключа из одноразового блокнота.

Каждый символ ключа используется только единожды и для единственного сообщения. Отправитель шифрует сообщения и уничтожает использованные страницы блокнота или использованную часть ленты. Получатель, в свою очередь, используя точно такой же блокнот, дешифрирует каждый символ шифротекста. Расшифровав сообщение, получатель уничтожает соответствующие страницы блокнота или часть ленты. Новое сообщение – новые символы ключа. Например, если сообщением является ONE, а ключевая последовательность в блокноте – TBF, то шифротекст будет выглядеть как IPK:

$$O + T \bmod 26 = I;$$

$$N + B \bmod 26 = P;$$

$$E + F \bmod 26 = K.$$

Таким образом, одноразовая система шифрует исходный открытый текст $X = (X_0, X_1,..., X_{n-1})$ в шифротекст $Y = (Y_0, Y_1, \ldots, Y_{n-1})$ посредством подстановки Цезаря $Y_i = (X_i + K_i) \bmod m$, $0 \le i < n$, где K_i – i-й элемент случайной ключевой последовательности.

Процедура расшифровывания описывается соотношением

$$X_i = (Y_i - K_i) \bmod m,$$

где K_i – i-й элемент той же самой случайной ключевой последовательности.

В предположении, что злоумышленник не сможет получить доступ к одноразовому блокноту, использованному для шифрования сообщения, эта схема совершенно безопасна. Данное шифрованное сообщение на вид соответствует любому открытому сообщению того же размера. Поскольку все ключевые последовательности совершенно одинаковы (помните, символы ключа генерируемся случайным образом), у противника отсутствует информация, позволяющая подвергнув шифротекст криптоанализу.

Поскольку все открытые тексты равновероятны, у криптоаналитика нет возможности определить, какой из открытых текстов является правильным. Случайная ключевая последовательность, сложенная с неслучайным открытым текстом, даст совершенно случайный шифротекст, и никакие вычислительные мощности не смогут это изменить.

Необходимо напомнить, что символы ключа должны генерироваться случайным образом. Любые попытки вскрыть такую схему сталкиваются со способом, который создает последовательность символов ключа. Использование генераторов псевдослучайных чисел не считается, у них всегда неслучайные свойства. Если вы используете действительно случайный источник, что намного труднее, чем кажется на первый взгляд – это совершенно безопасно.

Другой важный момент: ключевую последовательность никогда нельзя использовать второй раз. Даже если используется блокнот размером в несколько гигабайт, то если криптоаналитик получит несколько текстов с перекрывающимися ключами, он сможет восстановить открытый текст. Он сдвинет каждую пару шифротекстов относительно друг друга и подсчитает число совпадений в каждой позиции. Если шифротексты смещены правильно, соотношение совпадений резко возрастет – точное значение зависит от языка открытого текста. С этой точки зрения криптоанализ не представляет труда. Это похоже на показатель совпадений, но сравниваются два различных периода. Не используйте ключевую последовательность повторно.

Идея одноразового блокнота легко расширяется на двоичные данные. Вместо одноразового блокнота, состоящего из букв, используется одноразовый блокнот из битов. Вместо сложения открытого текста с ключом одноразового блокнота используйте XOR (операцию сложения по mod 2: $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$). Для дешифрирования применяется XOR к шифротексту с тем же одноразовым блокнотом. Все остальное не меняется, и безопасность остается такой же совершенной.

Существует несколько проблем. Поскольку ключевые биты должны быть случайными и не могут использоваться снова, длина ключевой последовательности должна равняться длине сообщения. Одноразовый блокнот удобен для нескольких небольших сообщений, но его нельзя использовать для работы по каналу связи с пропускной способностью 1,544 Мбит/с. Можно хранить 650 Мбайт случайных данных на CD-ROM, но и тут есть проблемы. Во-первых, вам нужно только две копии случайных битов, но CD-ROM экономичны только при больших тиражах. И, во-вторых, нужно уничтожать использованные биты. Для CD-ROM нет другой возможности

удалить информацию, кроме как физически разрушить весь диск. Гораздо больше подходит цифровая лента.

Даже если проблемы распределения и хранения ключей решены, придется точно синхронизировать работу отправителя и получателя. Если получатель пропустит бит (или несколько бит пропадут при передаче), сообщение потеряет всякий смысл. С другой стороны, если несколько бит изменятся при передаче (и ни один бит не будет удален или добавлен – что гораздо больше похоже на влияние случайного шума), то лишь эти биты будут расшифрованы неправильно. Но одноразовый блокнот не обеспечивает проверку подлинности.

Одноразовые блокноты используются и сегодня, главным образом для сверхсекретных каналов связи с низкой пропускной способностью. По слухам «горячая линия» между Соединенными Штатами и бывшим Советским Союзом шифровалась с помощью одноразового блокнота [1]. Многие сообщения шпионов зашифрованы с использованием одноразовых блокнотов. Эти сообщения нераскрыты сегодня и навсегда останутся нераскрытыми. На этот факт не повлияет время работы суперкомпьютеров над этой проблемой. Наложение белого шума в виде бесконечного ключа на исходный текст меняет статистические характеристики языка источника. Системы одноразового использования теоретически не расшифруемы, т.к. не содержат достаточной информации для восстановления текста.

Почему же эти системы неприменимы для обеспечения секретности при обработке информации? Ответ простой – они непрактичны, т.к. требуют независимого выбора значения ключа для каждой буквы исходного текста.

3.2.4. Система шифрования Вижинера

Последовательность ключа вида

$$K = (k_0, k_1, ..., k_{n-1}),$$

которая называется ключом пользователя, и продлим ее до бесконечной последовательности, повторяя цепочку. Таким образом, получим рабочий ключ

$$K = (k_0, k_1, ..., k_{n-1}), K_j = K_{j \bmod r}, 0 \le j < \infty.$$

Например, при $r = \infty$ и ключе пользователя 15 8 2 10 11 4 18 рабочий ключ будет периодической последовательностью

15 8 2 10 11 4 18 15 8 2 10 11 4 18 15 8 2 10 11 4 18 ... .

Подстановка Вижинера VIG_K определяется как

$$VIG_K: (X_0, X_1, ..., X_{n-1}) \to (Y_0, Y_1, ..., Y_{n-1}) = (X_0 + k_0, X_1 + k_1, .., X_{n-1} + k_{n-1}).$$

Таким образом:

1) исходный текст X делится на r *фрагментов*:

$$X_i = (X_i, X_{i+r}, ..., X_{i+r(n-1)}),\ 0 \le i < r;$$

2) i-й фрагмент исходного текста X_i шифруется при помощи подстановки Цезаря:

$$C_k: (X_i, X_{i+r}, ..., X_{i+r(n-1)}) \to (Y_i, Y_{i+r}, ..., Y_{i+r(n-1)}).$$

Вариант системы подстановок Вижинера при $m = 2$ называется системой Вернама (1917 г).

В то время ключ $K = (k_0, k_1, ..., k_{n-1})$ записывался на бумажной ленте. Каждая буква исходного текста в алфавите, расширенном некоторыми дополнительными знаками, сначала переводилась с использованием кода Бодо в пятибитовый символ. К исходному тексту Бодо добавлялся ключ (по mod2). Старинный телетайп фирмы AT&T со считывающим устройством Вернама и оборудованием для шифрования использовался корпусом связи армии США.

Очень распространена плохая, с точки зрения секретности, практика использовать слово или фразу в качестве ключа для того, чтобы $k = (k_0, k_1, ..., k_{n-1})$ было легко запомнить. В ИС для обеспечения безопасности информации это недопустимо. Для получения ключей должны использоваться программные или аппаратные средства случайной генерации ключей.

Таким образом, 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 рис. 3.6 и 3.7 показаны таблицы Вижинера для русского и английского алфавитов соответственно [5]. Таблица Вижинера используется как для зашифровывания, так и для расшифровывания. 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 выпи-

сывают 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.

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

Рис. 3.6. **Таблица Вижинера для русского алфавита**

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

Рис. 3.7. **Таблица Вижинера для английского алфавита**

Рассмотрим пример получения шифротекста с помощью таблицы Вижинера. Пусть выбрано ключевое слово

АЛГОРИТМ.

Необходимо зашифровать сообщение

УЛИЦА БЛОХИНА ПЯТЬ.

Выпишем исходное сообщение в строку и запишем под ним ключевое слово с повторением. В третью строку будем выписывать буквы шифротекста, определяемые из таблицы Вижинера:

Сообщение	У	Л	И	Ц	А	Б	Л	О	Х	И	Н	А	П	Я	Т	Ь
Ключ	А	Л	Г	О	Р	И	Т	М	А	Л	Г	О	Р	И	Т	М
Шифротекст	У	Ц	Л	Д	Р	Й	Э	Ь	Х	У	Р	А	Я	З	Д	Ж

3.2.5. Шифрование методом гаммирования

Гаммирование является также широко применяемым криптографическим преобразованием. На самом деле граница между гаммированием и использованием бесконечных ключей и шифров Вижинера, о которых речь шла выше, весьма условная.

Гамма шифра – это псевдослучайная последовательность, выработанная по заданному алгоритму для зашифровывания открытых данных и расшифровывания принятых данных [1].

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

Следует отметить, что перед зашифровыванием открытые данные разбивают на блоки $T_0^{(i)}$ одинаковой длины, обычно по 64 бита. Гамма шифра вырабатывается в виде последовательности блоков $Г_ш^{(i)}$ аналогичной длины. Уравнение шифрования можно записать в виде $T_ш^{(i)} = Г_ш^{(i)} \oplus T_0^{(i)}$, $i = 1 \ldots M$, где $T_ш^{(i)}$ – i-й блок шифротекста; $Г_ш^{(i)}$ – i-й блок гаммы шифра; $T_0^{(i)}$ – i-й блок открытого текста; M – количество блоков открытого текста.

Процесс расшифровывания сводится к повторной генерации гаммы шифра и наложению этой гаммы на принятые данные. Уравнение расшифровывания имеет вид $T_0^{(i)} = Г_ш^{(i)} \oplus T_ш^{(i)}$.

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.

Ниже рассматриваются наиболее распространенные методы генерации гамм, которые могут быть использованы на практике.

Чтобы получить линейные последовательности элементов гаммы, длина которых превышает размер шифруемых данных, используются датчики псевдослучайных последовательностей (ПСП). На основе теории групп было разработано несколько типов таких датчиков.

В настоящее время наиболее доступными и эффективными являются конгруэнтные генераторы ПСП. Для этого класса генераторов можно сделать математически строгое заключение о том, какими свойствами обладают выходные сигналы этих генераторов с точки зрения периодичности и случайности. Рассмотрим пример генераторов ПСП.

Одним из хороших конгруэнтных генераторов является линейный конгруэнтный датчик ПСП. Он вырабатывает последовательности псевдослучайных чисел $X(i)$, описываемые соотношением [6]:

$$X_{(i+1)} = (A \cdot X_{(i)} + C) \bmod m,$$

где A и C – константы; $X(0)$ – исходная величина, выбранная в качестве порождающего числа.

Очевидно, что эти три величины и образуют ключ.

Такой датчик ПСП генерирует псевдослучайные числа с определенным периодом повторения, зависящим от выбранных значений A и C. Значение m обычно устанавливается равным 2^n, где n – длина машинного слова в битах. Датчик имеет максимальный период M до того, как генерируемая последовательность начнет повторяться. По причине, отмеченной ранее, необходимо выбирать числа A и C такие, чтобы период M был максимальным. Как показано Д. Кнутом, линейный конгруэнтный датчик ПСП имеет максимальную длину M тогда и только тогда, когда C – нечетное, и $A \equiv 1 (\bmod 4)$.

Генератор ПСП $X_t \in A = \{0, 1, ..., N-1\}$ может быть построен также на основе квадратичного рекуррентного соотношения

$$X_{t+1} = \left(dX_t^2 + aX_t + c\right) \bmod N,\ t = 0, 1, ...,$$

где $X_0,\ a,\ c,\ d \in A$ – параметры генератора.

Генератор имеет максимальный период, $T_{\max} = N$, когда:

1) $c,\ N$ – взаимно простые числа;
2) $d,\ a-1$ – кратны p, p – любой нечетный простой делитель N;
3) d – четное число, причем:

$$d = \begin{cases} (a-1) \bmod 4, & \text{если } N \text{ кратно } 4; \\ (a-1) \bmod 4, & \text{если } N \text{ кратно } 2; \end{cases}$$

4) если N кратно 9, то либо $d \bmod 9 = 0$, либо $d \bmod 9 = 1$ и $cd \bmod 9 = 6$.

Генератор Эйхенауэра – Лена с обращением

Нелинейная ПСП Эйхенауэра – Лена с обращением определяется рекурсивным соотношением ($t = 0, 1, ...$):

$$X_{t+1} = \begin{cases} \left(aX_t^{-1} + c\right) \bmod N, & \text{если } X_t \geq 1; \\ c, & \text{если } X_t = 0, \end{cases}$$

где $X_t^{-1} \in A$ – обратный к X_t по $\bmod N$, т.е. $X_t^{-1} X_t = 1 \bmod N$;

$X_0,\ a,\ c \in A$ – параметры генератора;

Если $N = 2^q$, a, X_0 – нечетны, c – четно, то генератор имеет максимально возможный период $T_{\max} = 2^{q-1}$, когда:

$$a = 1 \bmod,\ 4c = 2 \bmod 4.$$

Для шифрования данных с помощью датчика ПСП может быть выбран ключ любого размера. Например, пусть ключ состоит из набора чисел $X(j)$ размерностью b, где $j = 1, 2, ..., n$. Тогда создаваемую гамму шифра G можно представить как объединение непересекающихся множеств $H(j)$.

Например, требуется закодировать методом гаммирования сообщение: МАЙ. Для данного примера необходимо сгенерировать гамму шифра длиной 3 символа. Согласно требованиям к генератору ПСП выберем константы: A = 5, т.к. $5 \equiv 1 (\bmod 4)$, C = 3, m = 32, $X_0 = 8$, которые являются секретным ключом для данного алгоритма. Используя представленное выше выражение вычислим:

$$X_1 = (5 \cdot 8 + 3) \bmod 32 = 11;$$

$$X_2 = (5 \cdot 11 + 3) \bmod 32 = 26.$$

Таким образом, гамма имеет вид: (8, 11,26).

Используя таблицу Вижинера, представим сообщение в виде чисел: М – 12, А – 0, Й – 9.

Шифрование выполним путем побитного сложения по mod 2 открытого текста и гаммы, представленных в двоичном виде. Причем, каждое десятичное число представляем пятиразрядным двоичным числом.

Открытый текст в двоичном виде	0	1	1	0	0	0	0	0	0	0	0	1	0	0	1
Гамма шифра в двоичном виде	0	1	0	0	0	0	1	0	1	1	1	1	0	1	0
Шифротекст в двоичном виде	0	0	1	0	0	0	1	0	1	1	1	0	0	1	1

Далее шифротекст представляется в виде последовательности десятичных чисел – (4, 11, 19), а затем в текстовом виде – ДЛУ.

Таким образом, сообщение МАЙ представляется в виде шифротекста «ДЛУ».

Шифрование с помощью датчика ПСП является довольно распространенным криптографическим методом. Во многом качество шифра, построенного на основе датчика ПСП, определяется не только и не столько характеристиками датчика, сколько алгоритмом получения гаммы. Один из фундаментальных принципов криптологической практики гласит: даже сложные шифры могут быть очень чувствительны к простым воздействиям.

Вопросы и задания для самопроверки

1. Какими способами можно обеспечить защиту передаваемой информации от несанкционированного доступа?
2. В чем различния между криптографией и криптоанализом?
3. Что называют шифрованием (расшифрованием)?
4. Что такое цифровая подпись?
5. Что такое криптостойкость?
6. Какую задачу решает имитостойкость?
7. Какие существуют модели каналов с криптографическим кодированием информации?
8. Какие требования предъявляются к современным криптосистемам?
9. Что является ключом в шифрующих таблицах?
10. Что является ключом в шифрующих таблицах при двойной перестановке?
11. В чем различия между аддитивной и аффинной системой Цезаря?
12. Какой основной недостаток систем шифрования Цезаря?
13. К какому типу шифров относится система шифрования Вижинера?
14. Какие достоинства имеет система Вижинера по отношению к системам Цезаря?
15. Какая криптосистема обеспечивает максимальную защиту информации?
16. Почему для большинства современных задач использование одноразовых шифров нерационально?
17. Что такое гамма шифра?
18. Чем ограничивается длина гаммы шифра?
19. На основе какого выражения можно сформировать псевдослучайную последовательность чисел?
20. В чем принципиальная разница между шифрованием методом гаммирования и одноразовой системой шифрования?
21. Результат зашифровывания сообщения КОМПЬЮТЕР методом простой перестановки (размер таблицы 3×3) выглядит:
    а) КПТОЬЕМЮР;
    б) КТПОЕЬМРЮ;
    в) ПКТЬОЕЮМР;
    г) МЮРКПТОЬЕ.
22. Зашифруйте методом Цезаря (K = 7) следующие сообщения:
    а) ОРГАНИЗАЦИЯ ДОРОЖНОГО ДВИЖЕНИЯ;
    б) КАСКАД УСИЛЕНИЯ;
    в) ДЫРОЧНАЯ ПРОВОДИМОСТЬ;
    г) ГАЛЬВАНИЧЕСКИЙ ЭЛЕМЕНТ.
23. Зашифруйте аффинной системой подстановок Цезаря (m = 32, a = 13, b = 7) следующие сообщения:
    а) ПЕРЕХОДНАЯ ФУНКЦИЯ;
    б) ТЕЛЕВИЗИОННЫЙ СТАНДАРТ;
    в) РАМОЧНАЯ АНТЕННА;
    г) ГИЛЬБЕРТОВО ПРОСТРАНСТВО.
24. Зашифруйте методом Вижинера (ключ: ГИЛЬЗА) следующие сообщения:
    а) ТЕОРИЯ ВЕРОЯТНОСТЕЙ;
    б) БАКТЕРИЦИДНАЯ ЛАМПА;
    в) ДАЛЬНЯЯ СВЯЗЬ;
    г) КВАРЦЕВЫЙ РЕЗОНАТОР.

Модуль 4

СОВРЕМЕННЫЕ СИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

Цель модуля – изучение слушателями курсов переподготовки симметричных криптосистем DES и ГОСТ 28147-89.

В результате изучения модуля слушатели переподготовки должны

знать:

- процедуры зашифровывания и расшифровывания информации симметричными алгоритмами DES и ГОСТ 28147-89;
- достоинства и недостатки симметричных криптосистем;

уметь

- использовать криптосистем DES и ГОСТ 28147-89 для шифрования информации;

иметь представление

- о размере ключа для надежного шифрования информации на современном уровне развития техники.

4.1. Американский стандарт шифрования DES

Стандарт шифрования данных DES (Data Encryption Standard), который ANSI (American National Standards Institute) называет алгоритмом шифрования данных DEA (Data Encryption Algorithm), а ISO – DEA-1, за 20 лет стал мировым стандартом [6].

4.1.1. История разработки стандарта

В начале 70-х годов XX века невоенные криптографические исследования были крайне редки. В этой области почти не публиковалось исследовательских работ. Большинство людей знали, что для своих коммуникаций военные используют специальную аппаратуру кодирования, но мало кто разбирался в криптографии как в науке. Заметными знаниями обладало Агентство национальной безопасности (National Security Agency, NSA), но оно даже не признавало публично своего собственного существования [6].

В 1972 году Национальное бюро стандартов (National Bureau of Standards, NBS), теперь называющееся Национальным институтом стандартов и техники (National Institute of Standards and Technology, NIST), выступило инициатором программы защиты линий связи и компьютерных данных. Одной из целей этой программы была разработка единого, стандартного криптографического алгоритма. Этот алгоритм мог бы быть про-

верен и сертифицирован, а использующие его различные криптографические устройства должны взаимодействовать; к тому же, был относительно недорогим и легкодоступным.

15 мая 1973 года в Federal Register NBS опубликовало требования к криптографическому алгоритму, который мог бы быть принят в качестве стандарта. В IBM существовала целая команда криптографов, работавшая в Кингстоне (Kingston) и Йорктаун Хайте (Yorktown Heights), в которую входили Рой Адлер (Roy Adler), Дон Копперсмит (Don Coppersmith), Хорст Фейстель (Horst Feistel), Эдна Кроссман (Edna Crossman), Алан Конхейм (Alan Konheim), Карл Майер (Carl Meyer), Билл Ноц (Bill Notz), Линн Смит (Lynn Smith), Уолт Тачмен (Walt Tuchman) и Брайант Такерман (Bryant Tuckerman).

Наконец, 17 марта 1975 года в Federal Register NBS опубликовало и подробности алгоритма, и заявление IBM о предоставлении неисключительной, бесплатной лицензии на алгоритм, а также предложило присылать комментарии по поводу данного алгоритма. В другой заметке в Federal Register, 1 августа 1975 года, различным организациям и широкой публике снова предлагалось прокомментировать предложенный алгоритм. Многие настороженно относились к участию «невидимой руки» NSA в разработке алгоритма. Боялись, что NSA изменит алгоритм, вставив в него потайную дверцу. Жаловались, что NSA уменьшило длину ключей с первоначальных 128 битов до 56. Жаловались на внутренние режимы работы алгоритма. Многие соображения NSA стали ясны и понятны в начале 90-х, но в 70-х они казались таинственными и тревожными [6].

Несмотря на критику Стандарт шифрования данных DES 23 ноября 1976 года был принят в качестве федерального стандарта и разрешен к использованию на всех несекретных правительственных коммуникациях. Официальное описание стандарта, FIPS PUB 46, «Data Encryption Standard», было опубликовано 15 января 1977 года и вступило в действие шестью месяцами позже. FIPS PUB 81, «Modes of DES Operation» (Режимы работы DES), было опубликовано в 1980 году. FIPS PUB 74, «Guidelines for Implementing and Using the NBS Data Encryption Standard» (Руководство по реализации и использованию Стандарта шифрования данных NBS), появилось в 1981 году. NBS также опубликовало FIPS PUB 112, специфицируя DES для шифрования паролей, и FIPS PUB 113, специфицируя DES для проверки подлинности компьютерных данных. (FIPS обозначает Federal Information Processing Standard.)

Эти стандарты были беспрецедентными. Никогда до этого оцененный NSA алгоритм не был опубликован. Возможно, эта публикация была следствием непонимания, возникшего между NSA и NBS. NSA считало, что DES будет реализовываться только аппаратно. В стандарте требова-

лась именно аппаратная реализация, но NBS опубликовало достаточно информации, чтобы можно было создать и программную реализацию DES. Не для печати NSA охарактеризовало DES как одну из своих самых больших ошибок. Если бы Агентство предполагало, что раскрытые детали позволят писать программное обеспечение, оно никогда бы не согласилось на это. Для оживления криптоанализа DES сделал больше, чем что-либо другое. Теперь для исследования был доступен алгоритм, который NSA объявило безопасным. Неслучайно следующий правительственный стандарт алгоритма, Skipjack, был засекречен.

Американский национальный институт стандартов (American National Standards Institute, ANSI) одобрил DES в качестве стандарта для частного сектора в 1981 году (ANSI X3.92.), назвав его Алгоритмом шифрования данных (Data Encryption Algorithm, DEA). ANSI опубликовал стандарт режимов работы DEA (ANSI X3.106), похожий на документ NBS, и стандарт для шифрования в сети, использующий DES (ANSI X3.105).

ISO сначала проголосовала за введение DES, называемого в ее интерпретации DEA-1, в качестве международного стандарта, а затем приняла решение не заниматься стандартизацией криптографии. Однако в 1987 году группа ISO, занимающаяся международными стандартами в области оптовой торговли, применила DES в международном стандарте проверки подлинности и для управления ключами. DES также используется в качестве австралийского банковского стандарта.

В стандарте DES было оговорено, что он будет пересматриваться каждые пять лет. В 1983 DES был повторно сертифицирован без всяких проблем. 6 марта 1987 года в Federal Register NBS попросило прокомментировать предложение на следующие пять лет. NBS предложило на обсуждение следующие три альтернативы: вновь подтвердить стандарт на следующие пять лет, отказаться от стандарта или пересмотреть применимость стандарта. После длительных споров DES был вновь утвержден в качестве правительственного стандарта США до 1992 года. В 1992 году альтернативы алгоритму DES все еще не было и его утвердили еще на 5 лет.

4.1.2. Обобщенная схема зашифровывания

Алгоритм DES использует комбинацию подстановок и перестановок. DES осуществляет зашифровывание 64-битовых блоков данных с помощью 64-битого ключа, в котором значащими являются 56 бит (остальные 8 бит - проверочные биты для контроля на четность) [5, 6]. Расшифровывание в DES является операцией, обратной шифрованию, и выполняется путем повторения операций зашифровывания в обратной последовательности.

Обобщенная схема процесса шифрования в алгоритме DES (рис. 4.1) заключается в начальной перестановке бит 64-битого блока, шестнадцати циклах шифрования и, наконец, вконечной перестановке бит.

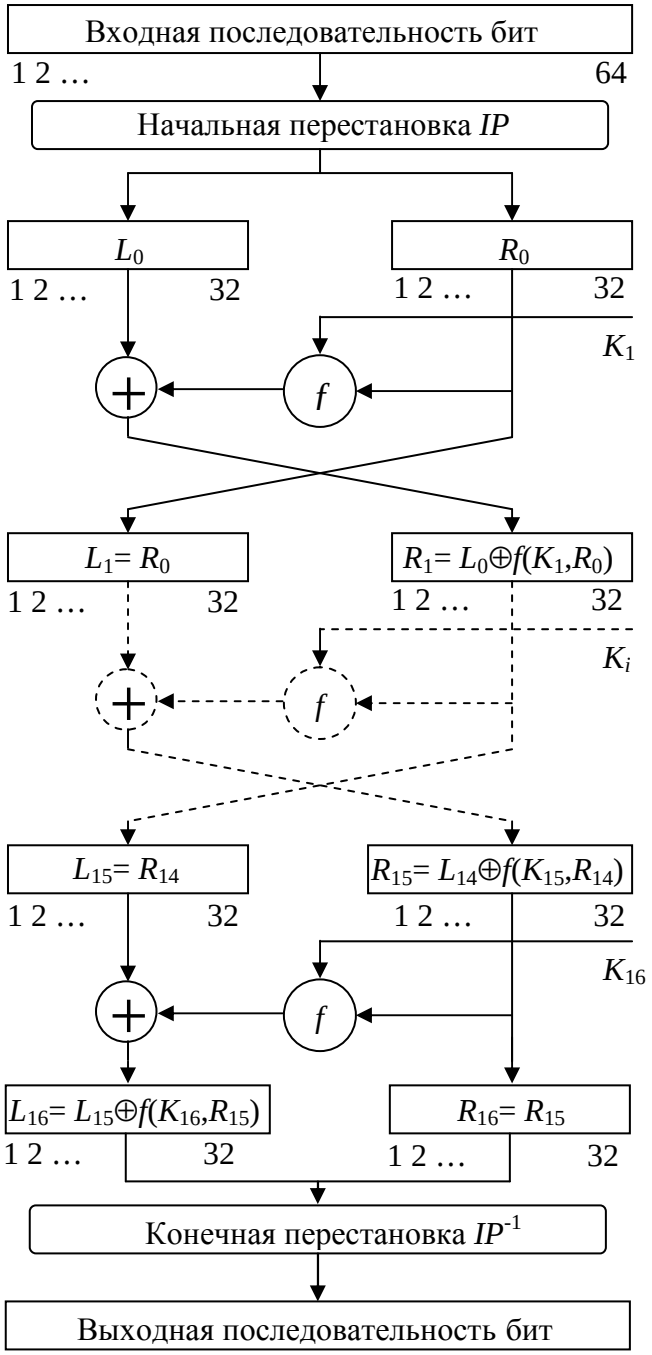

Рис. 4.1. **Обобщенная схема зашифровывания в алгоритме DES**

Следует 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 взлома шифра.

Пусть из файла исходного текста считан 64-битовый блок T_0. Он преобразуется с помощью матрицы начальной перестановки *IP* (табл. 4.1).

Биты входного блока T_0 (64 бита) переставляются в соответствии с матрицей IP: бит 58 входного блока T_0 становится битом 1, бит 50 – битом 2 и т.д. Эту перестановку можно описать выражением $T_0 = IP(T)$. Полученная последовательность бит T_0 разделяется на две последовательности: L_0 – левые, или старшие, биты, R_0 – правые, или младшие, биты – каждая из которых содержит 32 бита.

Таблица 4.1

Матрица начальной перестановки

Начальная перестановка *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

Затем выполняется итеративный процесс шифрования, состоящий из 16 циклов. Пусть $T_i -$ результат i-й итерации: $T_i = L_i R_i$, где $L_i = t_1 t_2 \ldots t_{32}$ (первые 32 бита); $R_i = t_{33} t_{34} \ldots t_{64}$ (последние 32 бита). Тогда результат i-й итерации описывается следующими формулами:

$$L_i = R_{i-1}, i = 1,\ 2,\ ...,\ 16;$$

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

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-битого ключа шифра *K*. (Подробнее функция шифрования f и алгоритм получения ключа *K* описаны ниже.)

На последнем шаге итерации получают последовательности R_{16} и L_{16} (без перестановки местами), которые конкатенируются в 64-битую последовательность $R_{16}L_{16}$.

По окончании шифрования осуществляется восстановление позиций бит с помощью матрицы обратной перестановки IP^{-1} (табл. 4.2).

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

Итеративный процесс расшифровывания может быть описан следующими формулами:

$$R_i = L_{i-1}, i = 1,\ 2,\ ...,\ 16;$$

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

Таблица 4.2

Матрица обратной перестановки

Обратная перестановка 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

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

4.1.3. Функция шифрования в DES

Схема вычисления функции шифрования $f\left(R_{i-1}, K_{\text{i}}\right)$ показана на рис. 4.2.

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 бит в 32-битовой последовательности).

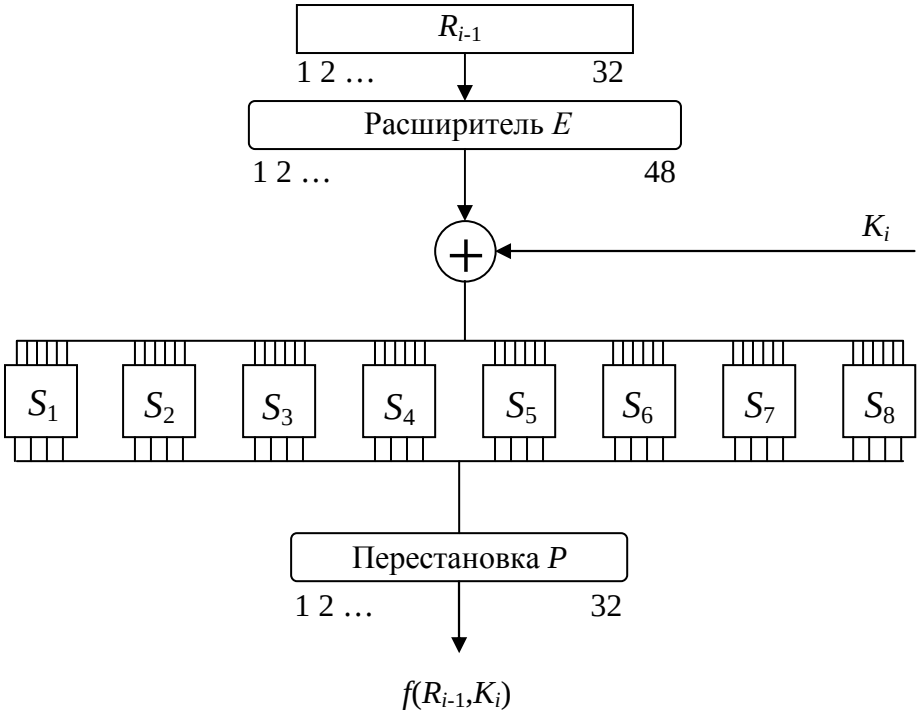

Рис. 4.2. **Схема вычисления функции шифрования** f

Приведем определения этих функций.

Аргументами функции шифрования f являются R_{i-1} (32 бита) и K_i (48 бит). Результат функции $E(R_{i-1})$ есть 48-битовое число. Функция расширения E, выполняющая расширение 32 бит до 48 (принимает блок из 32 и порождает блок из 48 бит), определяется табл. 4.3.

Таблица 4.3

Функция *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 табл. 4.3 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 2 с текущим значением ключа K_i и затем разбивается на восемь 6-битовых блоков $B_1, B_2, \ldots, B_8 = E(R_{i-1}) \oplus K_i$.

Далее каждый из этих блоков используется как номер элемента в функциях - матрицах $S_1, S_2, \ldots, S_8$, содержащих 4-битовые значения (табл. 4.4).

Таблица 4.4

Функция *S*

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

Следует отметить, что выбор элемента в матрице S осуществляется достаточно оригинальным образом. Пусть на вход матрицы S поступает 6-битовый блок $B_j = b_1 b_2 b_3 b_4 b_5 b_6$, тогда 2-битовое число $b_1 b_6$ указывает номер

строки матрицы, а 4-битовое число $b_2b_3b_4b_5$ - номер столбца. Например, если на вход матрицы S_1 поступает 6-битовый блок $B_1 = b_1b_2b_3b_4b_5b_6 = 100110_{(2)}$, то 2-битовое число $b_1b_6 = 10_{(2)} = 2_{(2)}$ указывает строку с номером 2 матрицы S_1, а 4-битовое число $b_2b_3b_4b_5 = 0011_{(2)} = 3_{(10)}$ – 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 $B_1, B_2, \ldots, B_8$ обеспечивает выбор 4-битового элемента в каждой из матриц $B_1, B_2, \ldots, B_8$.

В результате получаем $S_1(B_1), S_2(B_1), \ldots, S_8(B_1)$, т.е. 32-битовый блок (поскольку матрицы S, содержат 4-битовые элементы). Этот 32-битовый блок преобразуется с помощью функции перестановки бит P (табл. 4.5).

Таблица 4.5

Функция *P*

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

Таким образом, функция шифрования

$$f(R_{i-1}, K_i) = P(S_1(B_1), \ldots, S_8(B_1)).$$

4.1.4. Алгоритм вычисления ключей

Как нетрудно заметить, на каждой итерации используется новое значение ключа K_i (длиной 48 бит). Новое значение ключа K_i вычисляется из начального ключа K (рис. 4.3).

Ключ K представляет собой 64-битовый блок с 8 битами контроля по четности, расположенными в позициях 8, 16, 24, 32, 40, 48, 56, 64. Для удаления контрольных бит и подготовки ключа к работе используется функция G первоначальной подготовки ключа (табл. 4.6).

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 (первым битом C_0 будет бит 57 ключа шифра, затем бит 49 и т.д., а последними битами – биты 44 и 36 ключа).

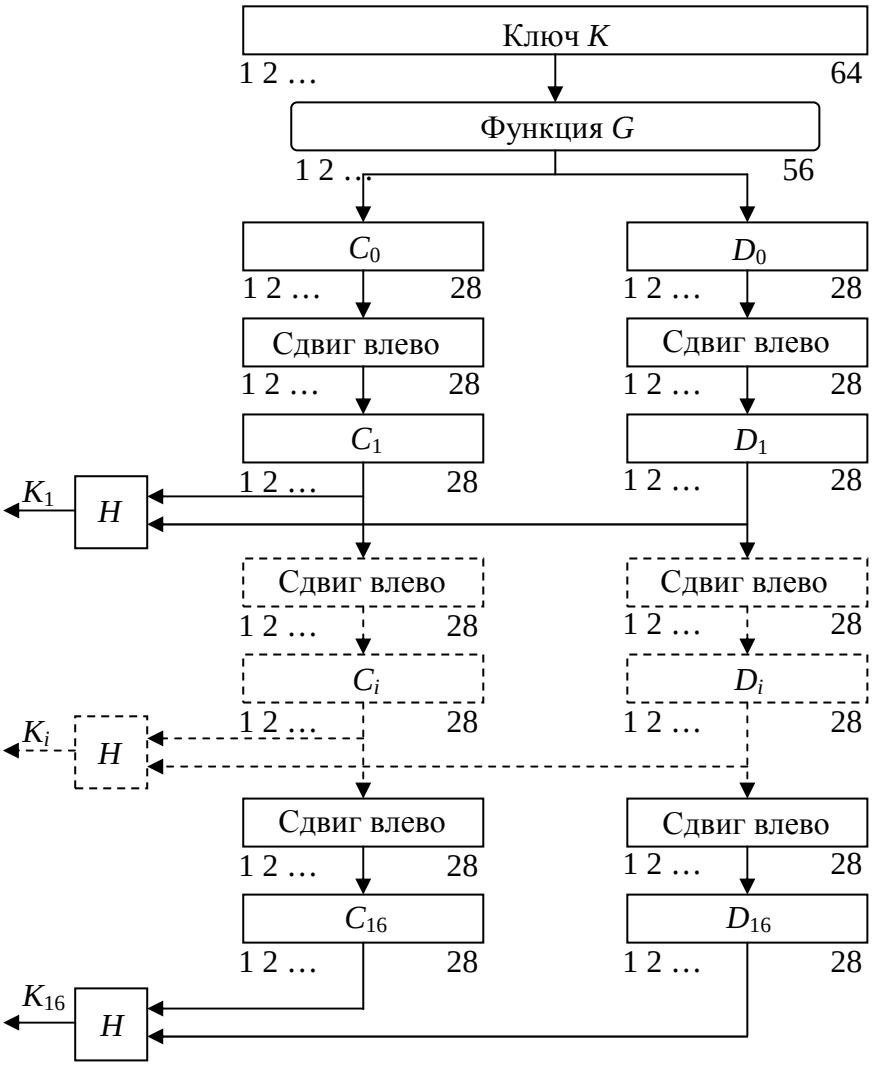

Рис. 4.3. **Схема алгоритма вычисления ключей K_i**

Таблица 4.6

Функция *G*

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

Следующие четыре строки матрицы G определяют, как выбираются биты последовательности D_0 (т.е. последовательность D_0 будет состоять из бит 63, 55, 47,..., 12, 4 ключа шифра).

Как видно из табл., 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, $i = 1,\ 2,\ \ldots,\ 16$. Для этого применяются операции циклического сдвига влево на один или два бита в зависимости от номера шага итерации (табл. 4.7).

Таблица 4.7

Таблица сдвигов для вычисления ключа

Итерация	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Сдвиг влево	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	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 бит из 56-битовой последовательности C_iD_i и их перестановки. Другими словами, ключ $K_i = H\left(C_iD_i\right)$, где функция H определяется матрицей, завершающей обработку ключа (табл. 4.8). Согласно табл., первым битом ключа K_i будет 14-й бит последовательности C_iD_i, вторым – 17-й, 47-м – 29-й, а 48м – 32-й бит C_iD_i.

Таблица 4.6

Функция *H*

14	17	11	24	1	5
3	28	15	6	21	10
23	19	22	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

4.1.5. Реальные критерии проектирования DES

После появления публикаций о дифференциальном криптоанализе IBM раскрыла критерии проектирования *S*-блоков и *P*-блока. Критериями проектирования *S*-блоков являлись [6]:

- У каждого *S*-блока 6 входных битов и 4 выходных бита. (Это самый большой размер, который мог быть реализован в одной микросхеме по технологии 1974 года.)
- Ни один выходной бит *S*-блока не должен быть слишком близок к линейной функции входных битов.
- Если зафиксировать крайние левый и правый биты *S*-блока, изменяя 4 средних бита, то каждый возможный 4-битовый результат получается только один раз.
- Если два входа *S*-блока отличаются только одним битом, результаты должны отличаться по крайней мере на 2 бита.
- Если два входа *S*-блока отличаются только двумя центральными битами, результаты должны отличаться по крайней мере на 2 бита.
- Если два входа *S*-блока отличаются двумя первыми битами, а последние их последние 2 бита совпадают, результаты не должны быть одинаковыми.
- Для любого ненулевого 6-битового отличия между входами не более, чем 8 из 32 пар входов могут приводить на выходе к одинаковому различию.
- Аналогичный предыдущему критерий, но для случая трех активных *S*-блоков.

Критериями проектирования *P*-блока являлись:

- 4 выходных бита каждого *S*-блока на этапе i распределены так, чтобы 2 из них влияют на средние биты *S*-блоков на этапе $i + 1$, а другие 2 бита влияют на последние биты.
- 4 выходных бита каждого *S*-блока влияют на шесть различных *S*-блоков, никакие 2 не влияют на один и тот же *S*-блок.
- Если выходной бит одного *S*-блока влияет на средние биты другого *S*-блока, то выходной бит этого другого *S*-блока не может влиять на средние биты первого *S*-блока.
- Сегодня совсем нетрудно генерировать *S*-блоки, но в начале 70-х годов это было нелегкой задачей. Программы, готовившие *S*-блоки, работали месяцами.

4.1.6. Основные режимы работы алгоритма DES. Комбинирование блочных алгоритмов

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 CPB (Cipher FeedBack);
- обратная связь по выходу OFB (Output FeedBack).

Режим Электронная кодовая книга

Длинный файл разбивают на 64-битые отрезки (блоки) по 8 байт. Каждый из этих блоков шифруют независимо с использованием одного и того же ключа шифрования (рис. 4.4).

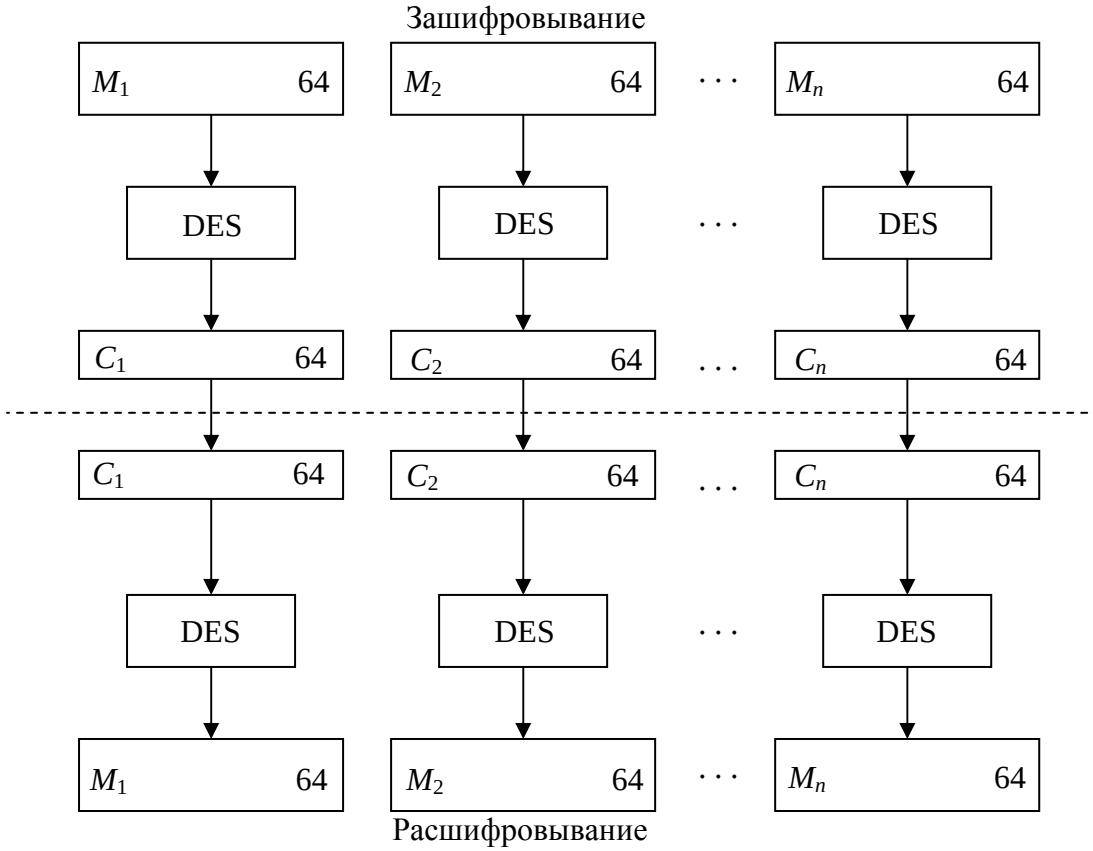

Рис. 4.4. **Схема алгоритма DES в режиме электронной кодовой книги**

Основное достоинство – простота реализации. Недостаток – относительно слабая устойчивость против квалифицированных криптоаналитиков. Из-за фиксированного характера шифрования при ограниченной длине блока 64 бита возможно проведение криптоанализа «со словарем». Блок такого размера может повториться в сообщении вследствие большой избы-

точности 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-битые 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 (рис. 4.5). 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.

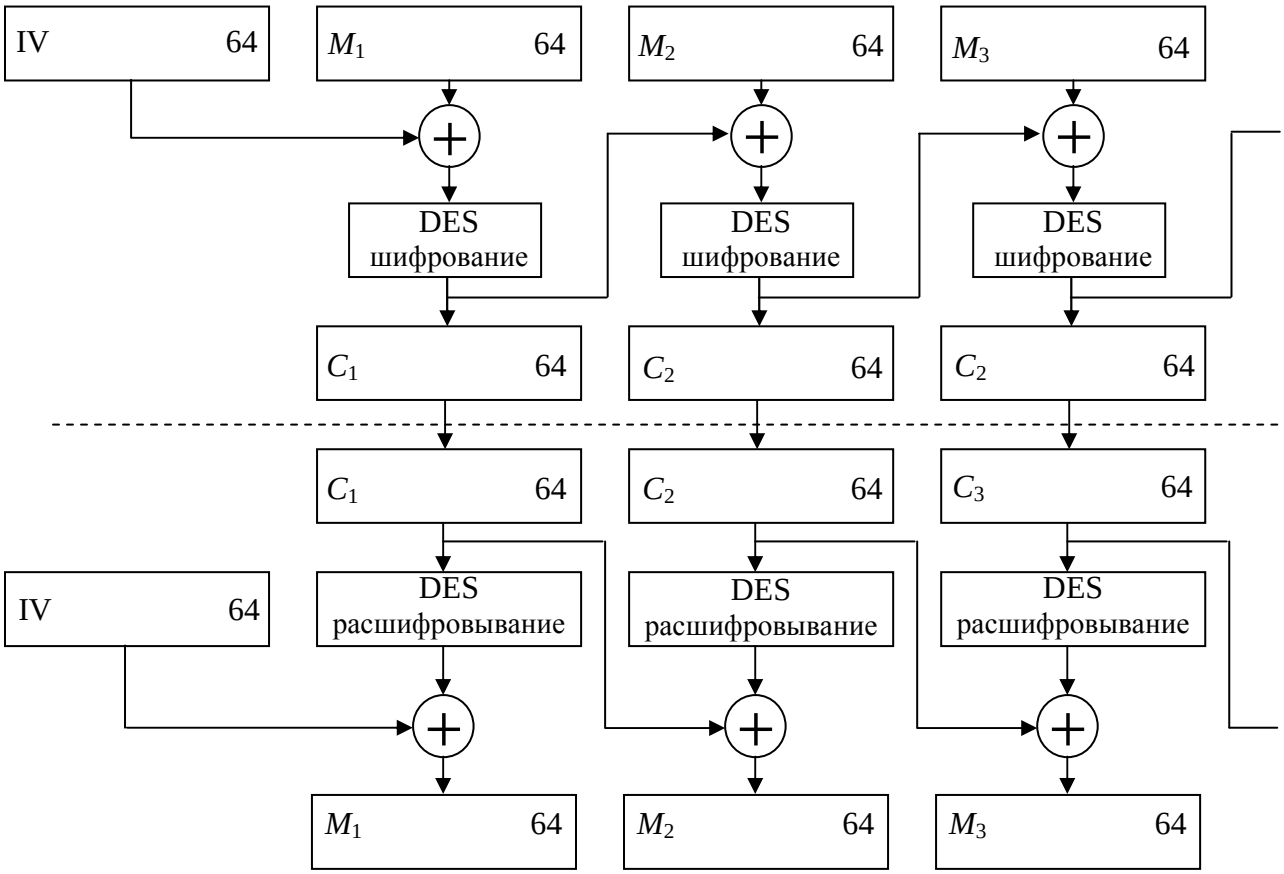

Рис. 4.5. **Схема алгоритма DES в режиме сцепления блоков шифра**

Очевидно, что последний 64-битовый блок шифротекста является функцией секретного ключа, начального вектора и каждого бита открыто-

го 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.

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 шифротексту

В этом режиме размер блока может отличаться от 64 бит (рис 4.6). Файл, подлежащий шифрованию (расшифровыванию), считывается последовательными блоками длиной k бит ($k = 1\ldots64$).

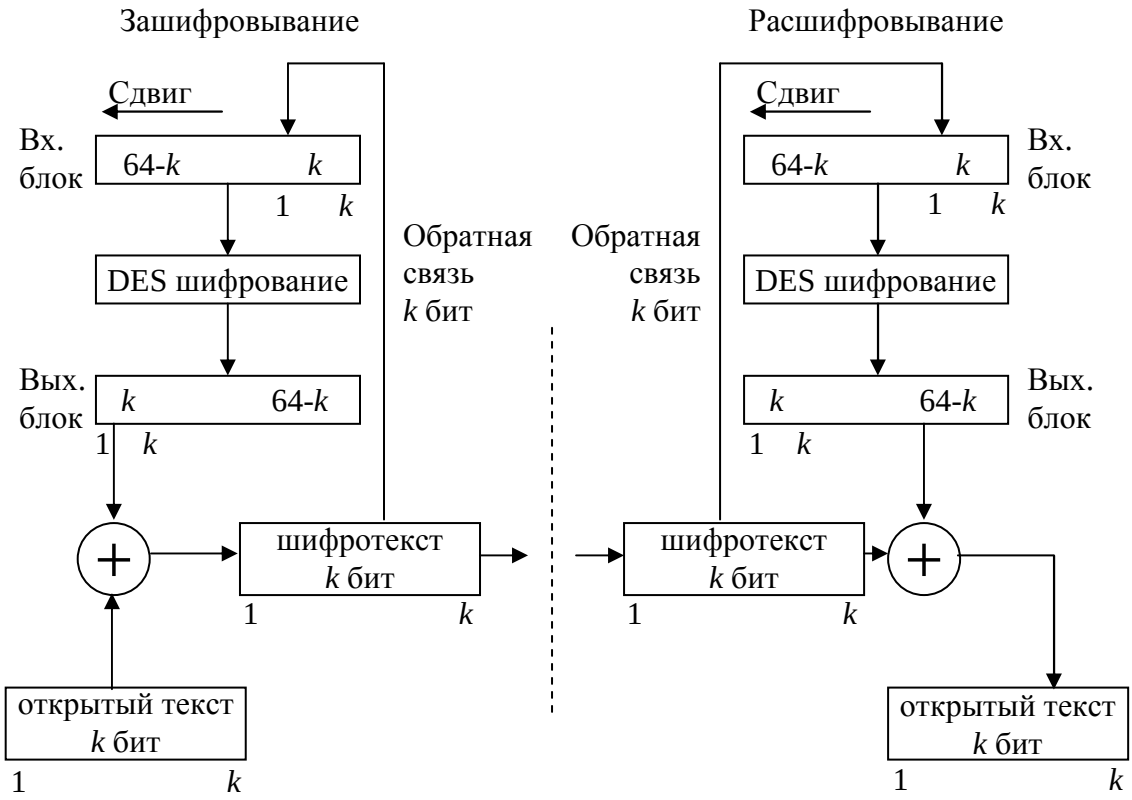

Рис. 4.6. **Схема алгоритма DES в режиме обратной связи по шифротексту**

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 бит 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 бит предыдущего зашифрованного блока.

Обновление сдвигового регистра осуществляется путем удаления его старших k бит и записи C_i в регистр. Восстановление зашифрованных данных также выполняется относительно просто: P_{i-1} и C_i вычисляются аналогичным образом и $M_i = C_i \oplus P_{i-1}$.

Режим Обратная связь по выходу

Этот режим тоже использует переменный размер блока и сдвиговый регистр, инициализируемый так же, как в режиме СРВ, а именно - входной блок вначале содержит вектор инициализации IV, выровненный по правому краю (рис. 4.7). При этом для каждого сеанса шифрования данных необходимо использовать новое начальное состояние регистра, которое должно пересылаться по каналу открытым текстом.

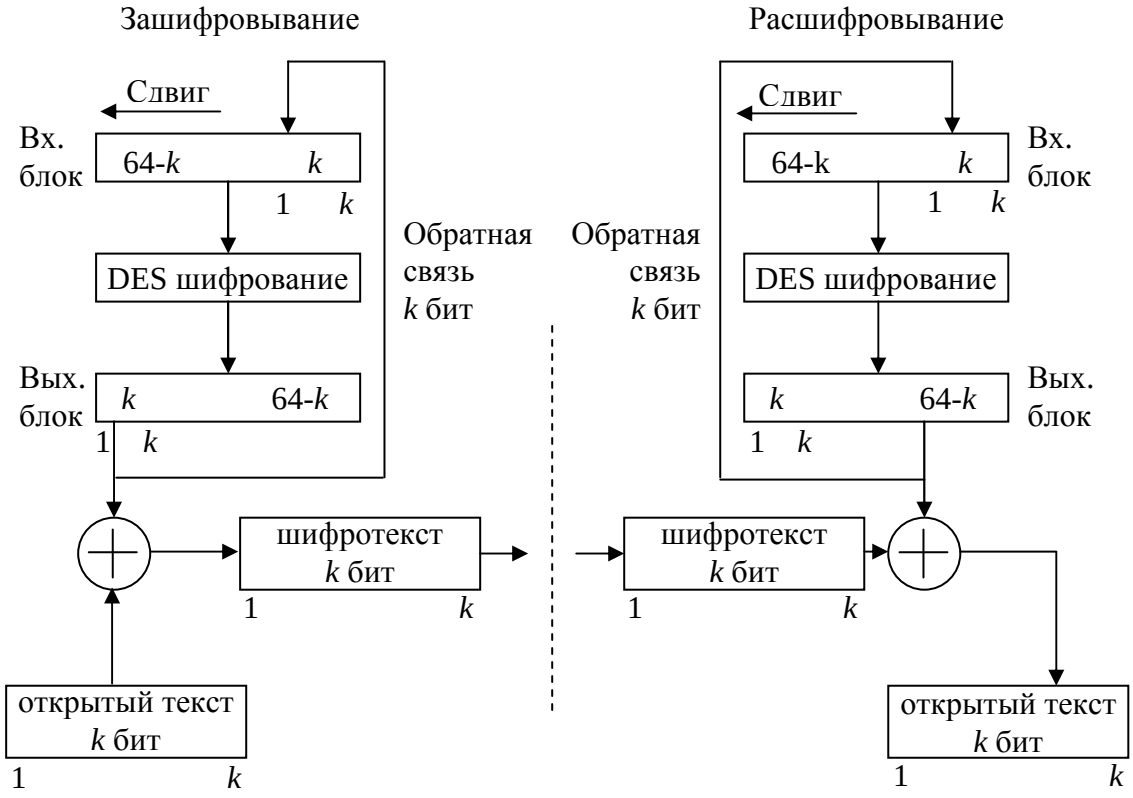

Рис. 4.7. **Схема алгоритма DES в режиме обратной связи по выходу**

Положим $M = M_1 M_2 ... M_n$ для всех $i = 1 \ldots n$ $C_i = M_i \oplus P_i$, где P_i - старшие k бит операции $\mathrm{DES}(C_{i-1})$. Отличие от режима обратной связи по шифротексту состоит в методе обновления сдвигового регистра.

Это осуществляется путем отбрасывания старших k бит и дописывания справа P_i.

4.2. Стандарт шифрования ГОСТ 28147-89

4.2.1. Введение

В качестве официального алгоритма криптографического преобразования данных для систем обработки информации в Республике Беларусь выбран алгоритм, стандартизованный в ГОСТ 28147-89. Он предназначен для аппаратной и программной реализации, удовлетворяет криптографическим требованиям и не накладывает ограничений на степень секретности защищаемой информации. Стандарт закреплен ГОСТом №28147-89, принятым в 1989 году в СССР.

Элементы данных при рассмотрении данного алгоритма обозначаются заглавными латинскими буквами с наклонным начертанием (например, X). Через $|X|$ обозначается размер элемента данных X в битах. Таким образом, если интерпретировать элемент данных X как целое неотрицательное число, можно записать следующее неравенство: $0 \le X < 2^{|X|}$.

Если элемент данных состоит из нескольких элементов меньшего размера, то обозначается следующим образом:

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

Процедура объединения нескольких элементов данных в один называется конкатенацией данных и обозначается символом «||». Естественно, для размеров элементов данных должно выполняться следующее соотношение:

$$|X| = |X_0| + |X_1| + ... + |X_{n-1}|.$$

При задании сложных элементов данных и операции конкатенации составляющие элементы данных перечисляются в порядке возрастания старшинства. Иными словами, если интерпретировать составной элемент и все входящие в него элементы данных как целые числа без знака, то можно записать следующее равенство:

$$(X_0, X_1, ..., X_{n-1}) = X_0 \,||\, X_1 \,||\, ... \,||\, X_{n-1} =$$
$$= X_0 + 2^{|X_0|}(X_1 + 2^{|X_1|}(...(X_{n-1} + 2^{|X_{n-1}|} X_{n-1})...)).$$

Если внимательно изучить оригинал ГОСТ 28147-89, можно заметить, что в нем содержится описание алгоритмов нескольких уровней. На самом верхнем находятся практические алгоритмы, предназначенные для шифрования массивов данных и выработки для них имитовставки. Все они опираются на три алгоритма низшего уровня, называемые в тексте ГОСТа циклами. Эти фундаментальные алгоритмы можно назвать как базовые циклы, чтобы отличать их от всех прочих циклов. Они имеют следующие названия и обозначения, последние приведены в скобках:

- цикл зашифрования ($32-3$);
- цикл расшифрования ($32-P$);
- цикл выработки имитовставки ($16-3$).

В свою очередь, каждый из базовых циклов представляет собой многократное повторение одной процедуры, называемой основным шагом криптопреобразования.

Таким образом, надо понять три следующие вещи:

- что такое основной шаг криптопреобразования;
- как из основных шагов складываются базовые циклы;
- как из трех базовых циклов складываются все практические алгоритмы ГОСТа.

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

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

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

4.2.2. Основной шаг криптопреобразования

Основной шаг криптопреобразования по своей сути является оператором, определяющим преобразование 64-битового блока данных. Допол-

нительным параметром этого оператора является 32-битовый блок, в качестве которого используется какой-либо элемент ключа. Схема алгоритма основного шага приведена на рис. 4.8. Ниже даны пояснения к алгоритму основного шага:

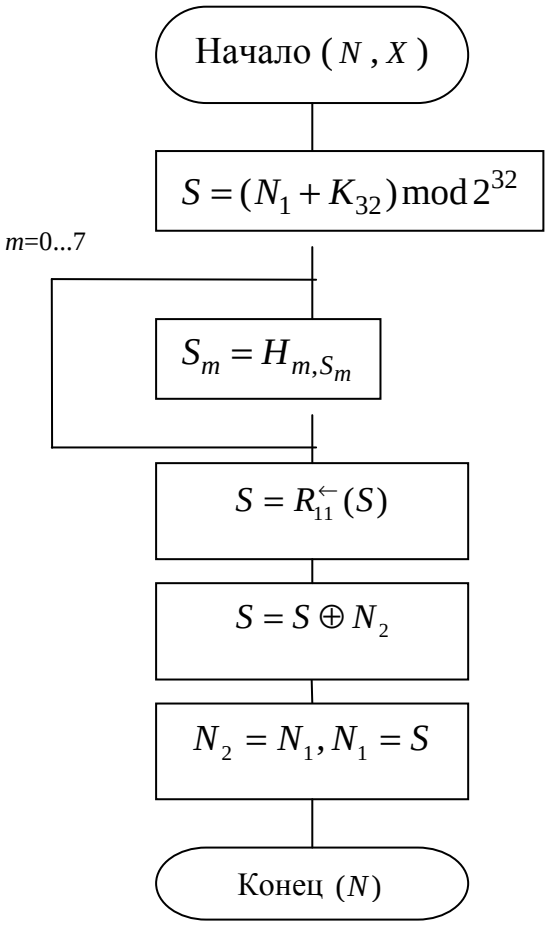

Рис. 4.8. **Схема основного шага криптопреобразования алгоритма ГОСТ 28147-89**

Шаг 0

Определяет исходные данные для основного шага криптопреобразования:

- N -преобразуемый 64-битовый блок данных, в ходе выполнения шага его младшая (N_1) и старшая (N_2) части обрабатываются как отдельные 32-битовые целые числа без знака. Таким образом, можно записать

$$N = N_1, N_2.$$

- X – 32-битовый элемент ключа.

Шаг 1

Сложение с ключом. Младшая половина преобразуемого блока складывается по модулю 2^{32} с используемым на шаге элементом ключа, результат передается на следующий шаг;

Шаг 2

Поблочная замена. 32-битовое значение, полученное на предыдущем шаге, интерпретируется как массив из восьми 4-битовых блоков кода: $S = (S_0, S_1, S_2, S_3, S_4, S_5, S_6, S_7)$.

Далее значение каждого из восьми блоков заменяется новым, которое выбирается по таблице замен следующим образом: значение блока S_i меняется на S_i-й по порядку элемент (нумерация с нуля) i-го узла замен (т.е. i-й строки таблицы замен, нумерация также с нуля). Другими словами, в качестве замены для значения блока выбирается элемент из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неот-

рицательного числа. Теперь становится понятным размер таблицы замен: число строк в ней равно числу 4-битовых элементов в 32-битовом блоке данных, т.е. восьми, а число столбцов – числу различных значений 4-битового блока данных, равному, как известно, 24, 16.

Шаг 3

Циклический сдвиг на 11 бит влево. Результат предыдущего шага сдвигается циклически на 11 бит в сторону старших разрядов и передается на следующий шаг. На схеме алгоритма символом $R_{11}^{\leftarrow}$ обозначена функция циклического сдвига своего аргумента на 11 бит влево, т.е. в сторону старших разрядов.

Шаг 4

Побитовое сложение: значение, полученное на шаге 3, побитно складывается по модулю 2 со старшей половиной преобразуемого блока.

Шаг 5

Сдвиг по цепочке: младшая часть преобразуемого блока сдвигается на место старшей, а на ее место помещается результат выполнения предыдущего шага.

Шаг 6

Полученное значение преобразуемого блока возвращается как результат выполнения алгоритма основного шага криптопреобразования.

4.2.3. Базовые циклы криптографических преобразований

ГОСТ относится к классу блочных шифров, т.е. единицей обработки информации в нем является блок данных. Следовательно, в нем определены алгоритмы для криптографических преобразований, т.е. для зашифрования, расшифрования и учета в контрольной комбинации одного блока данных. Именно эти алгоритмы и называются базовыми циклами ГОСТа, что подчеркивает их фундаментальное значение для построения этого шифра.

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

Базовые циклы заключаются в многократном выполнении основного шага с использованием разных элементов ключа и отличаются друг от друга только числом повторения шага и порядком использования ключевых элементов. Ниже приведен этот порядок для различных циклов.

1. Цикл зашифрования 32-*З*:

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

2. Цикл расшифрования 32-*Р*:

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

3. Цикл выработки имитовставки 16-*З*:

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

Каждый из циклов имеет собственное буквенно-цифровое обозначение, соответствующее шаблону $n-X$, где первый элемент обозначения (n) задает число повторений основного шага в цикле, а второй элемент обозначения (X) – порядок зашифрования $(З)$ или расшифрования $(Р)$ в использовании ключевых элементов.

Цикл расшифрования должен быть обратным циклу зашифрования, т.е. последовательное применение этих двух циклов к произвольному блоку должно дать в итоге исходный блок, что отражается следующим соотношением: $Ц_{32-Р}(Ц_{32-З}(Т)) = Т$, где $Т$ – произвольный 64-битовый блок данных, $Ц_X(Т)$ – результат выполнения цикла X над блоком данных $Т$. Для выполнения этого условия для алгоритмов, подобных ГОСТу, необходимо и достаточно, чтобы порядок использования ключевых элементов соответствующими циклами был взаимно обратным. В справедливости записанного условия для рассматриваемого случая легко убедиться, сравнив приведенные выше последовательности для циклов 32-*З* и 32-*Р*. Из двух взаимно обратных циклов любой может быть использован для зашифрования, тогда второй должен быть использован для расшифрования данных, однако стандарт ГОСТ 28147-89 закрепляет роли за циклами и не предоставляет пользователю права выбора в этом вопросе.

Цикл выработки имитовставки вдвое короче циклов шифрования, порядок использования ключевых элементов в нем такой же, как в первых 16 шагах цикла зашифрования, поэтому этот порядок в обозначении цикла кодируется той же самой буквой *З*.

Каждый из них принимает в качестве аргумента и возвращает в качестве результата 64-битовый блок данных, обозначенный на схемах *N*. Символ Шаг(*N*, *X*) обозначает выполнение основного шага криптопреобразования для блока *N* с использованием ключевого элемента *X*. Между циклами шифрования и вычисления имитовставки есть еще одно отличие: в конце базовых циклов шифрования старшая и младшая часть блока результата меняются местами, это необходимо для их взаимной обратимости.

4.2.4. Основные режимы шифрования

ГОСТ 28147-89 предусматривает три следующих режима шифрования данных:

- простая замена;
- гаммирование;
- гаммирование с обратной связью

и один дополнительный режим выработки имитовставки.

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

Обозначения, используемые на схемах:

$T_o, T_{Ш}$ – массивы соответственно открытых и зашифрованных данных;

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

n – число 64-битовых блоков в массиве данных;

$Ц_X$ – функция преобразования 64-битового блока данных по алгоритму базового цикла X.

Простая замена

Зашифрование в данном режиме заключается в применении цикла 32-*З* к блокам открытых данных, расшифрование – цикла 32-*Р* к блокам зашифрованных данных. Это наиболее простой из режимов, 64-битовые блоки данных обрабатываются в нем независимо друг от друга.

Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифрованию или расшифрованию, должен быть

кратен 64 битам: $|T_o| = |T_{Ш}| = 64 \cdot n$, после выполнения операции размер полученного массива данных не изменяется.

Режим шифрования простой заменой имеет следующие особенности:

1. Поскольку блоки данных шифруются независимо друг от друга и их позиции в массиве, при зашифровании двух одинаковых блоков открытого текста получаются одинаковые блоки шифртекста и наоборот. Отмеченное свойство позволит криптоаналитику сделать заключение о тождественности блоков исходных данных, если в массиве зашифрованных данных ему встретились идентичные блоки, что является недопустимым для серьезного шифра.

2. Если длина шифруемого массива данных не кратна 8 байтам или 64 битам, возникает проблема, чем и как дополнять последний неполный блок данных массива до полных 64 бит. Очевидные решения типа «дополнить неполный блок нулевыми битами» или «дополнить неполный блок фиксированной комбинацией нулевых и единичных битов» могут при определенных условиях дать в руки криптоаналитика возможность методами перебора определить содержимое этого самого неполного блока, и этот факт означает снижение стойкости шифра. Кроме того, длина шифртекста при этом изменится, увеличившись до ближайшего целого, кратного 64 битам, что часто бывает нежелательным.

ГОСТ предписывает использовать режим простой замены исключительно для шифрования ключевых данных.

Гаммирование

Гаммирование решает обе упомянутые проблемы; во-первых, все элементы гаммы различны для реальных шифруемых массивов и, следовательно, результат зашифрования даже двух одинаковых блоков в одном массиве данных будет различным. Во-вторых, хотя элементы гаммы и вырабатываются одинаковыми порциями в 64 бита, использоваться может и часть такого блока с размером, равным размеру шифруемого блока.

Гамма для этого режима получается следующим образом: с помощью некоторого алгоритмического рекуррентного генератора последовательности чисел (РГПЧ) вырабатываются 64-битовые блоки данных, которые далее подвергаются преобразованию по циклу 32-*З*, т.е. зашифрованию в режиме простой замены, в результате получаются блоки гаммы. Алгоритмы зашифрования и расшифрования в режиме гаммирования идентичны.

РГПЧ, используемый для выработки гаммы, является рекуррентной функцией:

$$\Omega_{i+1} = f(\Omega_i),$$

где Ω_i – элементы рекуррентной последовательности;

f – функция преобразования.

Следовательно, неизбежно возникает вопрос о его инициализации, то есть об элементе Ω_0. В действительности, этот элемент данных является параметром алгоритма для режимов гаммирования, на схемах он обозначен как S и называется в криптографии синхропосылкой, а в ГОСТе – начальным заполнением одного из регистров шифрователя. Разработчики ГОСТа решили использовать для инициализации РГПЧ не непосредственно синхропосылку, а результат ее преобразования по циклу 32-*З*: $\Omega_0 = Ц_{32-З}(S)$. Последовательность элементов, вырабатываемых РГПЧ, целиком зависит от его начального заполнения, т.е. элементы этой последовательности являются функцией своего номера и начального заполнения РГПЧ: $\Omega_i = f_i(\Omega_0)$, где $f_i(X) = f(f_{i-1}(X))$, $f_0(X) = X$. С учетом преобразования по алгоритму простой замены добавляется еще и зависимость от ключа:

$$Г_i = Ц_{32-З}(\Omega_i) = Ц_{32-З}(f_i(\Omega_0)) = Ц_{32-З}(f_i(Ц_{32-З}(S))) = \varphi_i(S,K),$$

где $Г_i$ – i-тый элемент гаммы;

K – ключ.

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

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

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

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

- в 64-битовом блоке старшая и младшая части обрабатываются независимо друг от друга:

$$\Omega_i = (\Omega_i^0, \Omega_i^1), |\Omega_i^0| = |\Omega_i^1| = 32, \Omega_{i+1}^0 = \hat{f}(\Omega_i^0), \Omega_{i+1}^1 = \tilde{f}(\Omega_i^1),$$

т.е. фактически, существуют два независимых РГПЧ – для старшей и младшей частей блока;

- рекуррентные соотношения для старшей и младшей частей следующие:

$$\Omega_{i+1}^0 = (\Omega_i^0 + C_1) \bmod 2^{32}, \text{ где } C_1 = 1010101_{16};$$

$$\Omega_{i+1}^1 = (\Omega_i^1 + C_2 - 1) \bmod (2^{32} - 1) + 1, \text{ где } C_2 = 1010104_{16}.$$

Нижний индекс в записи числа означает его систему счисления, таким образом, константы, используемые на данном шаге, записаны в 16-ричной системе счисления. Период повторения последовательности для младшей части составляет 2^{32}, для старшей части – $2^{32} - 1$, для всей последовательности – $2^{32} \cdot (2^{32} - 1)$.

Схема алгоритма шифрования в режиме гаммирования:

Шаг 0

Определяет исходные данные для основного шага криптопреобразования:

- $T_{0(Ш)}$ – массив открытых (зашифрованных) данных произвольного размера, подвергаемый процедуре зашифрования (расшифрования), по ходу процедуры массив подвергается преобразованию порциями по 64 бита;
- S – синхропосылка, 64-битовый элемент данных, необходимый для инициализации генератора гаммы.

Шаг 1

Начальное преобразование синхропосылки, выполняемое для ее «рандомизации», т.е. для устранения статистических закономерностей, присутствующих в ней, результат используется как начальное заполнение РГПЧ.

Шаг 2

Один шаг работы РГПЧ, реализующий его рекуррентный алгоритм. В ходе данного шага старшая (S_1) и младшая (S_0) части последовательности данных вырабатываются независимо друг от друга.

Шаг 3

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

Шаг 4

Результат работы алгоритма – зашифрованный (расшифрованный) массив данных.

Ниже перечислены особенности гаммирования как режима шифрования.

1. Одинаковые блоки в открытом массиве данных дадут при зашифровании различные блоки шифртекста, что позволит скрыть факт их идентичности.

2. Поскольку наложение гаммы выполняется побитно, шифрование неполного блока данных легко выполнимо как шифрование битов этого неполного блока, для чего используется соответствующие биты блока гаммы. Так, для зашифрования неполного блока в 1 бит можно использовать любой бит из блока гаммы.

3. Синхропосылка, использованная при зашифровании, каким-то образом должна быть передана для использования при расшифровании. Это может быть достигнуто следующими путями:

- хранить или передавать синхропосылку вместе с зашифрованным массивом данных, что приведет к увеличению размера массива данных при зашифровании на размер синхропосылки, то есть на 8 байт;
- использовать предопределенное значение синхропосылки или вырабатывать ее синхронно источником и приемником по определенному закону, в этом случае изменение размера передаваемого или хранимого массива данных отсутствует.

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

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

$$\overline{t_i}^{Ш} = t_i^{Ш} \oplus 1 = (t_i^{O} \oplus \gamma_1) \oplus 1 = (t_i^{o} \oplus 1) \oplus \gamma_i = \overline{t_i}^{O} \oplus \lambda_i,$$

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

Данное свойство дает злоумышленнику возможность, воздействуя на биты шифртекста, вносить предсказуемые и даже целенаправленные изменения в соответствующий открытый текст, получаемый после его расшифрования, не обладая при этом секретным ключом.

Гаммирование с обратной связью

Данный режим очень похож на режим гаммирования и отличается от него только способом выработки элементов гаммы – очередной элемент гаммы вырабатывается как результат преобразования по циклу 32-*3* предыдущего блока зашифрованных данных, а для зашифрования первого блока массива данных элемент гаммы вырабатывается как результат преобразования по тому же циклу синхропосылки. Этим достигается зацепление блоков – каждый блок шифротекста в этом режиме зависит от соответствующего и всех предыдущих блоков открытого текста. Поэтому данный режим иногда называется гаммированием с зацеплением блоков. На стойкость шифра факт зацепления блоков не оказывает никакого влияния.

Шифрование в режиме гаммирования с обратной связью обладает теми же особенностями, что и шифрование в режиме обычного гаммирования, за исключением влияния искажений шифротекста на соответст-

вующий открытый текст. Для сравнения запишем функции расшифрования блока для обоих упомянутых режимов:

$$T_i^o = T_i^{ш} \oplus Г_i \ – \text{гаммирование;}$$

$$T_i^o = T_i^{ш} \oplus Ц_{32-3}(T_{i-1}^{ш}) \ – \text{гаммирование с обратной связью.}$$

Выработка имитовставки к массиву данных

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

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

В качестве имитовставки берется часть блока, полученного на выходе, обычно 32 его младших бита.

В шифре ГОСТ используется 256-битовый ключ и объем ключевого пространства составляет 2^{256}. Ни на одной из существующих в настоящее время или предполагаемых к реализации в недалеком будущем ЭВМ общего применения нельзя подобрать ключ за время, меньшее многих сотен лет. Стандарт проектировался с большим запасом и по стойкости на много порядков превосходит американский стандарт DES с его реальным размером ключа в 56 бит и объемом ключевого пространства всего 2^{56}.

Вопросы и задания для самопроверки

1. К какому типу криптосистем относится алгоритм DES?
2. Представьте обобщенную схему зашифровывания DES.
3. Сколько различных ключей существует в алгоритме DES?
4. Сколько циклов шифрования в алгоритме DES?
5. Представьте обобщенную схему расшифровывания DES.
6. Какого размера ключ использует функция шифрования алгоритма DES?
7. Для решения какой задачи используется расширитель при реализации функции шифрования?

8. Представьте структурную схему реализации функции шифрования алгоритма DES.

9. Сколько ключей необходима при шифровании 64 битового блока алгоритмом DES?

10. Представьте схему алгоритма вычисления ключей для DES.

11. Поясните принцип работы блока замены алгоритма DES.

12. Какие существуют режимы работы алгоритма DES?

13. Какой размер блока открытого текста в алгоритме DES?

14. Каким методом можно повысить криптостойкость алгоритма DES?

15. К какому типу криптосистем относится алгоритм ГОСТ 28147-89?

16. Сколько различных ключей существует в алгоритме ГОСТ 28147-89?

17. Какие основные режимы (базовые циклы) работы предусматривает алгоритм ГОСТ 28147-89?

18. Какой размер имеет таблица замен в ГОСТ 28147-89?

19. Представьте схему основного шага криптопреобразования алгоритма ГОСТ 28147-89.

20. Представьте порядок использования ключевых элементов в циклах зашифрования и расшифрования ГОСТ 28147-89.

21. Представьте схему цикла зашифрования (расшифрования) алгоритма ГОСТ 28147-89.

22. Поясните отличие режима гаммирования от гаммирования с обратной связью в ГОСТ28147-89.

23. Сколько циклов используется в алгоритме ГОСТ 28147-89 для зашифровывания и расшифровывания информации?

24. Какой номер подключа, используемого на 9-м цикле расшифровывания информации в алгоритме ГОСТ 28147-89?

25. Какие основные достоинства и недостатки симметричных криптосистем?

26. Приведите качественное сравнение алгоритмов шифрования DES и ГОСТ 28147-89 и обоснуйте свои выводы.

Семинарское занятие № 1

СИММЕТРИЧНЫЕ СИСТЕМЫ ШИФРОВАНИЯ DES и ГОСТ 28147-89

Теория для занятия представлена в разделе 4.

Перед выполнением тестовых заданий проводится опрос с использованием вопросов, представленных в разделе 4.

Тестовые задания

1. Симметричной криптосистема называется потому, что:
   а) ключ, используемый в процессе зашифровывания, симметричен ключу, используемому в процессе расшифровывания;
   б) в процессе зашифровывания и расшифровывания используется один и тот же ключ;

в) шифротекст обладает внутренней симметрией;

г) в процессе зашифровывания и расшифровывания используются разные ключи.

2. Размер блока информации обрабатываемого алгоритмом DES равен:

а) 32 бита; б) 56 бит; в) 64 бита; г) 128 бит.

3. Количество циклов в алгоритме DES равно:

а) 15; б) 4; в) 16; г) 1.

4. Разрядность подключей K_i, используемых на каждом цикле алгоритма DES равна:

а) 32 бита; б) 48 бит; в) 64 бита; г) 56 бит.

5. Количество различных ключей в алгоритме DES равно:

а) 56; б) 2^{64}; в) 2^{56}; г) 2^{48}.

6. Размер блока информации обрабатываемого алгоритмом ГОСТ 28147-89 равен:

а) 48 бит; б) 64 бита; в) 128 бит; г) 256 бит.

7. Количество различных ключей в алгоритме ГОСТ 28147-89 равно:

а) 2^{127}; б) 2^{128}; в) 2^{64}; г) 11529215046068469 76.

8. Разрядность подключей K_i, используемых на каждом цикле алгоритма ГОСТ 28147-89, равна:

а) 32 бита; б) 48 бит; в) 64 бита; г) 56 бит.

9. Количество циклов в алгоритме ГОСТ 28147-89 равно:

а) 15; б) 32; в) 16; г) 64.

10. Номер подключа K_i, используемого на десятом цикле процедуры зашифровывания в алгоритме ГОСТ 28147-89, равен:

а) 1; б) 2; в) 6; г) 0.

Задачи

1. Переведите число 3^{43} в двоичную систему счисления.

2. Пусть каждая из 16 первых букв русского алфавита (абвгдежзийклмноп) имеет четырехразрядный двоичный код, соответствующий ее номеру от 0 до 15, т.е. *а* – 0000_2, *б* – 0001_2, …, *п* – 1111_2. Составьте из этих букв произвольное сообщение состоящее из 32 букв, затем разбейте полученное сообщение на блоки длиной 64 бита. Значения полученных блоков запишите в десятичной системе счисления.

3. Найдите сумму по mod 2 следующих пар чисел:

а) 224489301 и 28973675;

б) 3479913811 и 2301120149;
в) 3040958609 и 2781188359;
г) 3075166647 и 3785852425.

4. Найдите сумму по mod 3 следующих пар чисел:
а) 3496 и 3718;
б) 3668 и 1419;
в) 5563 и 6482;
г) 6379 и 1215.

5. Найдите сумму по модулю 2^{32} следующих пар чисел:
а) 3037741847 и 1257225448;
б) 2706981523 и 1587985773;
в) 2597745569 и 1697221728;
г) 13862897145741766693.

6. Найдите 4-х битовое число на выходе блока замены S_2 (функции шифрования алгоритма DES), если на вход подать число:
а) 15; б) 62; в) 23; г) 45.

7. Найдите 64-х битовое число на выходе блока перестановки *IP*, алгоритма DES, если на вход подано число:
а) 16175076002153763172;
б) 11870809655824700300;
в) 14436987034089088762;
г) 12727938706001950544.

Модуль 5

СОВРЕМЕННЫЕ АСИММЕТРИЧНЫЕ КРИПТОСИСТЕМЫ

Цель модуля – изучение слушателями курсов переподготовки ассиметричных криптосистем и особенностей их использования при криптографическом кодировании

В результате изучения модуля слушатели должны

знать

- процедуры зашифровывания и расшифровывания информации асимметричными алгоритмами RSA , Эль-Гамаля и Рабина;
- достоинства и недостатки асимметричных криптосистем;

уметь

- использовать криптосистемы RSA, Эль-Гамаля и Рабина для шифрования информации;

иметь представление

- о комбинированных методах шифрования.

5.1. Построения систем с открытым ключом. Алгоритм RSA

5.1.1. Особенности криптосистем с открытым ключом

Как бы ни были сложны и надежны криптографические системы, их слабое место при практической реализации – проблема распределения ключей. Для того чтобы был возможен обмен конфиденциальной информацией между двумя субъектами ИС, ключ должен быть сгенерирован одним из них, а затем каким-то образом опять же в конфиденциальном порядке передан другому. Т.е. в общем случае для передачи ключа опять же требуется использование какой-то криптосистемы.

Для решения этой проблемы на основе результатов, полученных классической и современной алгеброй, были предложены системы с открытым ключом. Можно сказать, что криптосистема с открытым ключом определяется тремя алгоритмами [3, 5]: генерации ключей, шифрования и расшифрования. Алгоритм генерации ключей открыт, всякий может подать ему на вход случайную строку r надлежащей длины и получить пару ключей ($K(o)$, $K(c)$). Один из ключей (например, $K(o)$) публикуем, он называется открытым, а второй $K(c)$, называемый секретным (закрытым), хранится в тайне. Алгоритмы шифрования $E_{K(o)}$ и расшифрования $D_{k(c)}$ таковы, что для любого открытого текста M: $D_{K(c)}(E_{K(o)}(M)) = M$.

Открытый ключ доступен любому, кто желает послать сообщение адресату. Исходный текст шифруется открытым ключом адресата и передается ему. Зашифрованный текст, в принципе, не может быть расшифрован тем же открытым ключом.

Дешифрование сообщения возможно только с использованием закрытого ключа, который известен только самому адресату.

Криптографические системы с открытым ключом используют так называемые необратимые или односторонние функции, которые обладают следующим свойством: при заданном значении x относительно просто вычислить значение $f(x)$, однако если $y = f(x)$, то нет простого пути для вычисления значения x.

В самом определении необратимости присутствует неопределенность. Под необратимостью понимается не теоретическая необратимость, а практическая невозможность вычислить обратное значение, используя современные вычислительные средства за обозримый интервал времени.

Поэтому для гарантии надежной защиты информации к системам с открытым ключом (СОК) предъявляются два важных и очевидных требования [3, 5]:

- преобразование исходного текста должно быть необратимым и исключать его восстановление на основе открытого ключа;
- определение закрытого ключа на основе открытого также должно быть невозможным на современном технологическом уровне. При этом желательна точная нижняя оценка сложности (количества операций) раскрытия шифра.

Алгоритмы шифрования с открытым ключом получили широкое распространение в современных информационных системах. Все предлагаемые сегодня криптосистемы с открытым ключом опираются на один из следующих типов необратимых преобразований [2, 5]:

- разложение больших чисел на простые множители;
- вычисление логарифма в конечном поле;
- вычисление корней алгебраических уравнений.

Здесь же следует отметить, что алгоритмы криптосистемы с открытым ключом можно использовать в трех назначениях:

- как самостоятельные средства защиты передаваемых и хранимых данных;
- как средства для распределения ключей;
- как средства для цифровой подписи документов.

Концепция криптографии с открытыми ключами была выдвинута Уитфилдом Диффи (Whitfield Diffie) и Мартином Хеллманом (Martin

Hellman) и независимо Ральфом Мерклом (Ralph Merkle). Их вкладом в криптографию было убеждение, что ключи можно использовать парами – ключ шифрования и ключ дешифрирования – и что может быть невозможно получить один ключ из другого. Диффи и Хеллман впервые представили эту идею на Национальной компьютерной конференции (National Computer Conference) 1976 года. С этого года было предложено множество криптографических алгоритмов с открытыми ключами. Многие из них небезопасны. Из тех, которые являются безопасными, многие непригодны для практической реализации. Либо они используют слишком большой ключ, либо размер полученного шифротекста намного превышает размер открытого текста.

Немногие алгоритмы являются и безопасными, и практичными. Некоторые из этих безопасных и практичных алгоритмов подходят только для распределения ключей. Другие подходят для шифрования (и для распределения ключей). Третьи полезны только для цифровых подписей. Только три алгоритма хорошо работают как при шифровании, так и для цифровой подписи: RSA, EIGamal, Rabin. Но все эти алгоритмы медленны. Они шифруют и дешифрируют данные намного медленнее, чем симметричные алгоритмы. Обычно их скорость недостаточна для шифрования больших объемов данных.

5.1.2. Алгоритм RSA. Шифрование и дешифрования RSA

Из всех предложенных за эти годы алгоритмов с открытыми ключами RSA проще всего реализовать, и этот алгоритм многие годы противостоит интенсивному криптоанализу [3, 5]. Хотя криптоанализ ни доказал, ни опроверг безопасность RSA, он, по сути, обосновывает уровень доверия к алгоритму. Безопасность RSA основана на трудности разложения на множители больших чисел. Открытый и закрытый ключи являются функциями двух больших (100 – 200 десятичных разрядов или даже больше) простых чисел. Предполагается, что восстановление открытого текста по шифротексту и открытому ключу эквивалентно разложению на множители двух больших чисел.

Несмотря на довольно большое число различных СОК, наиболее популярна – криптосистема RSA, разработанная в 1977 году и получившая название в честь ее создателей: Рона Ривеста (Rivest), Ади Шамира (Shamir) и Леонарда Адлемана (Adleman).

Они воспользовались тем фактом, что нахождение больших простых чисел в вычислительном отношении осуществляется легко, но разложение на множители произведения двух таких чисел практически невыполнимо. Доказано (теорема Рабина), что раскрытие шифра RSA эквивалентно такому разложению. Поэтому для любой длины ключа можно дать нижнюю

оценку числа операций для раскрытия шифра, а с учетом производительности современных компьютеров оценить и необходимое на это время. Время выполнения наилучших из известных алгоритмов разложения при значении модуля $n > 10^{145}$ выходит за пределы современных технологических возможностей.

Возможность гарантированно оценить защищенность алгоритма RSA стала одной из причин популярности этой СОК на фоне десятков других схем. Поэтому алгоритм RSA используется в банковских компьютерных сетях, особенно для работы с удаленными клиентами (обслуживание кредитных карточек).

Действия получателя криптограммы B:

1. Получатель B генерирует два произвольных больших простых числа p и q. Эти числа должны быть примерно одинаковыми, размерностью 100 - 200 десятичных разрядов. Они должны быть секретными.

2. Получатель B вычисляет значение модуля $n = p \cdot q$ и функции Эйлера $\varphi(n) = (p-1)\cdot(q-1)$ и выбирает значение открытого ключа K_O с соблюдением условий: $1 < K_O \le \varphi(n)$, $(K_O, \varphi(n)) = 1$, т.е. K_O и $\varphi(n)$ должны быть взаимно простыми.

3. Получатель B вычисляет значение секретного ключа K_C (обратного числа к числу K_O по модулю $\varphi(n)$):

$$K_C = \left(K_O^{-1}\right) \bmod \varphi(n).$$

4. В посылает A пару чисел n, K_O по открытому каналу.

Действия отправителя криптограммы A:

1. Разбивает исходный текст M на блоки M_i, $i = 1,2,\ldots,m$, т.е. $M = M_1, M_2, \ldots, M_m$. Величина $M_i < n$. Т.е. если p и q – 100-разрядные простые числа, то n будет содержать около 200 разрядов, и каждый блок сообщения m должен быть около 200 разрядов в длину. Если нужно зашифровать фиксированное число блоков, их можно дополнить несколькими нулями слева, чтобы гарантировать, что блоки всегда будут меньше n.

2. Шифрует каждое число M_i по формуле $C_i = \left(M_i^{K_O}\right) \bmod n$ и отправляет криптограмму $C = C_1, C_2, \ldots, C_m$.

Получатель B, получив криптограмму, расшифровывает каждый блок секретным ключом K_C, $M_i = \left(C_i^{K_C}\right) \bmod n$, и восстанавливает весь текст $M = M_1, M_2, \ldots, M_m$.

Пример. Шифрование сообщения «CAB»

Для простоты вычислений будут использоваться небольшие числа.

Действия получателя *B*:

1. Выбирает $p = 3$ и $q = 11$.
2. Вычисляет модуль $n = pq = 3 \cdot 11 = 33$.
3. Вычисляет значение функции Эйлера для $N = 33$:

$$\varphi(n) = \varphi(33) = (p-1)(q-1) = 2 \cdot 10 = 20.$$

Выбирает в качестве открытого ключа K_O произвольное число с учетом выполнения условий: $1 < K_O \le 20$, НОД (K_O, 20) = 1. Пусть $K_O = 7$.

4. Вычисляет значение секретного ключа K_C, используя расширенный алгоритм Евклида при сравнении $K_C \equiv 7^{-1} \pmod{20}$. Решение дает $K_C = 3$.
5. Пересылает *A* пару чисел ($n = 33$, $K_O = 7$).

Действия получателя криптограммы *A*:

6. Представляет шифруемое сообщение как последовательность целых чисел. Пусть буква A представляется как число 1, буква B – 2, C – 3. Тогда сообщение «CAB» можно представить как последовательность чисел 312, т.е. $M_1 = 3$, $M_2 = 1$, $M_3 = 2$.
7. Шифрует текст, представленный в виде последовательности чисел M_1, M_2, M_3, используя ключ $K_O = 7$, и $N = 33$, по формуле

$$C_i = M_i^{K_O} (\mathrm{mod}\, N) = M_i^7 (\mathrm{mod}\, 33).$$

Получает:

$C_1 = 3^7 \pmod{33} = 2187 \pmod{33} = 9$;

$C_2 = 1^7 \pmod{33} = 1 \pmod{33} = 1$;

$C_3 = 2^7 \pmod{33} = 128 \pmod{33} = 29$.

Отправляет *B* криптограмму C_1, C_2, C_3 = 9, 1, 29.

Действия *B*:

8. Расшифровывает принятую криптограмму C_1, C_2, C_3, используя секретный ключ $K_C = 3$, по формуле $M_i = C_i^{K_C} (\mathrm{mod}\, N) = C_i^3 (\mathrm{mod}\, 33)$.

Получает:

$M_1 = 9^3 \pmod{33} = 729 \pmod{33} = 3$;

$M_2 = 1^3 \pmod{33} = 1 \pmod{33} = 1$;

$M_3 = 29^3 \pmod{33} = 24389 \pmod{33} = 2$.

Таким образом, восстановлено исходное сообщение: CAB.

Существует вариант криптосистемы RSA, в которой вместо функции Эйлера используется функция Кармайкла λ, где λ(n) – наименьшее целое t, такое что для любого целого x, взаимно простого с n, выполняется $x^t = 1 \bmod n$. Если n выбирается так, как описано выше, то λ(n) = НОК ($p - 1$, $q - 1$).

Аспекты практической реализации и безопасности

Покажем, что при расшифровывании восстанавливается исходный текст. Согласно обобщению Эйлером малой теоремы Ферма: если

$$\text{НОД}(a,n)=1 \text{ и } a^{\varphi(n)+1} \equiv a \bmod n .$$

Открытый K_O и закрытый K_C ключи в алгоритме связаны соотношением $K_O \cdot K_C \equiv 1 \bmod \varphi(n)$, или $K_O \cdot K_C = k \cdot \varphi(n) + 1$ для некоторого целого k. Таким образом, процесс шифрования, а затем расшифровывания некоторого сообщения M_i выглядит следующим образом:

$$\left(\left(M_i^{K_O}\right) \bmod n\right)^{K_C} \bmod n = \left(M_i^{K_O \cdot K_C}\right) \bmod n = \left(M_i^{k \cdot \varphi(n)+1}\right) \bmod n = M_i .$$

В процессе применения RSA злоумышленник может иметь C_i, K_O, n и организовать дешифрование двумя способами:

- по C_i, K_O, n получить M_i. Для этого он решает задачу вычисления M_i из уравнения $C_i = M_i^{K_O} \bmod n$. Эта задача вычислительно трудна;
- по n вычислить p, q, затем найти $\varphi(n)$ и вычислить $K_C = \left(K_O^{-1}\right) \bmod \varphi(n)$ и дешифровать сообщение $M_i = C_i^{K_C} \bmod n$.

Однако задача разложения большого числа на простые множители вычислительно сложна.

Пользователи A и B должны быстро осуществлять все вычисления: вычислять K_O, шифровать и расшифровывать.

Вычисление K_O с использованием алгоритма Евклида – довольно быстрый процесс и не представляет трудности. Зашифровывание и расшифровывание – возведение большого числа в большую степень – требует определенных затрат времени, но, с учетом наличия быстрых алгоритмов и быстродействия современных компьютеров, это приемлемая процедура.

В настоящее время алгоритм RSA активно реализуется как в виде самостоятельных криптографических продуктов, так и в качестве встроенных средств.

Важная проблема практической реализации – генерация больших простых чисел. Решение задачи «в лоб» – генерация случайного большого числа n (нечетного) и проверка его делимости на множители. В случае неуспеха следует взять $n + 2$ и т.д.

Другая проблема – ключи какой длины следует использовать? В 1994 г. было факторизовано число со 129 десятичными цифрами. Это удалось осуществить математикам А. Ленстра и М. Манасси посредством организа-

ции распределенных вычислений на 1600 компьютерах, объединенных сетью, в течение восьми месяцев. По мнению А. Ленстра и М. Манасси, их работа компрометирует криптосистемы RSA и создает большую угрозу их дальнейшим применениям. Теперь разработчикам криптоалгоритмов с открытым ключом на базе RSA приходится избегать применения чисел длиной менее 200 десятичных разрядов. Последние публикации предлагают применять для этого числа длиной не менее 250 – 300 десятичных разрядов.

Была сделана попытка расчета оценок безопасных длин ключей асимметричных криптосистем на ближайшие 20 лет исходя из прогноза развития компьютеров и их вычислительной мощности, а также возможного совершенствования алгоритмов факторизации. Эти оценки (табл. 5.1) даны для трех групп пользователей (индивидуальных пользователей, корпораций и государственных организаций) в соответствии с различием требований к их информационной безопасности. Конечно, данные оценки следует рассматривать как сугубо приблизительные, как возможную тенденцию изменений безопасных длин ключей асимметричных криптосистем со временем.

Таблица 5.1

Оценка длин ключей для асимметричных криптосистем, бит

Год	Отдельные пользователи	Корпорации	Государственные организации
1995	768	1280	1536
2000	1024	1280	1536
2005	1280	1536	2048
2010	1280	1536	2048
2015	1536	2048	2048

Третий немаловажный аспект реализации RSA – вычислительный. Ведь приходится использовать аппарат длинной арифметики. Если используется ключ длиной k бит, то для операций по открытому ключу требуется $O(k^2)$ операций, по закрытому ключу – $O(k^3)$ операций, а для генерации новых ключей требуется $O(k^4)$ операций. Самая «быстрая» аппаратная реализация обеспечивает скорости в 100 раз больше компьютерной. По сравнению с тем же алгоритмом DES, RSA требует в тысячи раз большее время.

Алгоритм RSA входит в стандарт шифрования ISO9796. Сам алгоритм RSA запатентован только в США. Французское и австралийское банковские сообщества приняли RSA в качестве стандарта.

5.2. Криптосистема Эль-Гамаля. Алгоритм Рабина. Комбинированный метод шифрования

5.2.1. Криптосистема Эль-Гамаля

Схема Эль-Гамаля, предложенная в 1985 г., может быть использована как для шифрования, так и для цифровых подписей. Данная система является альтернативой RSA и при равном значении ключа обеспечивает ту же криптостойкость. В отличие от RSA метод Эль-Гамаля основан на проблеме дискретного логарифма. Если возводить число в степень в конечном поле достаточно легко, то восстановить аргумент по значению (т.е. найти логарифм) довольно трудно.

Множество параметров системы включает простое число p и целое число q, степени которого по модулю p порождают большое число элементов Z_p.

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

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 $a = q^K \bmod p$, $b = Y^K M \bmod p$. Пара чисел (a, b) является шифротекстом. Заметим, что длина шифротекста вдвое больше длины исходного открытого текста M.

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

$$M = b/a^X \bmod p.$$

Поскольку $a^X \equiv q^{Kx} \bmod p$, $b/a^x \equiv Y^K M/a^x \equiv G^{Kx} M/G^{Kx} \equiv M \pmod p$, то соотношение справедливо.

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

Вычисляем $Y = q^x \bmod p = 2^8 \bmod 11 = 256 \bmod 11 = 3$.

Итак, открытый ключ $Y = 3$.

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

Выполним расшифровывание этого шифротекста. Вычисляем сообщение *M*, используя секретный ключ *X*: $M = b/a^X \bmod p = 9/6^8 \bmod 11$. Выражение $M \equiv 9/6^8 \bmod 11$ можно представить в виде $6^8 \cdot M \equiv 9 \bmod 11$ или $1679616 \cdot M \equiv 9 \bmod 11$. Решая данное сравнение, находим $M = 5$.

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

5.2.2. Алгоритм Рабина

Безопасность схемы Рабина (Rabin) опирается на сложность поиска квадратных корней по модулю составного числа [6]. Эта проблема аналогична разложению на множители. Рассмотрим одну из реализаций этой схемы.

Сначала выбираются два простых числа p и q, конгруэнтных $3 \bmod 4$. Эти простые числа являются закрытым ключом, а их произведение $n = pq$ – открытым.

Для шифрования сообщения M (M должно быть меньше n) просто вычисляется

$$C = M^2 \bmod n.$$

Дешифрирование сообщения также несложно. Поскольку получатель знает p и q, он может решить две конгруэнтности с помощью китайской теоремы об остатках. Вычисляются

$$m_1 = C^{(p+1)/4} \bmod p;$$

$$m_2 = (p - C^{(p+1)/4}) \bmod p;$$

$$m_3 = C^{(q+1)/4} \bmod q;$$

$$m_4 = (q - C^{(q+1)/4}) \bmod q.$$

Затем выбирается целые числа $a = q(q^{-1} \bmod p)$ и $b = p(p^{-1} \bmod q)$. Четырьмя возможными решениями являются:

$$M_1 = (am_1 + bm_3) \bmod n;$$

$$M_2 = (am_1 + bm_4) \bmod n;$$

$$M_3 = (am_2 + bm_3) \bmod n;$$

$$M_4 = (am_2 + bm_4) \bmod n.$$

Один из четырех результатов, M_1, M_2, M_3, M_4, равен M. Если сообщение написано по-английски, выбрать правильное M нетрудно. С дру-

гой стороны, если сообщение является потоком случайных битов (скажем, для генерации ключей или цифровой подписи), способа определить, какое M – правильное, нет. Одним из способов решить эту проблему служит добавление к сообщению перед шифрованием известного заголовка.

Хью Вильяме (Hugh Williams) переопределил схему Рабина, чтобы устранить эти недостатки. В его схеме p и q выбираются так, чтобы

$$p \equiv 3 \bmod 8;$$

$$q \equiv 7 \bmod 8;$$

$$N = pq.$$

Кроме того, используется небольшое целое число S, для которого $J(S,N) = -1$, где J – это символ Якоби. N и S опубликовываются. Секретным ключом является k, для которого

$$k = 1/2(1/4(p-1)(q-1)+1).$$

Для шифрования сообщения M вычисляется c_1 такое, что $J(M,N) = (-1)^{c_1}$. Затем вычисляется $M' = (S^{c_1} * M) \bmod N$. Как и в схеме Рабина, $C = M'^2 \bmod N$. И $c_2 = M' \bmod 2$. Окончательным шифротекстом сообщения является тройка

$$(C, c_1, c_2).$$

Для дешифрирования C получатель вычисляет M'' с помощью

$$C^k \equiv \pm M'' (\bmod N).$$

Правильный знак M'' определяет c_2. Наконец,

$$M = (S^{c_1} * (-1)^{c_1} * M'') \bmod N.$$

5.2.3. 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 зашифровывания и расшифровывания, связанные с возведением в степень многозначного числа, достаточно громоздки.

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, характерными симметричным криптосистемам с секретным ключом. При таком подходе криптосистема с открытым ключом применяется для зашифровывания, передачи и последующего расшифровывания только секретного ключа симметричной криптосистемы. А симметричная криптосистема применяется для зашифровывания 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_S.
2. Зашифровать сообщение *M* на сеансовом ключе K_S.
3. Зашифровать сеансовый ключ K_S на открытом ключе K_O 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_C сеансовый ключ K_S.
6. С помощью полученного сеансового ключа K_S расшифровать и прочитать сообщение *M*.

При использовании комбинированного метода шифрования можно быть уверенным в том, что только пользователь *B* сможет правильно расшифровать ключ K_S и прочитать сообщение *M*.

Таким образом, при комбинированном методе шифрования применяются криптографические ключи как симметричных, так и асимметричных криптосистем. Очевидно, выбор длин ключей для каждого типа криптосистемы следует осуществлять таким образом, чтобы злоумышленнику

было одинаково трудно атаковать любой механизм защиты комбинированной криптосистемы. Структурная схема комбинированной системы представлена на рис. 5.1.

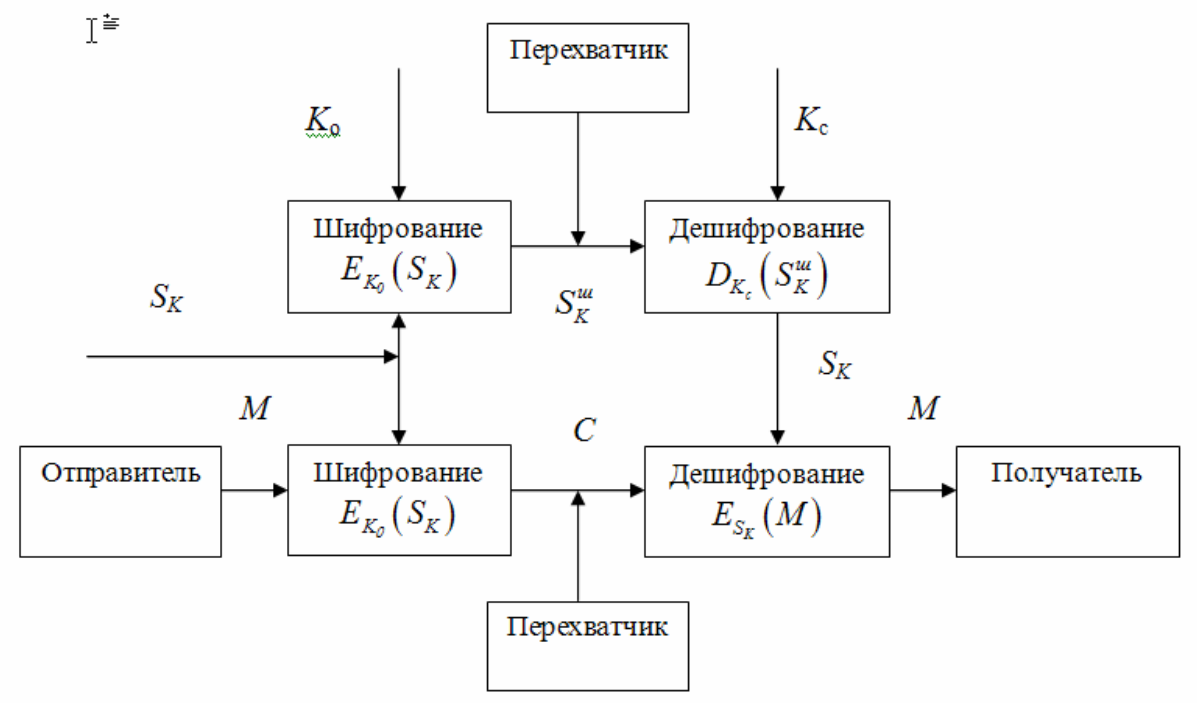

Рис. 5.1. **Схема комбинированного метода шифрования**

В табл. 5.2 приведены распространенные длины ключей симметричных и асимметричных криптосистем, для которых трудность атаки полного перебора примерно равна трудности факторизации соответствующих модулей асимметричных криптосистем.

Таблица 5.2

Длины ключей для симметричных и асимметричных криптосистем при одинаковой их криптостойкости

Для симметричной криптосистемы, бит	Для асимметричной криптосистемы, бит
56	384
64	512
80	768
112	1 792
128	2 304

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

Вопросы и задания для самопроверки

1. Какие требования предъявляются к ассимметричным криптосистемам?
2. В чем заключается сущность алгоритма рюкзака?
3. Как получить нормальную последовательность из сверхвозрастающей?
4. Каким образом формируются открытый и закрытый ключи в алгоритме рюкзака?
5. Приведите пример шифрования и дешифрования информации с помощью алгоритма рюкзака.
6. Кто создал алгоритм RSA?
7. Какой длины должен использоваться ключ в алгориме RSA?
8. Сформулируйте малую теорему Ферма.
9. Дайте определение функции Эйлера.
10. Как находится модуль *n* в криптосистеме RSA?
11. На чем основана криптостойкость RSA?
12. Как связаны открытый и секретный ключи в алгоритме RSA?
13. Сформулируйте последовательность действий получателя и отправителя сообщения при использовании алгоритма RSA.
14. Приведите пример шифрования сообщения алгориммом RSA.
15. Какими способами противник может организовать дешифрование RSA?
16. Какие размеры модуля *n* рекомендуется использовать в алгоритме RSA?
17. Что такое дискретный логарифм?
18. Как соотносится размер шифротекста и открытого текста в алгоритме Эль-Гамаля?
19. В чем заключается недостаток алгоритма Эль-Гамаля?
20. Приведите пример шифрования алгоримом Эль-Гамаля.
21. На чем основана безопасность алгорима Рабина?
22. В чем сущность модификации алгорима Рабина?
23. Приведите пример шифрования алгоримом Рабина и его модификацией?
24. Что такое цифровой конверт?
25. Что Вы понимаете под комбинированным методом шифрования?
26. Сформулируйте алгоритм работы комбинированной системы шифрования.
27. Представьте структурную схему комбинированной криптосистемы.
28. Каковы преимущества комбинированных систем шифрования?

Семинарское занятие № 2

АССИМЕТРИЧНЫЕ СИСТЕМЫ ШИФРОВАНИЯ

Теория для занятия представлена в разделе 5.

Перед выполнение тестовых заданий проводиться опрос с использованием вопросов, представленных в разделе 5.

Тестовые задания

1. Асимметричной, криптосистема называется потому, что:
   а) в процессе зашифровывания и расшифровывания используются разные ключи;

б) процесс зашифровывания непохож на процесс расшифровывания;
в) в процессе зашифровывания и расшифровывания используется один и тот же ключ;
г) шифротекст не обладает внутренней симметрией.

2. Функция Эйлера $\varphi(n)$ находит:
а) количество чисел больше n, которые являются взаимно простыми с n;
б) количество простых чисел меньше n;
в) количество чисел меньше n, которые не являются взаимно простыми с n;
г) количество простых чисел больше n.
д) количество чисел меньше n, которые являются взаимно простыми с n.

3. Какой тип необратимых преобразований используют агоритмы шифрования с открытым ключом:
а) разложение больших чисел на простые множители;
б) разложение простых чисел на множители;
в) вычисление логарифма в конечном поле;
г) вычисление корней алгебраических уравнений.

4. Алгоритмы криптосистемы с открытым ключом можно использовать:
а) как самостоятельные средства защиты передаваемых и хранимых данных;
б) как средства для распределения ключей;
в) как средства для сжатия информации;
г) как средства для цифровой подписи документов.

5. Как связаны открытый и секретный ключи в алгоритме RSA?
а) являются обратными числами по модулю n;
б) никак не связаны;
в) их произведение равно n;
г) являются обратными числами по модулю $\varphi(n)$.

6. Как связаны открытый и секретный ключи в алгоритме Эль-Гамаля?
а) являются обратными числами;
б) функцией дискретного логарифма;
в) никак не связаны;
г) функцией sin.

7. Как связаны открытый и секретный ключи в алгоритме Рабина?
а) являются обратными числами по модулю n;
б) никак не связаны;

в) открытый ключ равен произведению закрытых;

г) являются обратными числами по модулю $\varphi(n)$.

8. Алгоритмы, лежащие в основе криптосистем с открытым ключом, имеют недостатки:

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

б) невозможно оценить криптостойкость ассимметричных систем;

в) процедуры зашифровывания и расшифровывания, связанные с возведением в степень многозначного числа, достаточно громоздки;

г) низкая криптостойкость по сравнению с симметричными системами.

9. Какое количество ключей исползуется в комбинированной системе шифрования?

а) 2; б) 3; в) 4; г) 5.

10. В комбинированной системе шифрования по открытому каналу связи передается:

а) зашифрованный открытый ключ;

б) зашифрованный сеансовый ключ;

в) зашифрованное сообщение;

г) зашифрованный открытый ключ и зашифрованное сообщение.

Задачи

1. Найдите значение функции Эйлера $\varphi(n)$ следующих чисел:

а) 15; б) 72; в) 311; г) 128.

2. Зашифруйте методом RSA ($K_O = 91,\ n = 323$) следующие числа:

а) 35; б) 94; в) 248; г) 236.

3. Расшифруйте методом RSA ($K_C = 3,\ n = 33$) следующие числа:

а) 294; б) 531; в) 7; г) 111.

4. Зашифруйте методом Эль-Гамаля ($p = 43,\ q = 33,\ Y = 26,\ K = 11$) следующие числа:

а) 16; б) 7; в) 31; г) 23.

5. Расшифруйте методом Эль-Гамаля ($p = 47,\ X = 21$) следующие варианты шифротекста (a,b):

а) $(45,20)$; б) $(45,11)$; в) $(45,14)$; г) $(45,29)$.

ЛИТЕРАТУРА

1. Баричев, С.Г. Основы современной криптографии / С.Г. Баричев, В.В. Гончаров, Р.Е. Серов. – М.: Горячая линия – Телеком, 2001. – 152 с.

2. Молдовян, А.А. Криптография / А.А. Молдовян, Н.А. Молдовян, Б.А. Советов. – СПб.: Лань, 2000.

3. Саломаа, А. Криптография с открытым ключом / А. Саломаа; пер. с англ. – М.: Мир, 1995. – 318 с.

4. Скляров, Д.В. Искусство защиты и взлома информации / Д.В. Скляров. – СПб.: БХВ-Петербург, 2004. – 288 с.: ил.

5. Теория прикладного кодирования: в 2 т.: учеб. пособие / В.К. Конопелько [и др.]; под ред. проф. В.К. Конопелько. – Минск: БГУИР, 2004. – Т. 1.

6. Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Б. Шнайер; пер. с англ. – М.: Триумф, 2002. – 816 с.

СОДЕРЖАНИЕ

Учебное издание

БОГУШ Рихард Петрович
КУРИЛОВИЧ Андрей Владимирович

ОСНОВЫ ЗАЩИТЫ ИНФОРМАЦИИ

Учебно-методический комплекс
для слушателей ИПК специальности 1-40 01 73
«Программное обеспечение информационных систем»

Редактор *Т. А. Дарьянова*
Дизайн обложки *В. А. Виноградовой*

Подписано в печать 03.04.09. Формат 60х84 1/16. Гарнитура Таймс. Бумага офсетная. Ризография. Усл. печ. л. 5,57. Уч.-изд. л. 5,38. Тираж 30 экз. Заказ 661.

Издатель и полиграфическое исполнение –
учреждение образования «Полоцкий государственный университет»

ЛИ № 02330/0133020 от 30. 04. 04 ЛП № 02330/0133128 от 27.05.04

211440, г. Новополоцк, ул. Блохина, 29