Федеральное агентство по образованию
Сибирский государственный аэрокосмический университет
имени академика М. Ф. Решетнева

О.Н. ЖДАНОВ
В. В. ЗОЛОТАРЕВ

МЕТОДЫ И СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ

Рекомендовано в качестве учебного пособия
для студентов, обучающихся по специальностям
«Комплексное обеспечение информационной безопасности
автоматизированных систем»
«Информационная безопасность телекоммуникационных систем»

Красноярск 2007

УДК 004.042
ББК 32.973.26-018.2
З 80

РЕЦЕНЗЕНТЫ:

Зав. кафедрой высшей математики Сибирского государственного аэрокосмического университета, д.ф.-м.н., профессор А.М. Попов;

Профессор Кызыл-Кийского института технологий, экономики и права Баткенского государственного университета, к.ф.-м.н. Т.К. Юлдашев.

Жданов, О. Н., Золотарев, В. В.

З 80 Методы и средства криптографической защиты информации: Учебное пособие / О.Н. Жданов, В. В. Золотарев; СибГАУ. – Красноярск, 2007. – 217 с.

Цель учебного пособия – научить студентов основным методам и средствам криптографической защиты информации, их применению и анализу особенностей. В пособии приведены основные положения курса, а также некоторые новые результаты в предметной области, полученные авторами и их учениками. Пособие содержит контрольные вопросы для самопроверки и задачи по указанному курсу.

Учебное пособие предназначено для студентов, обучающихся по специальностям 090105 «Комплексное обеспечение информационной безопасности автоматизированных систем», 090106 «Информационная безопасность телекоммуникационных систем» всех форм обучения.

УДК 004.042
ББК 32.973.26-018.2

Учебное издание
ЖДАНОВ Олег Николаевич
ЗОЛОТАРЕВ Вячеслав Владимирович
МЕТОДЫ И СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ

Учебное пособие

Редактор
Компьютерная верстка
Подп. в печать _____. Формат 60×84/16. Бумага офисная.
Печать плоская. Усл. печ. л. 6,0. Уч.-изд. л. 6,0.
Тираж 100 экз. Заказ 410. С 70.
Санитарно-эпидемиологическое заключение
№ 24.04.953. П.000032.01.03. от 29.01.2003 г.
Редакционно-издательский отдел СибГАУ.
660014, г. Красноярск, просп. им. газ. «Красноярский рабочий», 31.
Отпечатано в типографии «Город».
660014, г. Красноярск, ул. Юности, 24а.

ОГЛАВЛЕНИЕ

СПИСОК ПРИНЯТЫХ СОКРАЩЕНИЙ

АНБ – Агентство национальной безопасности США (National Security Agency, NSA)

АПК -

АРМ – автоматизированное рабочее место

БСШ – [шифр], близкий к совершенному

ГПК – генератор псевдослучайных чисел

ДНФ – дизъюнктивная нормальная форма

ИНФ – индуктивная нормальная форма

КАС – код аутентификации сообщения

КНФ – конъюнктивная нормальная форма

КШ - криптошлюз

МНФ – минимальная нормальная форма

НСД – несанкционированный доступ

НФ – нормальная форма

ПФК – полная функциональная константа

РГПЧ – рекуррентный генератор последовательных чисел

СБИС – сверхбольшая интегральная схема

СКЗИ – средство криптографической защиты информации

СНФ – совершенно нормальная форма

УКЗД – устройство криптографической защиты данных

УЦ – удостоверяющий центр

ЭЦП – электронная цифровая подпись

AES – Advanced Encryption Standard (улучшенный стандарт шифрования)

ANSI – American National Standards Institute (Американский национальный институт стандартов)

BSAFE

CBC

CDMF – Commercial Data Masking Facility (коммерческое средство маскирования данных)

CFS – Cryptographic File System (криптографическая файловая система)

CFB

DEA – Data Encryption Algorithm

DES – Data Encryption Standard

DESX

DMS – Defense Messaging System (система защиты сообщений)

ECB

EDE – encrypt–decrypt–encrypt

GDES – Generalized DES (обобщенный DES)

IDEA – International Data Encryption Algorithm (международный алгоритм шифрования данных)

IPES – Improved Proposed Encryption Standard

ISDN

IV – /////// (инициализирующий вектор)

FEE – Fast Elliptic Encryption (быстрое эллиптическое шифрование)

LFSR – Linear Feedback Shift Register (регистр сдвига с линейной обратной связью)

MMB – Modular Multiplication-based Block cipher

NBS – National Bureau of Standards (Национальное бюро стандартов, США)

NIST – National Institute of Standards and Technology (Национальный институт стандартов и техники, США)

OFB

PES – Proposed Encryption Standard

RDES

RSA – Rivest, Shamir, Aldeman

SHA

TEMK – Triple Encryption with Minimum Key (тройное шифрование с минимальным ключом)

VNP - /////// (виртуальная частная сеть)

ПРЕДИСЛОВИЕ

В пособии рассмотрены основные направления деятельности специалиста в области криптографической защиты информации. Кроме того, описаны особенности различных сфер применения криптографических методов и средств, перечислены и кратко рассмотрены основные методы и средства криптографической защиты информации. Приведены справочные данные.

Данное учебное пособие предназначено для оказания помощи студентам в изучении дисциплин «Криптографические методы защиты информации», «Средства криптографической защиты информации», подготовке к лабораторным и практическим работам, занятиям и итоговому контролю.

Задачей группы дисциплин «Криптографические методы и средства защиты информации» является подготовка студентов в сфере разработки, исследования и эксплуатации методов и средств защиты информации, требующих использования криптографии. Знания и практические навыки, полученные из курса, могут использоваться студентами при подготовке к занятиям по предметам специализации, дипломном проектировании и в рамках научно-исследовательских работ.

В пособии приведены принципы и методы криптографического обеспечения, средства, реализующие их, кратко описаны принципы использования в рамках комплексного обеспечения информационной безопасности. Приведены примеры конкретных средств защиты информации и реальные схемы их использования на практике.

Учебное пособие подготовлено в соответствии с рабочей программой группы дисциплин «Криптографические методы и средства защиты информации», для каждой темы курсивом выделены основные положения, которые рекомендуется изучить. В конце каждого параграфа пособия даны задания для самостоятельной работы (контрольные вопросы по теме, задачи). В конце учебного пособия приведен словарь основных понятий и терминов, применяемых в предметной области, список литературы для дополнительного изучения. Пособие не является курсом лекций, поэтому более полную информацию можно получить, используя рекомендованную литературу и лекционный материал.

Учебное пособие соответствует Государственному образовательному стандарту подготовки специалистов по специальностям 090105 «Комплексное обеспечение информационной безопасности автоматизированных систем», 090106 «Информационная безопасность телекоммуникационных систем» всех форм обучения.

Для изложения общеизвестного теоретического материала в работе используются обширные цитаты из различных источников, в том числе

учебных пособий и тематических изданий. Кроме того, авторы и их ученики представляют некоторые новые результаты:

- исследование лавинного эффекта и теории экстремальных шифров совместно с Кулешом Александром Юрьевичем;
- исследование шифров, близких к совершенным, их классификация совместно с Егоровой Татьяной Михайловной;
- создание и применение подхода операционного анализа криптоалгоритмов совместно с Кукарцевым Анатолием Михайловичем;
- разработка направления мультибазисной криптографии совместно с Краковским Павлом Сергеевичем;
- изучение различных аспектов применения шифра RSA и создание тестирующих программ совместно с Чурмантаевым Динаром Мунировичем и Лубкиным Иваном Александровичем.

Авторы предлагают данное учебное пособие в качестве основы для теоретической подготовки студентов по курсам «Криптографические методы защиты информации», «Средства криптографической защиты информации» и близким дисциплинам. Работа будет также полезна аспирантам и научным работникам, исследующим аспекты разработки, применения и реализации криптографических методов и средств защиты информации.

1. КРИПТОГРАФИЯ КАК НАУЧНАЯ ДИСЦИПЛИНА

1. ОСНОВНЫЕ ПОНЯТИЯ

Как передать нужную информацию нужному адресату в тайне от других? Нетрудно прийти к выводу, что есть три возможности.

1.Создать абсолютно надежный, недоступный для посторонних канал связи между абонентами.

2.Использовать общедоступный канал связи, но скрыть сам факт передачи информации.

3.Использовать общедоступный канал связи, но передавать по нему нужную информацию в так преобразованном виде, чтобы восстановить ее мог только адресат.

Прокомментируем эти три возможности [14, гл. 1].

1.При современном уровне развития науки и техники сделать такой канал связи между удаленными абонентами для неоднократной передачи больших объемов информации практически нереально.

2.Разработкой средств и методов скрытия факта передачи сообщения занимается *стеганография.*

Первые следы стеганографических методов теряются в глубокой древности. Например, известен такой способ скрытия письменного сообщения: голову раба брили, на коже головы писали сообщение и после отрастания волос раба отправляли к адресату.

Из детективных произведений хорошо известны различные способы тайнописи между строк обычного, незащищаемого текста: от молока до сложных химических реактивов с последующей обработкой.

Также из детективов известен метод *«микроточки»:* сообщение записывается с помощью современной техники на очень маленький носитель(микроточку), который пересылается с обычным письмом, например, под маркой или где-нибудь в другом, заранее обусловленном месте.

В настоящее время в связи с широким распространением компьютеров известно много тонких методов «запрятывания» защищаемой информации внутри больших объемов информации, хранящейся в компьютере.

3. Разработкой методов преобразования *(шифрования)* информации с целью ее защиты от незаконных пользователей занимается *криптография.* Такие методы и способы преобразования информации называются *шифрами.*

Шифрование (зашифрование) — процесс применения шифра к защищаемой информации, т. е. преобразование защищаемой информации *(открытого текста)* в шифрованное сообщение *(шифртекст, криптограмму)* с помощью определенных правил, содержащихся в шифре.

Дешифрование — процесс, обратный шифрованию, т. е. преобразование шифрованного сообщения в защищаемую информацию с помощью определенных правил, содержащихся в шифре.

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

1.1.1.Предмет криптографии

Что же является предметом криптографии?

Прежде всего заметим, что эта задача возникает только для информации, которая нуждается в защите. Обычно в таких случаях говорят, что информация содержит тайну или является *защищаемой, приватной, конфиденциальной, секретной.* Для наиболее типичных, часто встречающихся ситуаций такого типа введены даже специальные понятия:

- государственная тайна;
- военная тайна;
- коммерческая тайна;
- юридическая тайна;
- врачебная тайна и т. д.

Далее мы будем говорить о защищаемой информации, имея в виду следующие признаки такой информации:

- имеется строго определенный круг *законных пользователей,* которые имеют право владеть этой информацией;
- имеются *незаконные пользователи,* которые стремятся овладеть этой информацией с тем, чтобы обратить ее себе во благо, а законным пользователям во вред.

Для простоты мы вначале ограничимся рассмотрением только одной *угрозы* — угрозы разглашения информации. Существуют и другие угрозы для защищаемой информации со стороны незаконных пользователей: подмена, имитация и др. О них мы поговорим ниже.

Ситуацию можно изобразить следующей схемой (см. рис. 1).

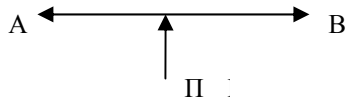


Рис.1. Угроза разглашения информации

Здесь А и В — удаленные законные пользователи защищаемой информации; они хотят обмениваться информацией по общедоступному каналу связи, П — незаконный пользователь *(противник),* который может перехватывать передаваемые по каналу связи сообщения и пытаться извлечь из них интересующую его информацию. Эту формальную схему можно считать моделью типичной ситуации, в которой применяются криптографические методы защиты информации.

Отметим, что исторически в криптографии закрепились некоторые военные слова («противник», «атака на шифр» и др.) Они наиболее точно

отражают смысл соответствующих криптографических понятий. Вместе с тем широко известная военная терминология, основанная на понятии кода (военно-морские коды, коды Генерального штаба, кодовые книги, кодобозначения и т.п.), уже не применяется в теоретической криптографии. Дело в том, что за последние десятилетия сформировалась *теория кодирования* — большое научное направление, которое разрабатывает и изучает методы защиты информации от случайных искажений в каналах связи. И если ранее термины кодирование и шифрование употреблялись как синонимы, то теперь это недопустимо. Так, например, очень распространенное выражение «кодирование — разновидность шифрования» становится просто неправильным.

Криптография занимается методами преобразования информации, которые бы не позволили противнику извлечь ее из перехватываемых сообщений. При этом по каналу связи передается уже не сама защищаемая информация, а результат ее преобразования с помощью шифра, и для противника возникает сложная задача *вскрытия шифра.*

Вскрытие (взламывание) шифра — процесс получения защищаемой информации из шифрованного сообщения без знания примененного шифра.

Однако помимо перехвата и вскрытия шифра противник может пытаться получить защищаемую информацию многими другими способами. Наиболее известным из таких способов является агентурный, когда противник каким-либо путем склоняет к сотрудничеству одного из законных пользователей и с помощью этого агента получает доступ к защищаемой информации. В такой ситуации криптография бессильна.

Противник может пытаться не получить, а уничтожить или модифицировать защищаемую информацию в процессе ее передачи. Это — совсем другой тип угроз для информации, отличный от перехвата и вскрытия шифра. Для защиты от таких угроз разрабатываются свои специфические методы.

Следовательно, на пути от одного законного пользователя к другому информация должна защищаться различными способами, противостоящими различным угрозам. Возникает ситуация цепи из разнотипных звеньев, которая защищает информацию. Естественно, противник будет стремиться найти самое слабое звено, чтобы с наименьшими затратами добраться до информации. А значит, и законные пользователи должны учитывать это обстоятельство в своей стратегии защиты: бессмысленно делать какое-то звено очень прочным, если есть заведомо более слабые звенья («принцип равнопрочности защиты»).

Не следует забывать и еще об одной важной проблеме: проблеме соотношения цены информации, затрат на ее защиту и затрат на ее добывание. При современном уровне развития техники сами средства связи, а также разработка средств перехвата информации из них и средств защиты информации требуют очень больших затрат. Прежде чем защищать информацию, задайте себе два вопроса:

1) является ли она для противника более ценной, чем стоимость атаки;

2) является ли она для вас более ценной, чем стоимость защиты.

Именно перечисленные соображения и являются решающими при выборе подходящих средств защиты: физических, стеганографических, криптографических и др.

Некоторые понятия криптографии удобно иллюстрировать историческими примерами, поэтому сделаем небольшое историческое отступление.

Долгое время занятие криптографией было уделом чудаков-одиночек. Среди них были одаренные ученые, дипломаты, священнослужители. Известны случаи, когда криптография считалась даже чёрной магией. Этот период развития криптографии как искусства длился с незапамятных времён до начала XX века, когда появились первые шифровальные машины. Понимание математического характера решаемых криптографией задач пришло только в середине XX века – после работ выдающегося американского ученого К. Шеннона

История криптографии связана с большим количеством дипломатических и военных тайн и поэтому окутана туманом легенд.

Свой след в истории криптографии оставили многие хорошо известные исторические личности. Первые сведения об использовании шифров в военном деле связаны с именем спартанского полководца Лисандра, Шифр "Сцитала". Этот шифр известен со времен войны Спарты против Афин в V веке до н.э. Для его реализации использовалась сцитала – жезл, имеющий форму цилиндра. На сциталу виток к витку наматывалась узкая папирусная лента (без просветов и нахлестов), а затем на этой ленте вдоль оси сциталы записывался текст. Лента разматывалась и получалось (для непосвященных), что поперек ленты в беспорядке написаны какие-то буквы. Затем лента отправлялась адресату. Адресат брал такую же сциталу, таким же образом наматывал на неё полученную ленту и читал сообщение вдоль оси сциталы. В этом шифре преобразование открытого текста в шифрованный заключается в определенной перестановке букв открытого текста. Поэтому класс шифров, к которым относится и шифр "Сцитала", называется шифрами перестановки.

Цезарь использовал в переписке шифр, который в историю вошёл как Шифр Цезаря. Этот шифр реализует следующее преобразование открытого текста: каждая буква открытого текста заменяется третьей после нее буквой в алфавите, который считается написанным по кругу, т.е. после буквы "я" следует буква "а". Цезарь заменял букву третьей после ней буквой, но можно заменять и какой-нибудь другой. Главное, чтобы тот, кому посылается шифрованное сообщение, знал эту величину сдвига. Класс шифров к которым относится шифр Цезаря, называется шифрами замены.

Поэтому класс шифров, к которым относится и шифр «Сцитала», называется *шифрами перестановки.*

Шифр Цезаря. Этот шифр реализует следующее преобразование открытого текста: каждая буква открытого текста заменяется третьей после нее буквой в алфавите, который считается написанным по кругу, т. е. после буквы «я» следует буква «а». Отметим, что Цезарь заменял букву третьей после нее буквой, но можно заменять и какой-нибудь другой. Главное, чтобы тот,

кому посылается шифрованное сообщение, знал эту величину сдвига. Класс шифров, к которым относится и шифр Цезаря, называется *шифрами замены.*

Из предыдущего изложения понятно, что придумывание хорошего шифра — дело трудоемкое. Поэтому желательно увеличить «время жизни» хорошего шифра и использовать его для шифрования как можно большего количества сообщений. Но при этом возникает опасность, что противник уже разгадал (вскрыл) шифр и читает защищаемую информацию. Если же в шифре есть сменный ключ, то, заменив ключ, можно сделать так, что разработанные противником методы уже не дают эффекта.

Под *ключом* в криптографии понимают сменный элемент шифра, который применяется для шифрования конкретного сообщения. Например, в шифре «Сцитала» ключом является диаметр сциталы, а в шифрах типа шифра Цезаря ключом является величина сдвига букв шифртекста относительно букв открытого текста.

Описанные соображения привели к тому, что безопасность защищаемой информации стала определяться в первую очередь ключом. Сам шифр, шифрмашина или принцип шифрования стали считать известными противнику и доступными для предварительного изучения, но в них появился неизвестный для противника ключ, от которого существенно зависят применяемые преобразования информации. Теперь законные пользователи, прежде чем обмениваться шифрованными сообщениями, должны тайно от противника обменяться ключами или установить одинаковый ключ на обоих концах канала связи. А для противника появилась новая задача — определить ключ, после чего можно легко прочитать зашифрованные на этом ключе сообщения.

Вернемся к формальному описанию основного объекта криптографии [14]. Теперь в него необходимо внести существенное изменение — добавить недоступный для противника секретный канал связи для обмена ключами (рис. 2). Создать такой канал связи вполне реально, поскольку нагрузка на него, вообще говоря, небольшая.

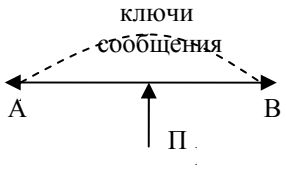


Рис. 2. Формальное описание объекта криптографии

Отметим теперь, что не существует единого шифра, подходящего для всех случаев. Выбор способа шифрования зависит от особенностей информации, ее ценности и возможностей владельцев по защите своей информации. Прежде всего подчеркнем большое разнообразие видов защищаемой информации: документальная, телефонная, телевизионная, компьютерная и т. д. Каждый вид информации имеет свои специфические особенности, и эти особенности сильно влияют на выбор методов шифрования информации. Большое значение имеют объемы и требуемая скорость передачи шифрованной информации. Выбор вида шифра и его

параметров существенно зависит от характера защищаемых секретов или тайны. Некоторые тайны (например, государственные, военные и др.) должны сохраняться десятилетиями, а некоторые (например, биржевые) — уже через несколько часов можно разгласить. Необходимо учитывать также и возможности того противника, от которого защищается данная информация. Одно дело — противостоять одиночке или даже банде уголовников, а другое дело — мощной государственной структуре.

Способность шифра противостоять всевозможным атакам на него называют *стойкостью шифра.*

Под *атакой на шифр* понимают попытку вскрытия этого шифра.

Понятие стойкости шифра является центральным для криптографии. Хотя качественно понять его довольно легко, но получение строгих доказуемых оценок стойкости для каждого конкретного шифра — проблема нерешенная. Это объясняется тем, что до сих пор нет необходимых для решения такой проблемы математических результатов. (Мы вернемся к обсуждению этого вопроса ниже.) Поэтому стойкость конкретного шифра оценивается только путем всевозможных попыток его вскрытия и зависит от квалификации *криптоаналитиков,* атакующих шифр. Такую процедуру иногда называют *проверкой стойкости.*

Важным подготовительным этапом для проверки стойкости шифра является продумывание различных предполагаемых возможностей, с помощью которых противник может атаковать шифр. Появление таких возможностей у противника обычно не зависит от криптографии, это является некоторой внешней подсказкой и существенно влияет на стойкость шифра. Поэтому оценки стойкости шифра всегда содержат те предположения о целях и возможностях противника, в условиях которых эти оценки получены.

Прежде всего, как это уже отмечалось выше, обычно считается, что противник знает сам шифр и имеет возможности для его предварительного изучения. Противник также знает некоторые характеристики открытых текстов, например, общую тематику сообщений, их стиль, некоторые стандарты, форматы и т. д.

Из более специфических приведем еще три примера возможностей противника:

- противник может перехватывать все шифрованные сообщения, но не имеет соответствующих им открытых текстов;

- противник может перехватывать все шифрованные сообщения и добывать соответствующие им открытые тексты;

- противник имеет доступ к шифру (но не к ключам!) и поэтому может зашифровывать и дешифровывать любую информацию.

На протяжении многих веков среди специалистов не утихали споры о стойкости шифров и о возможности построения абсолютно стойкого шифра. Мы вернемся к этому вопросу позднее.

В заключение данного раздела сделаем еще одно замечание — о терминологии. В последнее время наряду со словом «криптография» часто

встречается и слово «криптология», но соотношение между ними не всегда понимается правильно. Сейчас происходит окончательное формирование этих научных дисциплин, уточняются их предмет и задачи.

Криптология – наука, состоящая из двух ветвей: криптографии и криптоанализа.

Криптография наука о способах преобразования (шифрования) информации с целью ее защиты от незаконных пользователей.

*Криптоанализ – н*аука (и практика ее применения) о методах и способах вскрытия шифров. Соотношение криптографии и криптоанализа очевидно криптография - защита, т.е. разработка шифров, а криптоанализ - нападение, т. е. атака на шифры. Однако эти две дисциплины связаны друг с другом, и не бывает хороших криптографов, не владеющих методами криптоанализа.

1.1.2. Односторонние функции

В 1976 году была опубликована работа молодых американских математиков У.Диффи и М.Э.Хеллмана «Новые направления в криптографии», которая не только существенно изменила криптографию, но и привела к появлению и бурному развитию новых направлений в математике. Центральным понятием «новой криптографии» является понятие односторонней функции. [14, гл. 1]

Односторонней называется функция $F: X \longrightarrow Y$, обладающая двумя свойствами:

а) существует полиномиальный алгоритм вычисления значений $F(x)$;

б) не существует полиномиального алгоритма *инвертирования* функции F (т.е. решения уравнения $F(x) = y$ относительно x).

Отметим, что односторонняя функция существенно отличается от функций, привычных со школьной скамьи, из-за ограничений на сложность ее вычисления и инвертирования. Вопрос о существовании односторонних функций пока открыт.

Еще одним новым понятием является понятие *функции с секретом.* Иногда еще употребляется термин *функция с ловушкой. Функцией с секретом* K называется функция $F_K : X \rightarrow Y$, зависящая от параметра K и обладающая тремя свойствами:

а) существует полиномиальный алгоритм вычисления значения $F_K(x)$для любых K и x;

б) не существует полиномиального алгоритма инвертирования F_K при неизвестном K;

в) существует полиномиальный алгоритм инвертирования F_K при известном K.

Про существование функций с секретом можно сказать то же самое, что сказано про односторонние функции. Для практических целей криптографии было построено несколько функций, которые могут оказаться

функциями с секретом. Для них свойство б) пока строго не доказано, но считается, что задача инвертирования эквивалентна некоторой давно изучаемой трудной математической задаче. Наиболее известной и популярной из них является теоретико-числовая функция, на которой построим шифр RSA.

Применение функций с секретом в криптографии позволяет:

1) организовать обмен шифрованными сообщениями с использованием только открытых каналов связи, т.е. отказаться от секретных каналов связи для предварительного обмена ключами;

2) включить в задачу вскрытия шифра трудную математическую задачу и тем самым повысить обоснованность стойкости шифра;

3) решать новые криптографические задачи, отличные от шифрования *(электронная цифровая подпись* и др.).

Опишем, например, как можно реализовать п. 1). Пользователь А, который хочет получать шифрованные сообщения, должен выбрать какую-нибудь функцию F_K с секретом K. Он сообщает всем заинтересованным (например, публикует) описание функции F_K в качестве своего алгоритма шифрования. Но при этом значение секрета K он никому не сообщает и держит в секрете. Если теперь пользователь B хочет послать пользователю A защищаемую информацию $x \in X$, то он вычисляет $y = F_K(x)$ и посылает y по открытому каналу пользователю A. Поскольку A для своего секрета K умеет инвертировать F_K , то он вычисляет x по полученному y. Никто другой не знает K и поэтому в силу свойства б) функции с секретом не сможет за полиномиальное время по известному шифрованному сообщению $F_K(x)$ вычислить защищаемую информацию x.

Описанную систему называют *криптосистемой с открытым ключом,* поскольку алгоритм шифрования F_K является общедоступным или открытым. В последнее время такие криптосистемы еще называют *асимметричными,* поскольку в них есть асимметрия в алгоритмах: алгоритмы шифрования и дешифрования различны. В отличие от таких систем традиционные шифры называют *симметричными:* в них ключ для шифрования и дешифрования один и тот же. Для асимметричных систем алгоритм шифрования общеизвестен, но восстановить по нему алгоритм дешифрования за полиномиальное время невозможно.

Описанную выше идею Диффи и Хеллман предложили использовать также для электронной цифровой подписи сообщений, которую невозможно подделать за полиномиальное время. 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 $F_K(y) = x$, $F_K: X \longrightarrow Y$ — 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 ее неподделываемости;

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

Кроме принципа построения криптосистемы с открытым ключом, Диффи и Хеллман в той же работе предложили еще одну новую идею — *открытое распределение ключей.* Они задались вопросом: можно ли организовать такую процедуру взаимодействия абонентов *A* и *B* по открытым каналам связи, чтобы решить следующие задачи:

1) вначале *у A и B* нет никакой обшей секретной информации, но в конце процедуры такая общая секретная информация (общий ключ) у *A* и *B* появляется, т. е. вырабатывается;

2) пассивный противник, который перехватывает все передачи информации и знает, что хотят получить *A* и *B,* тем не менее не может восстановить выработанный общий ключ *A и B.*

Диффи и Хеллман предложили решать эти задачи с помощью функции

$$F(x) = a^x \bmod p,$$

где *p* — большое простое число, *x* — произвольное натуральное число, *a* — некоторый *примитивный элемент* поля *GF(p).* Общепризнанно, что инвертирование функции $a^x \bmod p$, т. е. дискретное логарифмирование, является трудной математической задачей.

Сама процедура или, как принято говорить, *протокол выработки общего ключа* описывается следующим образом.

Абоненты *A* и *B* независимо друг от друга случайно выбирают по одному натуральному числу — скажем x_A *и* x_B. Эти элементы они держат в секрете. Далее каждый из них вычисляет новый элемент:

$$y_A = a^{xA} \bmod p,\ y_B = a^{xB} \bmod p.$$

(Числа *p и a* считаются общедоступными.) Потом они обмениваются этими элементами по каналу связи. Теперь абонент *A,* получив y_B и зная свой секретный элемент x_A, вычисляет новый элемент:

$$y_B^{x_A} \bmod p = (a^{xB})^{xA} \bmod p.$$

Аналогично поступает абонент *B:*

$$y_A^{x_B} \bmod p = (a^{xA})^{xB} \bmod p.$$

Тем самым у *A* и *B* появился общий момент поля, равный a^{xAxB}. Этот элемент и объявляется общим ключом *A* и *B***.**

Из описания протокола видно**,** что противник знает *p, a,* a^{xA}, a^{xB}, не знает x^A и x^B и хочет узнать a^{xAxB}. В настоящее время нет алгоритмов действий противника, более эффективных, чем дискретное логарифмирование, а это — трудная математическая задача.

Успехи, достигнутые в разработке схем цифровой подписи и открытого распределения ключей, позволили применить эти идеи также и к другим задачам взаимодействия удаленных абонентов. Так возникло большое новое направление теоретической криптографии — криптографические протоколы.

Объектом изучения теории криптографических протоколов являются удаленные абоненты, взаимодействующие, как правило, по открытым каналам связи. Целью взаимодействия абонентов является решение какой-то задачи. Имеется также противник, который преследует собственные цели. При этом противник в равных задачах может иметь разные возможности: например, может взаимодействовать с абонентами от имени других абонентов или вмешиваться в обмены информацией между абонентами и т. д. Противником может даже оказаться один из абонентов или несколько абонентов, вступивших в сговор.

Приведем еще несколько примеров задач, решаемых удаленными абонентами.

1. Взаимодействуют два не доверяющих друг другу абонента. Они хотят подписать контракт. Это надо сделать так, чтобы не допустить следующую ситуацию: один из абонентов получил подпись другого, а сам не подписался.

Протокол решения этой задачи принято называть *протоколом подписания контракта.*

2. Взаимодействуют два не доверяющих друг другу абонента. Они хотят бросить жребий с помощью монеты. Это надо сделать так, чтобы абонент, подбрасывающий монету, не мог изменить результат подбрасывания после получения догадки от абонента, угадывающего этот результат.

Протокол решения этой задачи принято называть *протоколом подбрасывания монеты.*

Опишем один из простейших протоколов подбрасывания монеты по телефону (так называемая схема Блюма-Микали). Для его реализации у абонентов *A и B* должна быть односторонняя функция f: X —> У, удовлетворяющая следующим условиям:

1) X — множество целых чисел, которое содержит одинаковое количество четных и нечетных чисел;

2) любые числа $x_1, x_2 \in X$, имеющие один образ $f(x_1) = f(x_2)$, имеют одну четность;

3) по заданному образу $f(x)$ «трудно» вычислить четность неизвестного аргумента x.

Роль подбрасывания монеты играет случайный и равновероятный выбор элемента $x \in X$, а роль орла и решки — четность и нечетность x соответственно. Пусть A — абонент, подбрасывающий монету, а B — абонент, угадывающий результат. Протокол состоит из следующих шагов:

1) A выбирает x («подбрасывает монету»), зашифровывает x, т.е. вычисляет $y = f(x)$, и посылает y абоненту B;

2) *B* получает *y,* пытается угадать четность *x* и посылает свою догадку абоненту *A;*

3) *A* получает догадку от *B* и сообщает *B,* угадал ли он, посылая ему выбранное число *x;*

4) *B* проверяет, не обманывает ли *A,* вычисляя значение *f(x)* и сравнивая его с полученным на втором шаге значением *y.*

3. Взаимодействуют два абонента *A* и *B* (типичный пример: *A* — клиент банка, *B* — банк). Абонент *A* хочет доказать абоненту *B,* что он именно *A,* а не противник.

Протокол решения этой задачи принято называть *протоколом идентификации абонента.*

4. Взаимодействуют несколько удаленных абонентов, получивших приказы из одного центра. Часть абонентов, включая центр, могут быть противниками. Необходимо выработать единую стратегию действий, выигрышную для абонентов.

Эту задачу принято называть задачей о византийских генералах, а протокол ее решения — *протоколом византийского соглашения.*

Осмысление различных протоколов и методов их построения привело в 1985-1986 г.г. к появлению двух плодотворных математических моделей — *интерактивной системы доказательства* и *доказательства с нулевым разглашением.* Математические исследования этих новых объектов позволили доказать много утверждений, весьма полезных при разработке криптографических протоколов (подробнее об этом см. главу 2).

Под интерактивной системой доказательства *(P, V, S)* понимают протокол взаимодействия двух абонентов: *P* (доказывающий) и *V* (проверяющий). Абонент *P* хочет доказать *V,* что утверждение S истинно. При этом абонент *V* самостоятельно, без помощи P, не может проверить утверждение S (поэтому *V* и называется проверяющим). Абонент *P* может быть и противником, который хочет доказать *V,* что утверждение S истинно, хотя оно ложно. Протокол может состоять из многих *раундов* обмена сообщениями между *P* и *V* и должен удовлетворять двум условиям:

1) *полнота* - если S действительно истинно, то абонент P абонента *V* признать это;

2) *корректность* — если S ложно, то абонент *P* вряд ли убедит абонента *V,* что *S* истинно.

Здесь словами «вряд ли» мы для простоты заменили точную математическую формулировку.

Подчеркнем, что в определении системы *(P, V, S)* не допускалось, что *V* может быть противником. А если *V* оказался противником, который хочет «выведать» у *P* какую-нибудь новую полезную для себя информацию об утверждении S? В этом случае *P,* естественно, может не хотеть, чтобы это случилось в результате работы протокола *(P, V, S).* Протокол (P, *V, S)*, решающий такую задачу, называется доказательством с нулевым разглашением и должен удовлетворять, кроме условий 1) и 2), еще и следующему условию:

3) *нулевое разглашение* — в результате работы протокола (P, *V,* S) абонент *V* не увеличит свои знания об утверждении *S* или, другими словами, не сможет извлечь никакой информации о том, почему *S* истинно.

1.2. ЗАДАЧИ КРИПТОГРАФИИ

Целью настоящего раздела является определение основных понятий и задач криптографии. При написании данного раздела мы использовали учебное пособие [15].

Современная *криптография* является областью знаний, связанной с решением таких проблем безопасности информации, как конфиденциальность, целостность, аутентификация и невозможность отказа сторон от авторства. Достижение этих требований безопасности информационного взаимодействия и составляет основные цели криптографии. Они определяются следующим образом.

Обеспечение *конфиденциальности* — решение проблемы защиты информации от ознакомления с ее содержанием со стороны лиц, не имеющих права доступа к ней. В зависимости от контекста вместо термина "конфиденциальная" информация могут выступать термины "секретная", "частная", "ограниченного доступа" информация.

Обеспечение *целостности* — гарантирование невозможности несанкционированного изменения информации. Для гарантии целостности необходим простой и надежный критерий обнаружения любых манипуляций с данными. Манипуляции с данными включают вставку, удаление и замену.

Обеспечение *аутентификации* — разработка методов подтверждения подлинности сторон *(идентификация)* и самой информации в процессе информационного взаимодействия. Информация, передаваемая по каналу связи, должна быть аутентифицирована по источнику, времени создания, содержанию данных, времени пересылки и т. д.

Обеспечение *невозможности отказа от авторства* — предотвращение возможности отказа субъектов от некоторых из совершенных ими действий. Рассмотрим средства для достижения этих целей более подробно.

2.1. Проблемы безопасности информации

Конфиденциальность

Традиционной задачей криптографии является проблема обеспечения конфиденциальности информации при передаче сообщений по контролируемому противником каналу связи. В простейшем случае эта задача описывается взаимодействием трех субъектов (сторон). Владелец информации, называемый обычно *отправителем,* осуществляет преобразование исходной *(открытой)* информации (сам процесс преобразования называется *шифрованием)* в форму передаваемых *получателю*по открытому каналу связи *шифрованных* сообщений с целью ее защиты от противника.

Под *противником* понимается любой субъект, не имеющий права ознакомления с содержанием передаваемой информации. В качестве

противника может выступать *криптоаналитик,* владеющий методами раскрытия шифров. Законный получатель информации осуществляет *расшифрование* полученных сообщений. Противник пытается овладеть защищаемой информацией (его действия обычно называют *атаками)*. При этом он может совершать как пассивные, так и активные действия. *Пассивные* атаки связаны с прослушиванием, анализом трафика, перехватом, записью передаваемых шифрованных сообщений, *дешифрованием,* то есть попытками "взломать" защиту с целью овладения информацией.

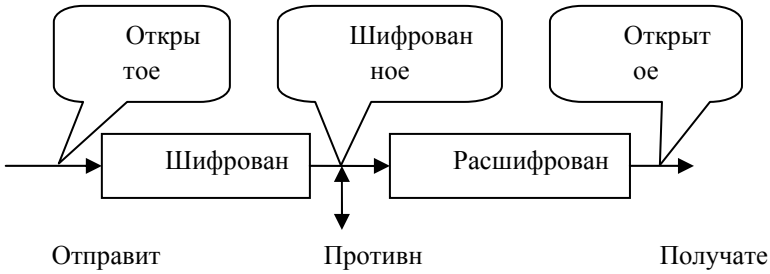


Рис.3. Передача шифрованной информации [15, с.57]

При проведении *активных* атак противник может прерывать процесс передачи сообщений, создавать поддельные (сфабрикованные) или модифицировать передаваемые шифрованные сообщения. Эти активные действия называют попытками *имитации* и *подмены* соответственно.

Под *шифром* обычно понимается семейство обратимых преобразований, каждое из которых определяется некоторым параметром, называемым ключом, а также порядком применения данного преобразования, называемым *режимом шифрования*.

Ключ — это важнейший компонент шифра, отвечающий за выбор преобразования, применяемого для зашифрования конкретного сообщения. Обычно ключ представляет собой некоторую буквенную или числовую последовательность. Эта последовательность как бы "настраивает" алгоритм шифрования.

Каждое преобразование однозначно определяется ключом и описывается некоторым *криптографическим алгоритмом*. Один и тот же криптографический алгоритм может применяться для шифрования в различных режимах. Тем самым реализуются различные способы шифрования (простая замена, гаммирование и т. п.). Каждый режим шифрования имеет как свои преимущества, так и недостатки. Поэтому выбор режима зависит от конкретной ситуации. При расшифровании используется криптографический алгоритм, который в общем случае может отличаться от алгоритма, применяемого для зашифрования сообщения. Соответственно могут различаться ключи зашифрования и расшифрования. Пару алгоритмов зашифрования и расшифрования обычно называют *криптосистемой (шифрсистемой),* а реализующие их устройства — *шифртехникой.*

Если обозначить через *M* открытое, а через *C* шифрованное сообщения, то процессы зашифрования и расшифрования можно записать в виде равенств

$E_{k1}(M)=C$

$D_{k2}(C)=M$

в которых алгоритмы зашифрования *E* и расшифрования *D* должны удовлетворять равенству

$D_{k2}(E_{k1}(M))=M$

Различают *симметричные* и *асимметричные* криптосистемы. В симметричных системах знание ключа зашифрования k_1 позволяет легко найти ключ расшифрования k_2 (в большинстве случаев эти ключи просто совпадают). В асимметричных криптосистемах знание ключа k_1, не позволяет определить ключ k_2. Поэтому для симметричных криптосистем оба ключа должны сохраняться в секрете, а для асимметричных — только один — ключ расшифрования k_2, а ключ k_1 можно сделать открытым (общедоступным). В связи с этим их называют еще *шифрами с открытым ключом.*

Симметричные криптосистемы принято подразделять на *поточные* и *блочные* системы. Поточные системы осуществляют зашифрование отдельных символов открытого сообщения. Блочные же системы производят зашифрование блоков фиксированной длины, составленных из подряд идущих символов сообщения.

Асимметричные криптосистемы, как правило, являются блочными. При их использовании можно легко организовать передачу конфиденциальной информации в сети с большим числом пользователей. В самом деле, для того чтобы послать сообщение, отправитель открыто связывается с получателем, который либо передает свой ключ отправителю, либо помещает его на общедоступный сервер. Отправитель зашифровывает сообщение на открытом ключе получателя и отправляет его получателю. При этом никто, кроме получателя, обладающего ключом расшифрования, не сможет ознакомиться с содержанием передаваемой информации. В результате такая система шифрования с общедоступным ключом позволяет существенно сократить объем хранимой каждым абонентом секретной ключевой информации.

Возможна и другая симметричная ситуация, когда открытый и секретный ключи меняются местами. Предположим, например, что для проведения контроля соблюдения выполнения каждой стороной договора об ограничении испытаний ядерного оружия создаются пункты контроля, которые ведут запись и конфиденциальную передачу сторонам, участвующим в договоре, сейсмологической информации. Поскольку на каждом таком пункте контролируемая сторона одна, а участников договора может быть очень много, то необходимо обеспечить такое шифрование информации, при котором зашифровать сообщение мог бы только один отправитель, а расшифровать мог бы каждый.

Не существует единого шифра, подходящего для всех случаев жизни. Выбор способа шифрования (то есть криптографического алгоритма и режима его использования) зависит от особенностей передаваемой *информации* (ее ценности, объема, способа представления, необходимой скорости передачи и т. д.), а также возможностей владельцев по защите своей

информации (стоимость применяемых технических устройств, удобство использования, надежность функционирования и т. п.). Имеется большое разнообразие *видов* защищаемой информации: текстовая, телефонная, телевизионная, компьютерная и т. д., причем у каждого вида информации имеются свои существенные особенности, которые надо учитывать при выборе способа шифрования. Большое значение имеют объемы и требуемая скорость передачи шифрованной информации, а также помехозащищенность используемого канала связи. Все это существенным образом влияет на выбор криптографического алгоритма и организацию защищенной связи.

Наличие надежного криптографического алгоритма и правильный выбор режима еще не гарантируют владельцу защищенность передаваемой информации. Немаловажную роль играет правильность их использования. Поскольку даже самые стойкие шифры при неправильном использовании существенно теряют свои качества, то конфиденциальность передаваемой информации во многом зависит от того, какие *ошибки* допускает ее владелец при использовании криптографической защиты. А то, что все пользователи допускают ошибки, — неизбежно и является непреложным и важным (для криптоаналитика) фактом, поскольку любые криптографические средства, какими бы они ни были удобными и прозрачными, всегда мешают пользователям в работе, а различные тонкости известны только криптоаналитикам и, как правило, непонятны пользователям этих средств. Более того, в качестве субъектов взаимодействия могут выступать не только люди, но и различные процессы, осуществляющие обработку информации в автоматизированной системе без участия человека. Поэтому защищенность информации в системе существенно зависит от того, насколько правильно там реализована криптографическая подсистема, отвечающая за выполнение криптографических функций. Одно наличие такой подсистемы еще ничего не гарантирует.

Подчеркнем разницу между терминами *"расшифрование"* и *"дешифрование".* При расшифровании действующий ключ считается известным, в то время как при дешифровании ключ неизвестен. Тем самым расшифрование должно осуществляться столь же просто, как и зашифрование; дешифрование представляет собой значительно более сложную задачу. Именно в этом и состоит смысл шифрования.

Для разных шифров задача дешифрования имеет различную сложность. Уровень сложности этой задачи и определяет главное свойство шифра — способность противостоять попыткам противника завладеть защищаемой информацией. В связи с этим говорят о *криптографической стойкости* шифра (или просто *стойкости),* различая более стойкие и менее стойкие шифры. Методы вскрытия шифров разрабатывает наука, носящая название *криптоанализ.* Термин криптоанализ ввел У. Фридман в 1920 г.

Целостность

Наряду с конфиденциальностью не менее важной задачей является обеспечение целостности информации, другими словами, — неизменности ее

в процессе передачи или хранении [см. 15, с. 62-68]. Решение этой задачи предполагает разработку средств, позволяющих обнаруживать не столько случайные искажения (для этой цели вполне подходят методы теории кодирования с обнаружением и исправлением ошибок), сколько целенаправленное навязывание противником ложной информации. Для этого в передаваемую информацию вносится избыточность. Как правило, это достигается добавлением к сообщению некоторой проверочной комбинации, вычисляемой с помощью специального алгоритма и играющей роль контрольной суммы для проверки целостности полученного сообщения. Главное отличие такого метода от методов теории кодирования состоит в том, что алгоритм выработки проверочной комбинации является "криптографическим", то есть зависящим от секретного ключа. Без знания секретного ключа вероятность успешного навязывания противником искаженной или ложной информации мала. Такая вероятность служит мерой *имитостойкости* шифра, то есть способности самого шифра противостоять активным атакам со стороны противника.

Итак, для проверки целостности к сообщению *M* добавляется проверочная комбинация *S,* называемая *кодом аутентификации сообщения* (сокращенно — КАС) или *имитовставкой.* В этом случае по каналу связи передается пара *C = (M, S).* При получении сообщения *M* пользователь вычисляет значение проверочной комбинации и сравнивает его с полученным контрольным значением *S* Несовпадение говорит о том, что данные были изменены.

Как правило, код аутентификации является значением некоторой (зависящей от секретного ключа) криптографической *хеш-функции* от данного сообщения: $h_k(M) = S$ К кодам аутентификации предъявляются определенные требования. К ним относятся:

— невозможность вычисления значения $h_k(M) = S$ для заданного сообщения *M* без знания ключа k,

— невозможность подбора для заданного сообщения *M* с известным значением $h_k(M)=S$ другого сообщения M_1 с известным значением h_k (M_1) = S_1, без знания ключа k.

Первое требование направлено против создания поддельных (сфабрикованных) сообщений при атаках типа *имитация;* второе — против модификации передаваемых сообщений при атаках типа *подмена.*

Аутентификация

Аутентификация — установление подлинности. В общем случае этот термин может относиться ко всем аспектам информационного взаимодействия: сеансу связи, сторонам, передаваемым сообщениям и т. д.

Установление подлинности (то есть проверка и подтверждение) всех аспектов информационного взаимодействия является важной составной частью проблемы обеспечения достоверности получаемой информации. Особенно остро эта проблема стоит в случае не доверяющих друг другу

сторон, когда источником угроз может служить не только третья сторона (противник), но и сторона, с которой осуществляется взаимодействие.

Рассмотрим эти вопросы более подробно.

Применительно к сеансу связи (транзакции) аутентификация означает проверку: целостности соединения, невозможности повторной передачи данных противником и своевременности передачи данных. Для этого, как правило, используют дополнительные параметры, позволяющие "сцепить" передаваемые данные в легко проверяемую последовательность. Это достигается, например, путем вставки в сообщения некоторых специальных чисел или *меток времени.* Они позволяют предотвратить попытки повторной передачи, изменения порядка следования или обратной отсылки части переданных сообщений. При этом такие вставки в передаваемом сообщении необходимо защищать (например, с помощью шифрования) от возможных подделок и искажений.

Применительно к сторонам взаимодействия аутентификация означает проверку одной из сторон того, что взаимодействующая с ней сторона — именно та, за которую она себя выдает. Часто аутентификацию сторон называют также *идентификацией*.

Основным средством для проведения идентификации являются *протоколы идентификации,* позволяющие осуществлять идентификацию (и аутентификацию) каждой из участвующих во взаимодействии и не доверяющих друг другу сторон. Различают *протоколы односторонней* и *взаимной идентификации.*

Протокол — это распределенный алгоритм, определяющий последовательность действий каждой из сторон. В процессе выполнения протокола идентификации каждая из сторон не передает никакой информации о своем секретном ключе, а хранит его у себя и использует для формирования ответных сообщений на запросы, поступающие при выполнении протокола.

Наконец, применительно к самой информации аутентификация означает проверку того, что информация, передаваемая по каналу, является подлинной по содержанию, источнику, времени создания, времени пересылки и т. д.

Проверка подлинности содержания информации сводится, по сути, к проверке ее неизменности (с момента создания) в процессе передачи или хранения, то есть проверке целостности.

Аутентификация источника данных означает подтверждение того, что исходный документ был создан именно заявленным источником.

Заметим, что если стороны доверяют друг другу и обладают общим секретным ключом, то аутентификацию сторон можно обеспечить применением кода аутентификации. Действительно, каждое успешно декодированное получателем сообщение может быть создано только отправителем, так как только он знает их общий секретный ключ. Для не доверяющих друг другу сторон решение подобных задач с использованием общего секретного ключа становится невозможным. Поэтому при

аутентификации источника данных нужен механизм цифровой подписи, который будет рассмотрен ниже.

В целом, аутентификация источника данных выполняет ту же роль, что и протокол идентификации. Отличие заключается только в том, что в первом случае имеется некоторая передаваемая информация, авторство которой требуется установить, а во втором требуется просто установить сторону, с шторой осуществляется взаимодействие.

Цифровая подпись

В некоторых ситуациях, например в силу изменившихся обстоятельств, отдельные лица могут отказаться от ранее принятых обязательств. В связи с этим необходим некоторый механизм, препятствующий подобным попыткам.

Так как в данной ситуации предполагается, что стороны не доверяют друг другу, то использование общего секретного ключа для решения поставленной проблемы становится невозможным. Отправитель может отказаться от факта передачи сообщения, утверждая, что его создал сам получатель (*отказ от авторства).* Получатель легко может модифицировать, подменить или создать новое сообщение, а затем утверждать, что оно получено от отправителя (*приписывание авторства).* Ясно, что в такой ситуации арбитр при решении спора не будет иметь возможность установить истину.

Основным механизмом решения этой проблемы является так называемая *цифровая подпись.*

Хотя цифровая подпись и имеет существенные отличия, связанные с возможностью отделения от документа и независимой передачей, а также возможностью подписывания одной подписью всех копий документа, она во многом аналогична обычной "ручной" подписи.

Схема цифровой подписи включает два алгоритма, один — для вычисления, а второй — для проверки подписи. Вычисление подписи может быть выполнено только автором подписи. Алгоритм проверки должен быть общедоступным, чтобы проверить правильность подписи мог каждый.

Для создания схемы цифровой подписи можно использовать симметричные шифрсистемы. В этом случае подписью может служить само зашифрованное на секретном ключе сообщение. Однако основной недостаток таких подписей состоит в том, что они являются одноразовыми: после каждой проверки секретный ключ становится известным. Единственный выход из этой ситуации в рамках использования симметрич-ных шифрсистем - это введение доверенной третьей стороны, выполняющей функции посредника, которому доверяют обе стороны. В этом случае вся информация пересылается через посредника, он осуществляет перешифрование сообщений с ключа одного из абонентов на ключ другого. Естественно, эта схема является крайне неудобной.

При использовании шифрсистем с открытым ключом возможны два подхода к построению системы цифровой подписи.

Первый подход состоит в преобразовании сообщения в форму, по которой можно восстановить само сообщение и, тем самым, проверить правильность "подписи". В данном случае подписанное сообщение имеет, как правило, ту же длину, что и исходное сообщение. Для создания такого "подписанного сообщения" можно, например, произвести зашифрование исходного сообщения на секретном ключе автора подписи. Тогда каждый может проверить правильность подписи путем расшифрования подписанного сообщения на открытом ключе автора подписи.

При втором подходе подпись вычисляется и передается вместе с исходным сообщением. Вычисление подписи заключается в преобразовании исходного сообщения в некоторую цифровую комбинацию (которая и является подписью). Алгоритм вычисления подписи должен зависеть от секретного ключа пользователя. Это необходимо для того, чтобы воспользоваться подписью мог бы только владелец ключа. В свою очередь, алгоритм проверки правильности подписи должен быть доступен каждому. Поэтому, как правило, этот алгоритм зависит от открытого ключа пользователя. В данном случае длина подписи не зависит от длины подписываемого сообщения.

Одновременно с проблемой цифровой подписи возникла проблема построения бесключевых криптографических *хеш-функций.* Дело в том, что при вычислении цифровой подписи оказывается более удобным осуществить сначала хеширование, то есть свертку текста в некоторую комбинацию фиксированной длины, а затем уже подписывать полученную комбинацию с помощью секретного ключа. При этом функция хеширования, хотя и не зависит от ключа и является открытой, должна быть "криптографической". Имеется в виду свойство *односторонности* этой функции: по значению комбинации-свертки никто не должен иметь возможность подобрать соответствующее сообщение. В настоящее время имеются стандарты на криптографические хеш-функции, утверждаемые независимо от стандартов на криптографические алгоритмы и схемы цифровой подписи,

1.2.2. Управление секретными ключами

Порядок использования криптографической системы определяется системами установки и управления ключами.

Система установки ключей определяет алгоритмы и процедуры генерации, распределения, передачи и проверки ключей.

Система управления ключами определяет порядок использования, смены, хранения и архивирования, резервного копирования и восстановления, замены или изъятия из обращения скомпрометированных, а также уничтожения старых ключей.

Предварительное распределение ключей

Для надежной защиты информации, передаваемой по открытому каналу связи, применяют криптографические средства. Чтобы

воспользоваться ими, необходимо осуществить первоначальный выбор и установку ключей. Для генерации ключей могут применяться различные алгоритмы. Выбранные ключи необходимо как-либо передать взаимодействующим сторонам. Поэтому для первоначального распределения ключей необходим защищенный канал связи. Самый надежный способ первоначального распределения ключей — это личная встреча всех взаимодействующих сторон. Можно использовать также специальных курьеров, которые будут развозить ключи. Однако при большом числе взаимодействующих сторон требуется предварительная рассылка значительного объема ключевой информации и последующее ее хранение. Поэтому на практике применяют специальные *системы предварительного распределения ключей,* предусматривающие распределение и хранение не самих ключей, а некоторой меньшей по объему исходной информации, на основе которой в дальнейшем каждая сторона может вычислить ключ для взаимодействия с другой стороной. Система предварительного распределения ключей включает два алгоритма. С помощью первого алгоритма осуществляется генерация исходной информации. Эта информация включает открытую часть, которая будет передана всем сторонам или помещена на общедоступном сервере, а также секретные части каждой стороны. Второй алгоритм предназначен для вычисления действующего значения ключа для взаимодействия между абонентами по имеющейся у них секретной и общей открытой части исходной ключевой информации.

Система предварительного распределения ключей должна быть *устойчивой,* то есть учитывать возможность раскрытия части ключей при компрометации, обмане или сговоре абонентов, и *гибкой* — допускать возможность быстрого восстановления путем исключения скомпрометированных и подключения новых абонентов.

Пересылка ключей

После того как предварительное распределение ключей произведено, может потребоваться передача ключей для каждого конкретного сеанса взаимодействия. Передача этих ключей может осуществляться с помощью шифрования с использованием ранее полученных ключей. Для передачи зашифрованных ключей по открытому каналу связи между не доверяющими друг другу абонентами требуется решение всего комплекса задач по установлению подлинности различных аспектов взаимодействия, начиная от подлинности субъектов взаимодействия, подлинности передаваемых сообщений, подлинности самого сеанса связи и кончая подтверждением правильности (идентичности) полученных абонентами ключей.

Для централизованного управления пересылкой ключей создаются специальные доверенные центры, выполняющие функции центров распределения или перешифрования ключей. Различие между этими центрами заключается в том, что в первом случае генерация ключей осуществляется в центре распределения, а во втором случае — самими абонентами.

Открытое распределение ключей

Наиболее просто распределение ключей осуществляется в *системах открытого распределения (секретных) ключей.* Для сетей связи с большим числом абонентов традиционные подходы к построению системы распределения ключей оказываются очень неудобными. Диффи и Хеллман впервые показали, как можно решить эту задачу, используя незащищенный канал связи.

В предложенной ими системе открытого распределения ключей каждая из сторон изначально имеет свой секретный параметр. Стороны реализуют определенный протокол взаимодействия по открытому каналу связи. При этом они обмениваются некоторыми сообщениями (образованными с помощью своих секретных параметров) и по результатам этого обмена вычисляют общий секретный связной ключ. В более поздних работах такие протоколы стали называть *протоколами выработки общего ключа,* поскольку изначально ни одна из сторон не имеет ключа и как такового распределения или пересылки ключей в нем не происходит. В исходном виде система Диффи и Хеллмана имела существенные недостатки, связанные с возможностью для третьей стороны по осуществлению активного вхождения в канал связи и проведению полного контроля передаваемой информацией. Однако после небольших модификаций и дополнений их протокол уже позволяет осуществлять не только выработку общего ключа, но и одновременно проверять и подтверждать правильность вычислений, а также проводить взаимную аутентификацию взаимодействующих сторон.

Схема разделения секрета

Еще одной задачей современной криптографии, тесно связанной с проблемой распределения ключей и активно развивающейся в последние годы, является задача построения *схем разделения секрета.* Для многих практически важных приложений, связанных с запуском или активизацией критических процессов или определяющих порядок получения доступа к значимым данным, ответственное лицо должно ввести секретный ключ. Чтобы обезопасить процедуру принятия решения и не отдавать все на волю одного человека, являющегося обладателем ключа, используют метод разделения секрета. Он состоит в назначении определенной группы лиц, которая имеет право принимать решение. Каждый член группы владеет определенной долей секрета (точнее, специально выбранным набором данных), полная совокупность которых позволяет восстановить секретный ключ. При этом схема разделения секрета выбирается с таким условием, что для восстановления секретного ключа требуется обязательное присутствие всех членов группы, так как в случае отсутствия хотя бы одного из участников объединение долей оставшихся членов группы гарантированно не позволяет получить никакой информации о секретном ключе.

Таким образом, *схема разделения секрета* определяется двумя алгоритмами, удовлетворяющими сформулированному выше условию:

первый алгоритм определяет порядок вычисления значений долей по заданному значению секретного ключа, а второй предназначен для восстановления значения секрета по известным долям.

Задачу построения схемы разделения секрета можно обобщить

— либо путем введения так называемой *структуры доступа,* когда решение может приниматься не одной, а несколькими различными группами, причем часть из участников может наделяться правом "вето",

— либо путем добавления механизмов, позволяющих обнаружить обман или сговор участников,

— либо введением специального протокола распределения долей между участниками с подтверждением правильности полученной информации и аутентификацией сторон.

1.3. МАТЕМАТИЧЕСКИЕ ОСНОВЫ КРИПТОГРАФИИ

Большое влияние на развитие криптографии оказали появившиеся в середине XX века работы американского математика Клода Шеннона. В этих работах были заложены основы теории информации, а также был разработан математический аппарат для исследований во многих областях науки, связанных с информацией. Более того, принято считать, что теория информации как наука родилась в 1948 году после публикации работы К. Шеннона «Математическая теория связи».

В своей работе «Теория связи в секретных системах» Клод Шеннон обобщил накопленный до него опыт разработки шифров. Оказалось, что даже в очень сложных шифрах в качестве типичных компонентов можно выделить такие простые шифры как *шифры замены, шифры перестановки* или их сочетания.

Формальные модели простых шифров

Шифр замены является простейшим, наиболее популярным шифром. Типичными примерами являются шифр Цезаря, «цифирная азбука» Петра Великого и «пляшущие человечки» А. Конан Дойла. Как видно из самого названия, шифр замены осуществляет преобразование замены букв или других «частей» открытого текста на аналогичные «части» шифрованного текста. Легко дать математическое описание шифра замены. Пусть X *и* Y – два алфавита (открытого и шифрованного текстов соответственно), состоящие из одинакового числа символов. Пусть также $g: X \longrightarrow Y$ — взаимнооднозначное отображение X в Y. Тогда шифр замены действует так: открытый текст $x_1x_2...x_n$ преобразуется в шифрованный текст $g(x_1)g(x_2)... g(x_n)$.

Шифр перестановки, как видно из названия, осуществляет преобразование перестановки букв в открытом тексте. Типичным примером шифра перестановки является шифр «Сцитала». Обычно открытый текст разбивается на отрезки равной длины и каждый отрезок шифруется независимо. Пусть, например, длина отрезков равна n и σ — взаимнооднозначное отображение множества $\{1,2,..., n\}$ в себя. Тогда шифр перестановки действует так: отрезок открытого текста $x_1...x_n$ преобразуется в отрезок шифрованного текста

Важнейшим для развития криптографии был результат К. Шеннона о существовании и единственности абсолютно стойкого шифра. Единственным таким шифром является какая-нибудь форма так называемой *ленты однократного использования,* в которой открытый текст «объединяется» с полностью случайным ключом такой же длины.

Этот результат был доказан К. Шенноном с помощью разработанного им теоретико-информационного метода исследования шифров. Мы не будем

здесь останавливаться на докозательстве подробно, заинтересованному читателю рекомендуем изучить работу К. Шеннона.

Обсудим особенности строения абсолютно стойкого шифра и возможности его практического использования [13, гл.1]. Типичным и наиболее простым примером реализации абсолютно стойкого шифра является шифр Вернама, который осуществляет побитовое сложение n-битового открытого текста и n-битового ключа:

$$y_i = x_i \oplus k_i,\ i = 1,...,n.$$

Здесь $x_1...x_n$ — открытый текст, $k_1,...,k_n$ — ключ, $y_1...y_n$ — шифрованный текст.

Подчеркнем, что для абсолютной стойкости существенным является каждое из следующих требований к ленте однократного использования:

1) полная случайность (равновероятность) ключа (это, в частности, означает, что ключ нельзя вырабатывать с помощью какого-либо детерминированного устройства);

2) равенство длины ключа и длины открытого текста;

3) однократность использования ключа.

В случае нарушения хотя бы одного из этих условий шифр перестает быть абсолютно стойким и появляются принципиальные возможности для его вскрытия (хотя они могут быть трудно реализуемыми).

Но, оказывается, именно эти условия и делают абсолютно стойкий шифр очень дорогим и непрактичным. Прежде чем пользоваться таким шифром, мы должны обеспечить всех абонентов достаточным запасом случайных ключей и исключить возможность их повторного применения. А это сделать необычайно трудно и дорого.

Как отмечал Д. Кан: «Проблема создания, регистрации, распространения и отмены ключей может показаться не слишком сложной тому, кто не имеет опыта передачи сообщений по каналам военной связи, но в военное время объем передаваемых сообщений ставит в тупик даже профессиональных связистов. За сутки могут быть зашифрованы сотни тысяч слов. Создание миллионов ключевых знаков потребовало бы огромных финансовых издержек и было бы сопряжено с большими затратами времени. Так как каждый текст должен иметь свой собственный, единственный и неповторимый ключ, применение идеальной системы потребовало бы передачи по крайней мере такого количества знаков, которое эквивалентно всему объему передаваемой поенной информации.»

В силу указанных причин абсолютно стойкие шифры применяются только в сетях связи с небольшим объемом передаваемой информации, обычно это сети для передачи особо важной государственной информации.

Теперь уже понятно, что чаще всего для защиты своей информации законные пользователи вынуждены применять неабсолютно стойкие шифры. Такие шифры, по крайней мере теоретически, могут быть вскрыты. Вопрос только в том, хватит ли у противника сил, средств и времени для разработки и реализации соответствующих алгоритмов. Обычно эту мысль выражают

так: противник с неограниченными ресурсами может вскрыть любой неабсолютно стойкий шифр.

Как же должен действовать в этой ситуации законный пользователь, выбирая для себя шифр? Лучше всего, конечно, было бы доказать, что никакой противник не может вскрыть выбранный шифр, скажем, за 10 лет и тем самым получить теоретическую оценку стойкости. К сожалению, математическая теория еще не дает нужных теорем — они относятся к нерешенной *проблеме нижних оценок вычислительной сложности задач.*

Поэтому у пользователя остается единственный путь— получение практических оценок стойкости. Этот путь состоит из следующих этапов:

- понять и четко сформулировать, от какого противника мы собираемся защищать информацию; необходимо уяснить, что именно противник знает или сможет узнать о системе шифра, а также какие силы и средства он сможет применить для его вскрытия;

- мысленно стать в положение противника и пытаться с его позиций атаковать шифр, т.е. разрабатывать различные алгоритмы вскрытия шифр;;. при этом необходимо в максимальной мере обеспечить моделирование сил, средств и возможностей противника;

- наилучший из разработанных алгоритмов использовать для практической оценки стойкости шифра.

Здесь полезно для иллюстрации упомянуть о двух простейших методах вскрытия шифра: случайное угадывание ключа (он срабатывает с маленькой вероятностью, зато имеет маленькую сложность) и перебор всех подряд ключей вплоть до нахождения истинного (он срабатывает всегда, зато имеет очень большую сложность). Отметим также, что не всегда нужна атака на ключ: для некоторых шифров можно сразу, даже не зная ключа, восстанавливать открытый текст по шифрованному.

Формальные модели шифров

Используем источник **[**15, гл.2]. Для того чтобы иметь возможность доказывать в криптографии точные результаты, нужны математические модели основных исследуемых объектов, к которым относятся в первую очередь шифр и открытый текст. Введем сначала алгебраическую модель шифра (шифрсистемы), предложенную, по сути дела, К. Шенноном. С моделями открытых текстов мы познакомимся ниже.

Как мы уже знаем, криптография защищает информацию с помощью *шифрования* — процедуры, использующей некое обратимое преобразование. При этом преобразование открытого текста в шифрованный называется зашифрованием, а обратный процесс - расшифрованием. Шифрование предполагает наличие множества обратимых преобразований. Выбор преобразования из указанного множества для зашифрования данного сообщения осуществляется с помощью ключа. Имеется однозначное соответствие между множеством ключей и множеством преобразований.

Выбор ключа естественным образом определяет функцию (вообще говоря, многозначную), отображающую множество возможных открытых

текстов в множество возможных шифрованных текстов. Способ вычисления значения этой функции для произвольного аргумента будем называть *правилом зашифрования.* Выбранный ключ будем называть *ключом зашифрования.* Требование однозначности расшифрования определяет обратную функцию, отображающую множество возможных (при выбранном ключе) шифрованных текстов в множество возможных открытых текстов. Способ вычисления значения этой функции для произвольного аргумента будем называть *правилом расшифрования.* Ключ, определяющий выбор правила расшифрования, будем называть *ключом расшифрования.*

Формализуем сказанное.

Пусть X,K,Y — конечные множества возможных открытых текстов, ключей и шифрованных текстов соответственно; $E_k : X \longrightarrow Y$ — правило зашифрования на ключе $k \in K$. Множество $\{E_k: k \in K\}$ обозначим через E, а множество $\{E_k (x): x \in X\}$ — через $E_k (X)$. Пусть D_k: $E_k (X) \longrightarrow X$ — правило расшифрования на ключе $k \in K$, и D — это множество $\{D_k: \mathrm{k} \in K\}$.

Здесь и далее мы будем предполагать, что если $k \in K$ представляется в виде $\mathrm{k} = (k_3,k_p)$, где k_3 — ключ зашифрования, а k_p— ключ расшифрования (причем $\mathrm{k}_3 \neq k_p$), то E_k понимается как функция E_{k3} ,а $\mathrm{D_k}$ — как функция D_{kp}.

Определение 1. *Шифром (шифрсистемой) назовем совокупность*

$$\Sigma_A = (X,K,Y,E,D)$$

введенных множеств, для которых выполняются следующие свойства:

1) *для любых $x \in X$ и $k \in K$ выполняется равенство $D_k(E_k(x)) = x$;*

2) $Y = \bigcup_{k \in K} E_k(X)$

Неформально, шифр — это совокупность множеств возможных открытых текстов (то, что шифруется), возможных ключей (то, с помощью чего шифруется), возможных шифр-текстов (то, во что шифруется), правил зашифрования и правил расшифрования.

Отметим, что условие 1) отвечает требованию однозначности расшифрования. Условие 2) означает, что любой элемент $y \in Y$ может быть представлен в виде $E_k (x)$ для подходящих элементов $x \in X$ и $\mathrm{k} \in K$. Отметим также, что в общем случае утверждение "для любых $\mathrm{k} \in K$ и $y \in E_k (X)$ выполняется равенство $E_k (D_k (y)) = y$" является неверным.

Легко проверить, что из условия 1) следует свойство инъективности функции E_k. Другими словами, если $x_1, x_2 \in X$, причем $x_1 \neq x_2$, то при любом $\mathrm{k} \in K$ выполняется неравенство $E_k(x_1) \neq E_k(x_2)$.

По сути дела определение 1 вводит математическую модель, отражающую основные свойства реальных шифров. В силу этого мы будем отождествлять реальный шифр с его моделью Σ_A, которую будем назвать *алгебраической моделью* шифра. Для подавляющего большинства известных шифров несложно составить такую модель, как это будет видно из дальнейшего.

Введем теперь вероятностную модель шифра. Следуя К. Шеннону, определим априорные распределения вероятностей $P(X), P(K)$ на множествах X и K соответственно. Тем самым для любого $x \in X$ определена вероятность

$p_x(x) \in P(X)$ и для любого $k \in K$ — вероятность $p_K(k) \in P(K)$, причем выполняются равенства

$$\sum_{x \in X} p_X(x) = 1 \quad и \quad \sum_{k \in K} p_K(k) = 1.$$

В тех случаях, когда нам требуется знание распределений $P(X)$ и $P(K)$, мы будем пользоваться *вероятностной моделью* Σ_B, состоящей из пяти множеств, связанных условиями 1) и 2) определения 1, и двух вероятностных распределений:

$\Sigma_B = (X, K, Y, E, D, P(X), P(K))$.

Забегая вперед, отметим, что вероятностные характеристики шифров используются лишь в *криптоанализе* — разделе криптографии, посвященном решению задач *вскрытия* (или *взлома)* шифров.

В большинстве случаев множества X и Y представляют собой объединения декартовых степеней некоторых множеств A и B соответственно, так что для некоторых натуральных L и L_1 .

$$X = \bigcup_{i=1}^{L} A^i, \quad Y = \bigcup_{i=1}^{L_1} B^i$$

Множества A и B называют соответственно *алфавитом открытого текста* и *алфавитом шифрованного текста.* Другими словами, открытые и шифрованные тексты записываются привычным образом в виде последовательностей букв.

Введем шифр простой замены в алфавите A .

Определение 2. *Пусть* $X = Y = \bigcup_{i=1}^{L} A^i, \quad K \subseteq S(A), \quad$ *где* $S(A)$ — *симметрическая группа подстановок множества A. Для любого ключа* $k \in K$, *открытого текста* $x = (x_1,...,x_l)$ *и шифрованного текста* $y = (y_1,...,y_l)$ *правила зашифрования и расшифрования шифра простой замены в алфавите A определяются формулами*

$E_k(x) = (k(x_1), ..., k(x_l))$,

$D_k = (k^{-1}(y_1), ..., k^{-1}(y_l))$,

где k^{-1} *— подстановка, обратная к k.*

В более общей ситуации для шифра простой замены $X = \bigcup_{i=1}^{L} A^i, \quad Y = \bigcup_{i=1}^{L} B^i$, причем $|A| = |B|$, а K представляет собой множество всех биекций множества A на множество B. Правила зашифрования и расшифрования определяются для k е K, x е X, y е Y (и обратной к k биекции k^{-1}) формулами.

Определим еще один шифр, называемый *шифром перестановки.*

Определение 3. *Пусть* $X = Y = A^L$ *и пусть* $K \subseteq S_L$, *где* S_L — *симметрическая группа подстановок множества* {1,2,...,L} . *Для любого ключа k, открытого текста* $x = (x_1,...,x_L)$ *и шифрованного текста* $y = (y_1,...,y_L)$ *правила зашифрования и расшифрования шифра перестановки определяются формулами*

$$E_k(x) = (x_{k(1)}, ..., x_{k(L)}), \quad D_k(y) = (y_{k^{-1}(1)}, ..., y_{k^{-1}(L)}),$$

где k^{-1} *— подстановка, обратная к k.*

Шифры, введенные определениями 2 и 3, являются представителями двух наиболее важных классов симметричных шифров, а именно *шифров замены* и *шифров перестановки,* которые будут подробно рассматриваться ниже. Другими симметричными шифрами являются композиции (или последовательные применения) некоторых шифров замены и шифров перестановки.

Приведем пример асимметричного шифра. В следующем определении *шифра RSA* мы будем пользоваться общепринятыми в алгебре терминологией и обозначениями.

Определение 4. *Пусть n=pq, где p и q — простые числа. Пусть* $X = Y = Z_n$ *— кольцо вычетов по модулю n. Положим* $K = \{(n, p, q, a, b): a, b \in Z_n, n=pq, ab \equiv 1(\mathrm{mod}\,\varphi(n))\}$,

где φ *— функция Эйлера. Представим ключ* $k \in K$ *в виде* $k = (k_з, k_р)$, *где* $k_з=(n,b)$ *и* $k_р=(n,p,q,a)$ *ключи зашифрования и расшифрования соответственно. Правила зашифрования и расшифрования шифра RSA определим для* $x \in X$ *и* $y \in Y$ *формулами*

$$E_{kз}(x) = x^b \bmod n,\ D_{kр}(y) = y^a \bmod n.$$

Аббревиатура RSA определяется начальными буквами фамилий создателей этого шифра — Rivest, Shamir, Adleman. Корректность формул (2) следует из малой теоремы Ферма (подробнее это сделано в § 11.1).

Введенные определения и термины не исчерпывают полный перечень необходимых нам понятий, которые будут вводиться далее по необходимости.

1.3.2 Модели открытых текстов

Введенная нами математическая модель шифра Σ_B содержит вероятностные распределения *P(X)* и *P(K)* на множествах открытых текстов и ключей соответственно. Если *P(K)* определяется свойствами устройств, служащих для генерации ключей (которые могут быть случайными или псевдослучайными), то *P(X)* определяется частотными характеристиками самих текстов, подлежащих шифрованию. Характер таких текстов может быть различный: это могут быть обычные литературные тексты, формализованные данные межмашинного обмена и т. д. Так или иначе, открытые тексты обладают многими закономерностями, некоторые из которых наследуются шифрованными текстами. Именно это является определяющим фактором, влияющим на надежность шифрования.

Математические модели открытого текста

Вернемся к источнику [15]. Потребность в математических моделях открытого текста продиктована, прежде всего, следующими соображениями. Во-первых, даже при отсутствии ограничений на временные и материальные затраты по выявлению закономерностей, имеющих место в открытых текстах, нельзя гарантировать того, что такие свойства указаны с

достаточной полнотой. Например, хорошо известно, что частотные свойства текстов в значительной степени зависят от их характера. Поэтому при математических исследованиях свойств шифров прибегают к упрощающему моделированию, в частности, реальный открытый текст заменяется его моделью, отражающей наиболее важные его свойства. Во-вторых, при автоматизации методов криптоанализа, связанных с перебором ключей, требуется "научить" ЭВМ отличать открытый текст от случайной последовательности знаков. Ясно, что соответствующий критерий может выявить лишь адекватность последовательности знаков некоторой модели открытого текста.

Один из естественных подходов к моделированию открытых текстов связан с учетом их частотных характеристик, приближения для которых можно вычислить с нужной точностью, исследуя тексты достаточной длины. Основанием для такого подхода является устойчивость частот *к* -грамм или целых словоформ реальных языков человеческого общения (то есть отдельных букв, слогов, слов и некоторых словосочетаний). Основанием для построения модели может служить также и теоретико-информационный подход, развитый в работах К. Шеннона.

Учет частот k-грамм приводит к следующей модели открытого текста. Пусть $P^{(k)}(A)$ представляет собой массив, состоящий из приближений для вероятностей $p(b_1,b_2,...,b_k)$ появления k-грамм $b_1b_2...b_k$ в открытом тексте, $k \in N$,

$A = (a_1,...,a_n)$ — алфавит открытого текста, $b_i \in A$, $i = 1,k$.

Тогда источник "открытого текста" генерирует последовательность $c_1,c_2,...,c_k,c_{k+1},...$ знаков алфавита A, в которой k-грамма $c_1c_2...c_k$ появляется с вероятностью $p(c_1c_2...c_k) \in P^{(k)}(A)$, следующая k-грамма $c_1c_2...c_{k+1}$ появляется с вероятность $p(c_2c_3...c_{k+1}) \in P^{(k)}(A)$ и т. д. Назовем построенную модель открытого текста *вероятностной моделью k-го приближения.*

Таким образом, простейшая модель открытого текста -*вероятностная модель первого приближения* – представляет собой последовательность знаков $c_1,c_2,...$, в которой каждый знак ci, $i = 1,2,...$, появляется с вероятностью $p(c_i) \in P^{(1)}(A)$, независимо от других знаков. Будем называть также эту модель *позначной моделью открытого текста.* В такой модели открытый текст $c_1c_2...c_1$ имеет вероятность

$$p(c_1c_2...c_l) = \prod_{i=1}^{l} p(c_i).$$

В вероятностной модели второго приближения первый знак c_1 имеет вероятность $p(c_1) \in P^{(1)}(A)$, а каждый следующий знак c_i зависит от предыдущего и появляется с вероятностью

$$p(c_i / c_{i-1}) = \frac{p(c_{i-1}c)}{p(c_{i-1})},$$

где $p(c_{i-1}c_i) \in P^{(2)}(A)$, $p(c_{i-1}) \in P^{(1)}(A)$, $i = 2,3,....$ Другими словами, модель открытого текста второго приближения представляет собой *простую*

однородную цепь Маркова. В такой модели открытый текст $c_1c_2...c_l$ имеет вероятность

$$p(c_1c_2...c_l) = p(c_1)\cdot\prod_{i=2}^{l} p(c_i / c_{i-1}).$$

Модели открытого текста более высоких приближений учитывают зависимость каждого знака от большего числа предыдущих знаков. Ясно, что чем выше степень приближения, тем более "читаемыми" являются соответствующие модели. Проводились эксперименты по моделированию открытых текстов с помощью ЭВМ.

Отметим, что с более общих позиций открытый текст может рассматриваться как реализация *стационарного эргодического случайного процесса с дискретным временем и конечным числом состояний*.

Критерии распознавания открытого текста

Заменив реальный открытый текст его моделью, мы можем теперь построить критерий распознавания открытого текста. При этом можно воспользоваться либо стандартными методами различения статистических гипотез, либо наличием в открытых текстах некоторых запретов, таких, например, как биграмма ЪЪ в русском тексте. Проиллюстрируем первый подход при распознавании позначной модели открытого текста.

Итак, согласно нашей договоренности, открытый текст представляет собой реализацию независимых испытаний случайной величины, значениями которой являются буквы алфавита $A = \{a_1,...,a_n\}$, появляющиеся в соответствии с распределением вероятностей $P^{(1)}(A) = (p(a_1),...,p(a_n))$. Требуется ' определить, является ли случайная последовательность $c_1c_2...c_l$ букв алфавита A открытым текстом или нет.

Пусть H_0 — гипотеза, состоящая в том, что данная последовательность — открытый текст, H_1 — альтернативная гипотеза. В простейшем случае последовательность $c_1c_2...c_l$ можно рассматривать при гипотезе H_1 как случайную и равновероятную. Эта альтернатива отвечает субъективному представлению о том, что при расшифровании криптограммы с помощью ложного ключа получается "бессмысленная" последовательность знаков. В более общем случае можно считать, что при гипотезе H1 последовательность $c_1c_2...c_l$ представляет собой реализацию независимых испытаний некоторой случайной величины, значениями которой являются буквы алфавита $A = \{a_1,...,a_n\}$, появляющиеся в соответствии с распределением вероятностей $Q^{(1)}(A) = (q(a_l),...,q(a_n))$. При таких договоренностях можно применить, например, *наиболее мощный критерий* различения двух простых гипотез, который дает *лемма Неймана—Пирсона*.

В силу своего вероятностного характера такой критерий может совершать ошибки двух родов. Критерий может принять открытый текст за случайный набор знаков. Такая ошибка обычно называется *ошибкой первого рода,* ее вероятность равна $\alpha = p\{H_1/H_0\}$. Аналогично вводится *ошибка второго рода* и ее вероятность $\beta = p\{H_0/H_1\}$. Эти ошибки определяют качество работы критерия. В криптографических исследованиях естественно

минимизировать вероятность ошибки первого рода, чтобы не "пропустить" открытый текст. Лемма Неймана—Пирсона при заданной вероятности первого рода минимизирует также вероятность ошибки второго рода.

Критерии на открытый текст, использующие запретные сочетания знаков, например *к* -граммы подряд идущих букв, будем называть *критериями запретных k-грамм.* Они устроены чрезвычайно просто. Отбирается некоторое число *s* редких *k*-грамм, которые объявляются запретными. Теперь, просматривая последовательно *k*-грамму за *k*-граммой анализируемой последовательности $c_1c_2...c_l$, мы объявляем ее случайной, как только в ней встретится одна из запретных *k*-грамм, и открытым текстом в противном случае. Такие критерии также могут совершать ошибки в принятии решения. В простейших случаях их можно рассчитать. Несмотря на свою простоту, критерии запретных *k*-грамм являются весьма эффективными.

1.3.3. Классификация шифров по различным признакам

В качестве первичного признака, по которому производится классификация шифров, используется тип преобразования, осуществляемого с открытым текстом при шифровании. Если фрагменты открытого текста (отдельные буквы или группы букв) заменяются некоторыми их эквивалентами в шифртексте, то соответствующий шифр относится к классу *шифров замены.* Если буквы открытого текста при шифровании лишь меняются местами друг с другом, то мы имеем дело с *шифром перестановки.*

Шифры перестановки*,* или транспозиции*,* изменяют только порядок следования символов или других элементов исходного текста. Классическим примером такого шифра является система, использующая карточку с отверстиями - **решетку Кардано***,* которая при наложении на лист бумаги оставляет открытыми лишь некоторые его части. При зашифровке буквы сообщения вписываются в эти отверстия. При расшифровке сообщение вписывается в диаграмму нужных размеров, затем накладывается решетка, после чего на виду оказываются только буквы открытого текста.

Решетки можно использовать двумя различными способами. В первом случае зашифрованный текст состоит только из букв исходного сообщения. Решетка изготавливается таким образом, чтобы при ее последовательном использовании в различных положениях каждая клетка лежащего под ней листа бумаги оказалась занятой. Примером такой решетки является **поворотная решетка,** показанная на рис.5. Если такую решетку последовательно поворачивать на 90° после заполнения всех открытых при данном положении клеток, то при возврате решетки в исходное положение все клетки окажутся заполненными. Числа, стоящие в клетках, облегчают изготовление решетки. В каждом из концентрических окаймлений должна быть вырезана только одна клетка из тех, которые имеют одинаковый номер. Второй, стеганографический метод использования решетки позволяет скрыть факт передачи секретного сообщения. В этом случае заполняется только

часть листа бумаги, лежащего под решеткой, после чего буквы или слова исходного текста окружаются ложным текстом.

1	2	3	4	5	1
5	1	2	3	1	2
4	3	1	1	2	3
3	2	1	1	3	4
2	1	3	2	1	5
1	5	4	3	2	1

Рис.4. Пример поворотной решетки

Рассмотрим усложненную перестановку по таблице. Пример таблицы для реализации этого метода шифрования показан на рис.4. Таблица представляет собой матрицу размерностью 6 х 6, в которую построчно вписывается искомое сообщение. При считывании информации по столбцам в соответствии с последовательностью чисел ключа получается шифротекст. Усложнение заключается в том, что некоторые ячейки таблицы не используются. При зашифровании сообщения

КОМАНДОВАТЬ ПАРАДОМ БУДУ Я

получим:

ОЬБНАОДКДМУМВ АУ ОТР ААПДЯ,

Ключ					
2	4	0	3	5	1
К	О		М	А	Н
Д		О	В	А	
	Т	Ь		П	А
	Р		А	Д	О
М		Б	У		Д
У				Я	

Рис.5. Пример шифрования методом усложненной перестановки по таблице

При расшифровании буквы шифротекста записываются по столбцам в соответствии с последовательностью чисел ключа, после чего исходный текст считывается по строкам. Для удобства запоминания ключа применяют перестановку столбцов таблицы по ключевому слову или фразе, всем символам которых ставятся в соответствие номера, определяемые порядком соответствующих букв в алфавите. Например, при выборе в качестве ключа

слова ИНГОДА последовательность использования столбцов будет иметь вид 462531.

Математическая модель шифра замены

Воспользуемся моделью $\Sigma_A = (X,K,Y,E,D)$ произвольного шифра замены, приведенной в [15, с.87]. Будем считать, что открытые и шифрованные тексты являются словами в алфавитах A и B соответственно: $X \subset A^*$, $Y \subset B^*$, $|A|=n$, $|B| = m$. Здесь и далее C* обозначает множество слов конечной длины в алфавите C.

Перед зашифрованием открытый текст предварительно представляется в виде последовательности подслов, называемых *шифрвеличинами.* При зашифровании шифрвеличины заменяются некоторыми их эквивалентами в шифртексте, которые назовем *шифробозначениями.* Как шифрвеличины, так и шифробозначения представляют собой слова из A^* и B^* соответственно.

Пусть $U = \{u_1,..,u_N\}$ — множество возможных шифрвеличин, $V = \{v_1,...,v_M\}$ — множество возможных шифробозначений. Эти множества должны быть такими, чтобы любые тексты $x \in X$, $y \in Y$ можно было представить словами из U^*, V^* соответственно. Требование однозначности расшифрования влечет неравенства $N \geq n$, $M \geq m$, $M \geq N$. Для определения правила зашифрования $E_k(x)$ в общем случае нам понадобится ряд обозначений и понятие *распределителя,* который, по сути, и будет выбирать в каждом такте шифрования замену соответствующей шифрвеличине.

Поскольку $M \geq N$, множество V можно представить в виде объединения $V = \bigcup_{i=1}^{N} V_\alpha^{(i)}$ непересекающихся непустых подмножеств $V^{(i)}$. Рассмотрим произвольное семейство, состоящее из r таких разбиений множества V :

$$V = \bigcup_{i=1}^{N} V_\alpha^{(i)},\ \alpha = \overline{1,r},\ r \in N,$$

и соответствующее семейство биекций

$$\varphi_\alpha : U \to \{V_\alpha^{(1)},...,V_\alpha^{N}\},$$

для которых $\varphi_\alpha(u_i) = V_\alpha^{(i)},\ i = \overline{1,N}$.

Рассмотрим также произвольное отображение $\psi : K \times N \to N_r^*$, где $N_r = \{1,2,...,r\}$, такое, что для любых $k \in K, l \in N$

$$\psi(k,l) = \alpha_1^{(k)}...\alpha_l^{(k)},\ \alpha_j^{(k)} \in N_r,\quad j = \overline{1,l}.$$

Назовем последовательность $\psi(\kappa,l)$ *распределителем,* отвечающим данным значениям $\kappa \in K$, $l \in$ N.

Теперь мы сможем определить правило зашифрования произвольного шифра замены. Пусть

$$x \in X,\ x = x_1...x_l,\ x_i \in U,\ i = 1,l;\ k \in K$$

и $\psi(\kappa,I) = a_1^{(k)}...a_1^{(k)}$. Тогда $E_\kappa(x) = y$, где $y = y_1...y_l$,

$y_j = \varphi_{\alpha_j^{(k)}}(x),\ j = \overline{1,l}.$

В качестве y_j можно выбрать любой элемент множества m $\varphi_{\alpha_j^{(k)}}(x_j)$. Всякий раз при шифровании этот выбор можно производить случайно, например, с помощью некоторого *рандомизатора* типа игровой рулетки. Подчеркнем, что такая многозначность при зашифровании не препятствует расшифрованию, так как $V_\alpha^{(i)} \bigcap V_\alpha^{(j)} = \emptyset$ при $i \neq j$.

Классификация шифров замены

Используем классификацию, приведенную в [15, гл.3]. Если ключ зашифрования совпадает с ключом расшифрования: $k_3 = k_p$, то такие шифры называют *симметричными,* если же $k_3 \neq k_p$ — *асимметричными.*

В связи с указанным различием в использовании ключей сделаем еще один шаг в классификации.

Отметим также, что в приведенном определении правило зашифрования $E_k(x)$ является, вообще говоря, *многозначной функцией.* Выбор ее значений представляет собой некоторую проблему, которая делает многозначные функции $E_k(x)$ не слишком удобными для использования. Избавиться от этой проблемы позволяет использование однозначных функций, что приводит к естественному разделению всех шифров замены на *однозначные* и *многозначные замены* (называемых также в литературе *омофонами).*

Для однозначных шифров замены справедливо свойство:

$$\forall \alpha, i : \left| V_\alpha^{(i)} \right| = 1;$$

для многозначных шифров замены:

$$\exists \alpha, i : \left| V_\alpha^{(i)} \right| > 1;$$

Исторически известный шифр — *пропорциональной замены* представляет собой пример шифра многозначной замены, *шифр гаммирования* - пример шифра однозначной замены. Далее мы будем заниматься в основном изучением однозначных замен, получивших наибольшее практическое применение. Итак, далее $M = N$ и $\varphi_\alpha(u_i) = v_{\alpha,i}$, $i = \overline{1,M}$.

Заметим, что правило зашифрования E_k естественным образом индуцирует отображение $\widetilde{E}_k : U \to V$, которое в свою очередь продолжается до отображения $\widetilde{E}_k : U^* \to V^*$. Для упрощения записи будем использовать одно обозначение E_k для каждого из трех указанных отображений.

В силу инъективности (по k) отображения E_k и того, что $|U| = |V|$, введенные в общем случае отображения φ_α являются биекциями $\varphi_\alpha : U \leftrightarrow V$, определенными равенствами $\varphi_\alpha(u_i) = v_\alpha^{(i)}$, $i = \overline{1,N}$, $\alpha = \overline{1,r}$.. Число таких биекций не превосходит $N!$.

Для шифра однозначной замены определение правила зашифрования можно уточнить: в формуле включение следует заменить равенством

$$y_j = \varphi_{\alpha_j^{(k)}}(x_j), \quad j = \overline{1,l}.$$

Введем еще ряд определений.

Если для некоторого числа $q \in \mathrm{N}$ выполняются включения $v_i \in B^q$, $i=1,N$, то соответствующий шифр замены будем называть *шифром равнозначной замены.* В противном случае — *шифром разнозначной замены.*

В подавляющем большинстве случаев используются шифры замены, для которых $U \in A^p$, для некоторого $p \in \mathrm{N}$. При $p = 1$ говорят о *поточных шифрах замены,* при $p > 1$ — о *блочных шифрах замены.*

Следующее определение. В случае r = 1 шифр замены называют *одноалфавитным шифром замены* или *шифром простой замены.* В противном случае – *многоалфавитным шифром замены.*

Приведем примеры. Вскрытие одноалфавитных шифров основано на учете частоты появления отдельных букв или их сочетаний (биграмм, триграмм и т. п.) в данном языке. Классические примеры вскрытия таких шифров содержатся в рассказах Э. По "Золотой жук" и А.Конан Дойля "Пляшущие человечки".

Примером многоалфавитного шифра замены является так называемая система Виженера. Шифрование осуществляется по таблице, представляющей собой квадратную матрицу размерностью n X n, где n - число символов используемого алфавита. Таблица Виженера для русского языка (алфавит Z_{32}- 32 буквы и пробел). Первая строка содержит все символы алфавита. Каждая следующая строка получается из предыдущей циклическим сдвигом последней на символ влево.

Выбирается ключ или ключевая фраза. После чего процесс зашифрования осуществляется следующим образом. Под каждой буквой исходного сообщения последовательно записываются буквы ключа; если ключ оказался короче сообщения, его используют несколько раз. Каждая буква шифротекста находится на пересечении столбца таблицы, определяемого буквой открытого текста, и строки, определяемой буквой ключа. Пусть, например, требуется зашифровать сообщение:

ГРУЗИТЕ АПЕЛЬСИНЫ БОЧКАМИ ТЧК БРАТЬЯ КАРАМАЗОВЫ ТЧК

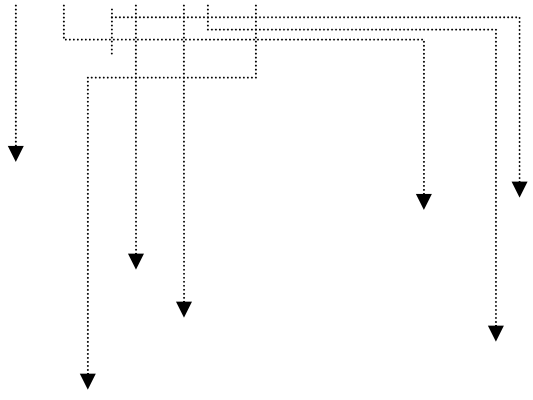
С помощью ключа ВЕНТИЛЬ. Запишем строку исходного текста с расположенной под ней строкой с циклически повторяемым ключом:

ГРУЗИТЕ АПЕЛЬСИНЫ БОЧКАМИ ТЧК БРАТЬЯ КАРАМАЗОВЫ ТЧК
ВЕНТИЛЬВЕНТИЛЬВЕНТИЛЬВЕНТИЛЬВЕНТИЛЬВЕНТИЛЬВЕНТИЛЬВЕ

В результате зашифрования, начальный этап которого показан на рисунке 11, получим шифротекст

ЕХ ЩРЭАБЕЫЧУДККТИСЙЩРМЕЩЬЗЭРМДОБИЭУАДЧТШЛЕВМЪФГКЛЩП

Г Р У З И Т Е А П Е Л Ь С И Н Ы Б О Ч К А М И



А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
В	Г	Д	**Е**	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С
Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф
Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Э	Ы	Ь
Т	У	Ф	Х	Ц	Ч	Ш	**Щ**	Ъ	Ы	Ь	Э	Ю	Я		А
И	Й	К	Л	М	Н	О	П	**Р**	С	Т	У	Ф	Х	Ц	Ч
Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ
Ь	Э	Ю	Я		**А**	Б	В	Г	Д	Е	Ж	З	И	Й	К
	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О

Рис.11. Принцип шифрования по таблице Виженера

Расшифрование осуществляется следующим образом. Под буквами шифротекста последовательно записываются буквы ключа; в строке таблицы, соответствующей очередной букве ключа, происходит поиск соответствующей буквы шифротекста. Находящаяся над ней в первой строке таблицы буква является соответствующей буквой исходного текста.

Для увеличения надежности шифра можно рекомендовать его использование после предварительной псевдослучайной перестановки букв в каждой строке таблицы. Возможны и другие модификации метода.

Ограничиваясь наиболее важными классами шифров замены и исторически известными классами шифров перестановки, сведем результаты классификации в схему, изображенную на рисунке.

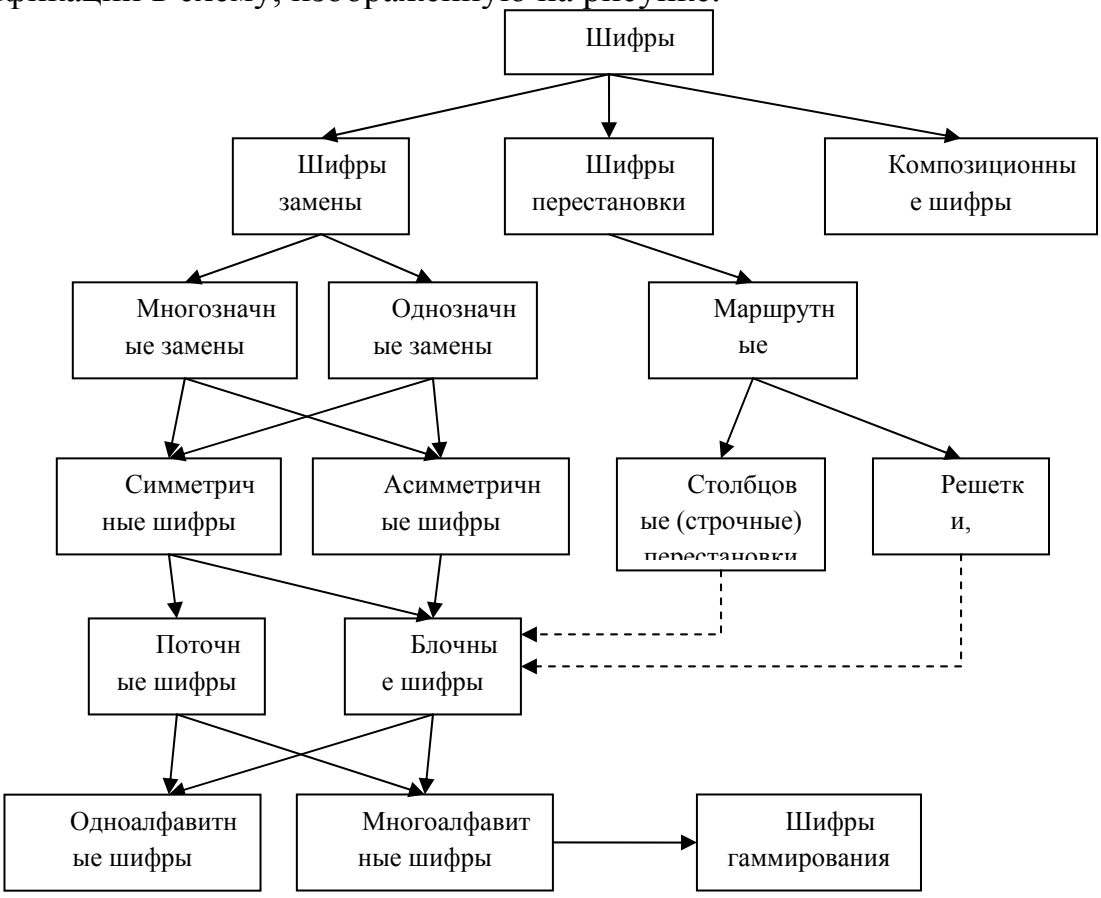


Рис. 12. Классификация шифров

Пунктирные стрелки, ведущие из подклассов шифров перестановки, означают, что эти шифры можно рассматривать и как блочные шифры замены в соответствии с тем, что открытый текст делится при шифровании на блоки фиксированной длины, в каждом из которых производится некоторая перестановка букв. Одноалфавитные и многоалфавитные шифры могут быть как поточными, так и блочными. В то же время шифры гаммирования, образующие подкласс многоалфавитных шифров, относятся к поточным, а не к блочным шифрам. Кроме того, они являются симметричными, а не асимметричными шифрами.

С целью повышения надежности шифрования шифрованный текст, полученный применением некоторого шифра, может быть еще раз зашифрован с помощью другого шифра. Всевозможные такие композиции различных шифров приводят к третьему классу шифров, которые обычно называют *композиционными шифрами.* Заметим, что композиционный шифр может не входить ни в класс шифров замены, ни в класс шифров перестановки. В результате получаем первый уровень классификации шифров:

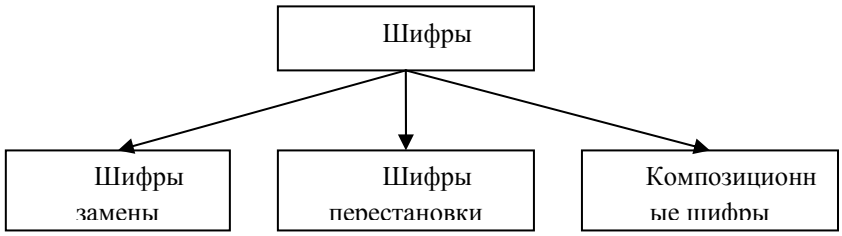


Рис. 13. Классификация простых шифров

Идея, лежащая в основе составных, или композиционных, блочных шифров, состоит в построении криптостойкой системы путем многократного применения относительно простых криптографических преобразований, в качестве которых К. Шеннон предложил использовать преобразования подстановки ,substitution) и перестановки (permutation); схемы, реализующие эти преобразования, называются SP-сетями.

Многократное использование этих преобразований (рис.13) позволяет обеспечить два свойства, которые должны быть присущи стойким шифрам: рассеивание (diffusion) и перемешивание (confusion). Рассеивание предполагает распространение влияния одного знака открытого текста, и также одного знака ключа на значительное количество знаков шифротекста. Наличие у шифра этого свойства:

- позволяет скрыть статистическую зависимость между знаками открытого текста, иначе говоря, перераспределить избыточность исходного языка посредством распространения ее на весь текст;
- не позволяет восстанавливать неизвестный ключ по частям.

Например, обычная перестановка символов позволяет скрыть частоты появления биграмм, триграмм и т. д.

Цель перемешивания - сделать как можно более сложной зависимость между ключом и шифротекстом. Криптоаналитик на основе статистического анализа перемешанного текста не должен получить сколько-нибудь

значительного количества информации об использованном ключе. Обычно перемешивание осуществляется при помощи подстановок. Как будет видно ниже, применение к каждому элементу открытого текста своей собственной подстановки приводит к появлению абсолютно стойкого шифра. Применение рассеивания и перемешивания порознь не обеспечивает необходимую стойкость (за исключением вышеупомянутого предельного случая), стойкая криптосистема получается только в результате их совместного использования.

В современных блочных криптосистемах раундовые шифры строятся в основном с использованием операций замены двоичных кодов небольшой разрядности (схемы, реализующие эту нелинейную операцию, называются S-блоками*;* как правило, именно от их свойств в первую очередь зависит стойкость всей системы), перестановки элементов двоичных кодов, арифметических и логических операций над двоичными кодами.

Важным достоинством многих составных шифров является их симметричность относительно операций зашифрования и расшифрования, которые по этой причине могут быть реализованы на одном устройстве. Переход от одного режима к другому обеспечивается заменой последовательности раундовых ключей на обратную.

Составные шифры, использующие в качестве раундовых криптографически слабые преобразования, становятся нестойкими, если становятся известными какие-либо промежуточные результаты преобразований. По этой причине использование этой информации при криптоанализе составных шифров является некорректным.

Представим шифруемый блок данных (открытого $\tilde{p}_i$ или закрытого c_i текста)

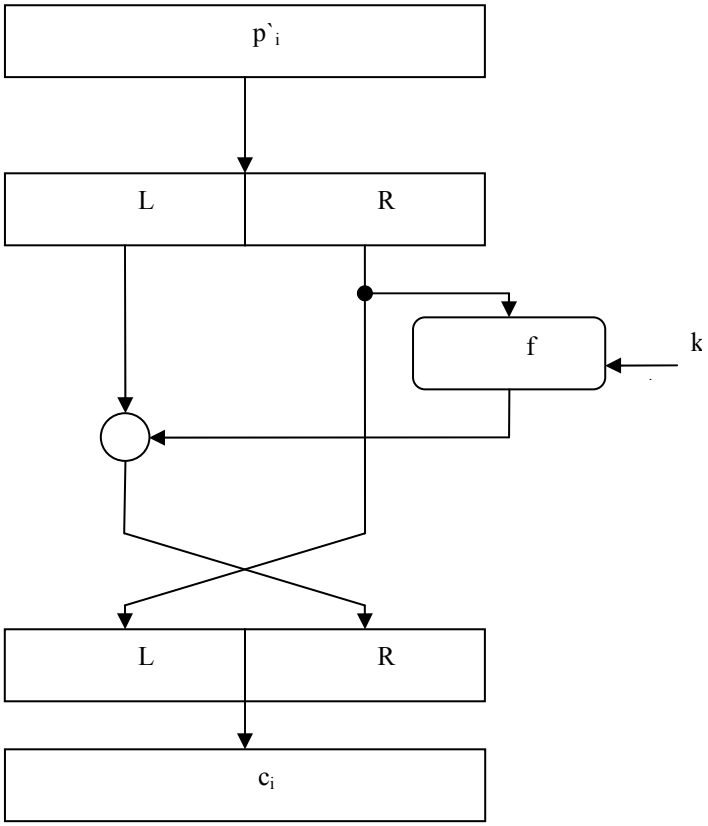
длиной n ($|\tilde{p}_i| = (|c_i| = n)$ в виде пары полублоков в 2 раза меньшего размера

$\tilde{p}_i = c_i, = (L, R), |L| = |R| = n/2$.

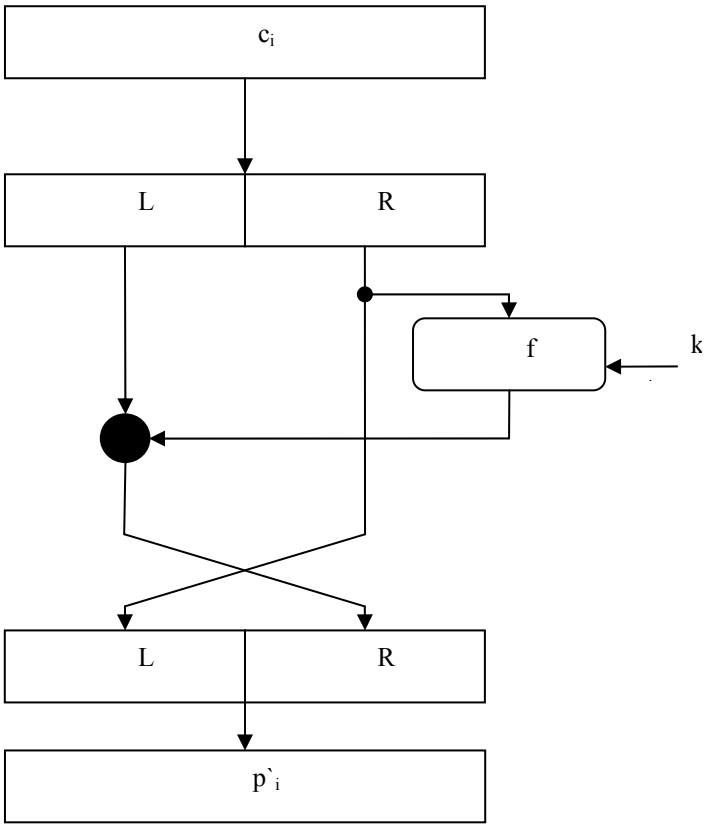
Выполним зашифрование старшего полублока L (Left) блока $\tilde{p}_i$ с помощью младшего R (Right), используя некоторую функцию f_i зависящую от раундового ключа k_t, и обратимую бинарную операцию ° над n/2-битовыми блоками данных. Для подготовки к следующему аналогичному раунду выполним перестановку частей блока $\tilde{p}_i$: $L°f_i(R) \leftrightarrow R$. Таким образом, раундовая функция зашифрования будет иметь вид $F_i(\tilde{p}_i) = F_t(L, R) = (R, L°f_i(R))$ (рис.14), для которой легко построить обратное, или расшифровывающее, преобразование $F_i^{-1}(c)$:

$F_i^{-1}(c_i) = F_i^{-1}(L, R) = (R, L \bullet f_i(R))$,

где • - операция, обратная **o**. Композиционный шифр, использующий раундовые функции такого вида, называется шифром Фейстеля [16, 17]. В подавляющем большинстве шифров рассматриваемой структуры используется разрядность блока, равная 64 битам, а в качестве операций **o** и • - поразрядное сложение по модулю 2 (XOR).



а



б

Рис.14. Схема петли Фейстеля:
а – зашифрование; *б* – расшифрование

Первыми широко известными практическими реализациями

итерационного блочного шифра были разработанные Х. Фейстелем, Д. Копперсмитом и другими сотрудниками фирмы IBM криптоалгоритмы Lucifer и созданный на его основе в 1974 г. в качестве стандарта шифрования данных в государственных и частных организациях DES (Data Encryption Standard). DES работает с блоками данных разрядностью 64 бита с применением 56-рапрядного ключа, из которого по специальному фиксированному алгоритму, использующему перестановки и сдвиги, вырабатываются раундовые ключи). Применяемые преобразования- поразрядное сложение по модулю 2, подстановки и перестановки; число раундов равно 16; перед началом первого раунда выполняется начальная фиксированная перестановка IP, после 16-го раунда выполняется обратная перестановка IP^{-1}. Следуя рекомендациям Шеннона, в каждом раунде выполняется один шаг перемешивания (с использованием соответствующего раундового ключа и S-блоков), после которого следует шаг рассеивания, не зависящий от ключа.

Интересно отметить, что в первоначальной схеме, предложенной IBM, все шестнадцать 48-разрядных раундовых ключей выбирались независимо, т. е. размер ключа был равен 768 битам. Однако по требованию Агентства национальной безопасности США (АНБ), во-первых, размер ключа был уменьшен до 64 бит, из которых только 56 являются секретными, во-вторых, в алгоритме определены перестановки лишь специального вида, не зависящие от ключа, что наводило критиков этого алгоритма на мысль, что АНБ могла использовать известные ей слабости алгоритма для его взлома. На протяжении последних десятилетий DES подвергался интенсивному и всестороннему исследованию и по современным понятиям уже не считается надежным.

Существует несколько предложений, направленных на усовершенствование DES. Наиболее известное из них заключается в трехкратном применении алгоритма (Triple DES).

Наиболее известные блочные шифры - IDEA, BLOWFISH, SKIPJACK.

IDEA (International Data Encryption Algorithm) разработан в 1991 г, работает с блоками данных длиной 64 бита, используя ключ длиной 128 бит, число раундов равно восьми. Используемые преобразования - умножение по модулю $2^{16} + 1$, сложение по модулю 2, сложение по модулю 2^{16}. Авторы - К. Лэй, Д. Мэссей.

BLOWF1SH - разработан в 1994 г. Б. Шнайером; работает с блоками данных разрядностью 64 бита, используя ключ переменной длины (максимальная разрядность равна 448 битам), число раундов равно 16. Используемые преобразования - подстановка, сложение по модулю 2, сложение по модулю 2^{32}.

SKIPJACK - разработан в 1990 г. NSA в качестве криптоалгоритма для микросхем Clipper и Capstone. Первоначально алгоритм был объявлен секретным, однако впоследствии его описание "просочилось" в Интернет. Шифр работает с блоками данных разрядностью 64 бита с использованием 80-разрядного ключа. Число раундов равно 32.

Поточные цифры

Шифр Вернама можно считать исторически первым поточным шифром. Так как поточные шифры, в отличие от блочных, осуществляют поэлементное шифрование потока данных без задержки в криптосистеме, их важнейшим достоинством является высокая скорость преобразования, соизмеримая со скоростью поступления входной информации. Таким образом обеспечивается шифрование практически в реальном масштабе времени вне зависимости от объема и разрядности потока преобразуемых данных.

Простейшие устройства синхронного и самосинхронизирующегося шифрования с использованием ГПК, реализованного на основе *N*-разрядного *регистра сдвига с лилейной обратной связью* - *LFSR* (Linear Feedback Shift Register), называются *скремблерами***,** *а* сам процесс преобразования — скремблированием.

В синхронных поточных шифрах гамма формируется независимо от входной последовательности, каждый элемент (бит, символ, байт и т. п.) которой таким образом шифруется независимо от других элементов. В синхронных поточных шифрах отсутствует эффект размножения ошибок, т. е. число искаженных элементов в расшифрованной последовательности равно числу искаженных элементов зашифрованной последовательности, пришедшей из канала связи. Вставка или выпадение элемента зашифрованной последовательности недопустимы, так как из-за нарушения синхронизации это приведет к неправильному расшифрованию всех последующих элементов.

Контрольные вопросы и задания

1. Приведите примеры шифров, для которого сам открытый текст является ключом.
2. Какие шифры называются омофонами? В чем их преимущество перед шифрами простой замены?
3. Что является ключом шифра Вижинера?
4. Приведите пример шифра, допускающего неоднозначное зашифрование.
5. В чем отличие симметричных шифрсистем от ассиметричных?
6. Перечислите виды активных угроз и виды пассивных угроз.
7. На основе каких характеристик в общем случае строится классификация криптографических систем?
8. Приведите пример безусловно защищенной (абсолютно стойкой) схемы шифрования.
9. Продолжите фразу: «Схема шифрования называется защищенной по вычислениям, если…».
10. В чем различие между алгебраической и вероятной моделями шифра?

11. Приведите пример шифра перестановки, который может рассматриваться и как блочный шифр замены.
12. Что более целесообразно для надежной защиты информации: архивация открытого текста с последующим шифрованием или шифрование открытого текста с последующей архивацией?
13. В шифре гаммирования в качестве гаммы был использован текст художественного произведения. Предложите метод вскрытия такого шифра.

2. АЛГОРИТМЫ БЛОЧНОГО ШИФРОВАНИЯ И ЭЛЕМЕНТЫ КРИПТОАНАЛИЗА

Алгоритмы блочного шифрования – одна из фундаментальных областей криптографии. За последние десятилетия были разработаны сртни алгоритмов, большинство из которых представляет собой последовательное применение простого преобразования. В итоге получается достаточно стойкий шифр, даже если одно преобразование (один раунд) является нестойким (слабым).

К коротким сообщениям алгоритм можно применять непосредственно. Если же длина сообщения больше длины ключа (чаще всего бывает именно так), то необходимо использовать один из режимов работы блочных шифров.

Мы начинаем изучение блочных шифров с алгоритма , который является хорошей «моделью», т. к. обладает многими свойствами, присущими большинству алгоритмов.

2.1. СТАНДАРТ ШИФРОВАНИЯ ДАННЫХ DES (DATA ENCRYPTION STANDARD)

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 проверен и сертифицирован, а использующие его различные криптографические устройства могли бы взаимодействовать. Он мог бы, к тому же, быть относительно недорогим и легко доступным [17].

15 мая 1973 года в *Federal Register* NBS опубликовало требования к криптографическому алгоритму, который мог бы быть принят в качестве стандарта. Было приведено несколько критериев оценки проекта:

—Алгоритм должен обеспечивать высокий уровень безопасности.

—Алгоритм должен быть полностью определен и легко понятен.

—Безопасность алгоритма должна основываться на ключе и не должна зависеть от сохранения в тайне самого алгоритма.

—Алгоритм должен быть доступен всем пользователям.

—Алгоритм должен позволять адаптацию к различным применениям.

—Алгоритм должен позволять экономичную реализацию в виде электронных приборов.

—Алгоритм должен быть эффективным в использовании.

—Алгоритм должен предоставлять возможности проверки.

—Алгоритм должен быть разрешен для экспорта.

Реакция общественности показала, что к криптографическому

стандарту существует заметный интерес, но опыт в этой области чрезвычайно мал. Ни одно из предложений не удовлетворяло предъявленным требованиям.

27 августа 1972 года в *Federal Register* NBS опубликовало повторное предложение. Наконец, у Бюро появился подходящий кандидат: алгоритм под именем Люцифер, в основе которого лежала разработка компании IBM, выполненная в начале 70-х

Несмотря на определенную сложность алгоритм был прямолинеен. Он использовал только простые логические операции над небольшими группами битов и мог быть довольно эффективно реализован в аппаратуре.

17 марта 1975 года в *Federal Register* NBS опубликовало и подробности алгоритма, и заявление IBM о предоставлении неисключительной, бесплатной лицензии на алгоритм.

В 1976 году NBS провело два симпозиума по оценке предложенного стандарта. На первом обсуждались математика алгоритма и возможность потайной дверцы. На втором - возможности увеличения длины ключа алгоритма. Были приглашены создатели алгоритма, люди, оценивавшие алгоритм, разработчики аппаратуры, поставщики, пользователи и критики. По всем отчетам симпозиумы были весьма оживленными.

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.

Принятие стандарта

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).

Американская ассоциация банкиров разрабатывает необязательные стандарты для финансовой индустрии. Они опубликовали стандарт, рекомендующий DES для шифрования, и другой стандарт для управления криптографическими ключами.

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.

Было отмечено, что DES широко используется в бизнесе (особенно в финансах), и что приемлемой альтернативы не существует. Отказ от стандарта оставил бы многие организации без защиты данных. После длительных споров DES был вновь утвержден в качестве правительственного стандарта США до 1992 года.

Наконец было разрешено сертифицировать и программные реализации DES.

2.1.1. Описание DES

DES представляет собой блочный шифр, он шифрует данные 64-битовыми блоками. С одного конца алгоритма вводится 64-битовый блок открытого текста, а с другого конца выходит 64-битовый блок шифротекста. DES является симметричным алгоритмом: для шифрования и дешифрирования используются одинаковые алгоритм и ключ (за исключением небольших различий в использовании ключа).

Длина ключа равна 56 битам. (Ключ обычно представляется 64-битовым числом, но каждый восьмой бит используется для проверки четности и игнорируется. Биты четности являются наименьшими значащими битами байтов ключа.) Ключ, который может быть любым 56-битовым числом, можно изменить в любой момент времени. Ряд чисел считаются слабыми ключами, но их можно легко избежать. Безопасность полностью определяется ключом.

На простейшем уровне алгоритм не представляет ничего большего, чем комбинация двух основных методов шифрования: смещения и диффузии. Фундаментальным строительным блоком DES является применение к тексту единичной комбинации этих методов (подстановка, а за ней - перестановка), зависящей от ключа. Такой блок называется этапом. DES состоит из 16 этапов, одинаковая комбинация методов применяется к открытому тексту 16 раз (см. рисунок 15).

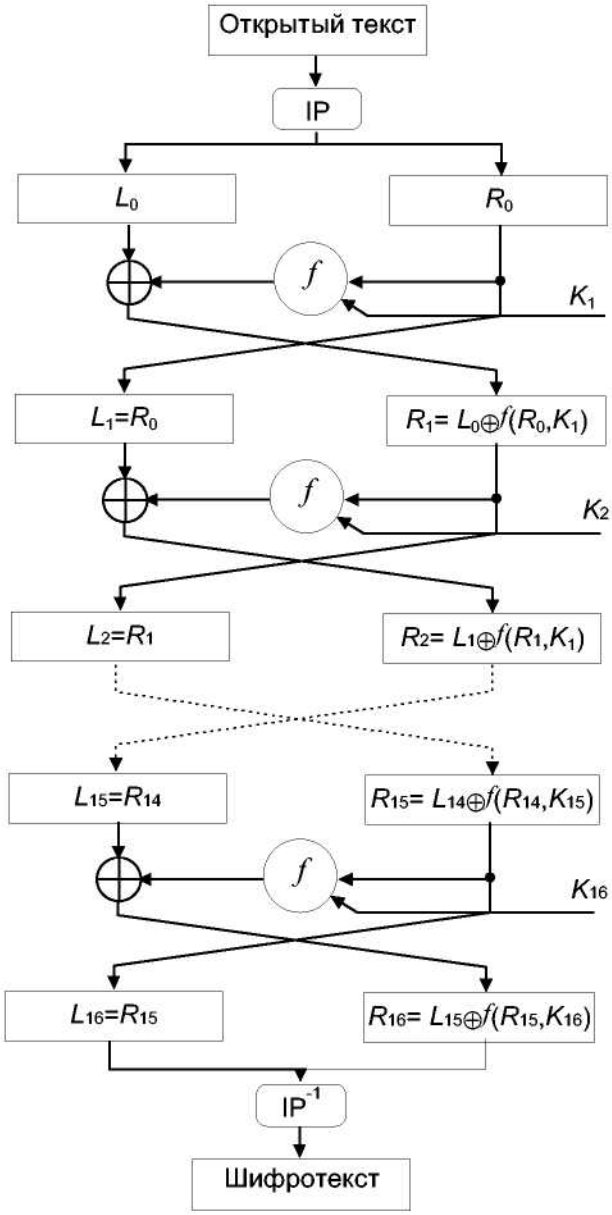


Рис. 15. DES.

Алгоритм использует только стандартную арифметику 64-битовых чисел и логические операции, поэтому он легко реализовывался в аппаратуре второй половины 70-х. Изобилие повторений в алгоритме делает его идеальным для реализации в специализированной микросхеме.

Схема алгоритма

Для описания воспользуемся сведениями [17]. DES работает с 64-битовым блоком открытого текста. После первоначальной перестановки блок разбивается на правую и левую половины длиной по 32 бита. Затем выполняется 16 этапов одинаковых действий, называемых функцией f, в которых данные объединяются с ключом. После шестнадцатого этапа правая и левая половины объединяются и алгоритм завершается заключительной перестановкой (обратной по отношению к первоначальной).

На каждом этапе (см. рисунок 16) биты ключа сдвигаются, и затем из 56 битов ключа выбираются 48 битов. Правая половина данных увеличивается до 48 битов с помощью перестановки с расширением, объединяется посредством XOR с 48 битами смещенного и переставленного ключа, проходит через 8 S-блоков, образуя 32 новых бита, и переставляется снова. Эти четыре операции и выполняются функцией f Затем результат функции f объединяется с левой половиной с помощью другого XOR. В итоге этих действий появляется новая правая половина, а старая правая половина становится новой левой. Эти действия повторяются 16 раз, образуя 16 этапов DES.

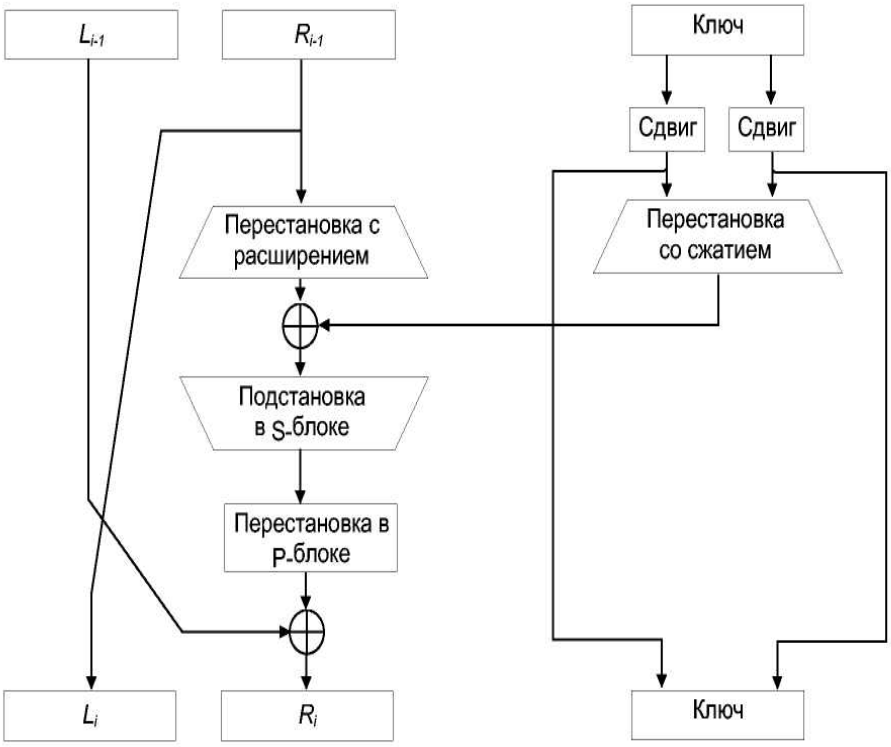


Рис. 16. Один этап DES.

Если B_i - это результат i-ой итерации, L_i и R_i - левая и правая половины B_i, K_i - 48-битовый ключ для этапа i, а f - это функция, выполняющие все подстановки, перестановки и XOR с ключом, то этап можно представить как:

$L_i = R_{i-1}$

$R_i = L_{i-1} \oplus f(R_{i-1}, K_i)$

Начальная перестановка

Начальная перестановка выполняется еще до этапа 1, при этом входной блок переставляется, как показано в 11-й. Эту и все другие таблицы этой главы надо читать слева направо и сверху вниз. Например, начальная перестановка перемещает бит 58 в битовую позицию 1, бит 50 - в битовую позицию 2, бит 42 - в битовую позицию 3, и так далее.

Табл. 1 Начальная перестановка

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	14	3
61	53	45	37	29	21	13	5	63	55	47	39	31	23	15	7

Начальная перестановка и соответствующая заключительная перестановка не влияют на безопасность DES. (Как можно легко заметить, эта перестановка в первую очередь служит для облегчения побайтной загрузки данных открытого текста и шифротекста в микросхему DES. Не забывайте, что DES появился раньше 16- и 32-битовых микропроцессорных шин.) Так как программная реализация этой многобитовой перестановки нелегка (в отличие от тривиальной аппаратной), во многих программных реализациях DES начальная и заключительные перестановки не используются. Хотя такой новый алгоритм не менее безопасен, чем DES, он не соответствует стандарту DES и, поэтому, не может называться DES.

Преобразования ключа

Сначала 64-битовый ключ DES уменьшается до 56-битового ключа отбрасыванием каждого восьмого бита, как показано в 10-й. Эти биты используются только для контроля четности, позволяя проверять правильность ключа. После извлечения 56-битового ключа для каждого из 16 этапов DES генерируется новый 48-битовый подключ**.** Эти подключи, K_i, определяются следующим образом.

Табл. 2 Перестановка ключа

57	49	41	33	25	17	9	1	58	50	42	34	26	18
10	2	59	51	43	35	27	19	14	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

Во-первых, 56-битовый ключ делится на две 28-битовых половинки. Затем, половинки циклически сдвигаются налево на один или два бита в зависимости от этапа. Этот сдвиг показан в 9-й.

Табл. 3 Число битов сдвига ключа в зависимости от этапа

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

После сдвига выбирается 48 из 56 битов. Так как при этом не только выбирается подмножество битов, но и изменяется их порядок, эта операция называется **перестановка со сжатием.** Ее результатом является набор из 48 битов. Перестановка со сжатием (также называемая переставленным выбором) определена в 8-й. Например, бит сдвинутого ключа в позиции 33 перемещается в позицию 35 результата, а 18-й бит сдвинутого ключа отбрасывается.

Табл. 4 Перестановка со сжатием

14	17	14	24	1	5	3	28	15	6	21	10

23	19	14	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

Из-за сдвига для каждого подключа используется отличное подмножество битов ключа. Каждый бит используется приблизительно в 14 из 16 подключей, хотя не все биты используются в точности одинаковое число раз.

Перестановка с расширением

Эта операция расширяет правую половину данных, R_i, от 32 до 48 битов. Так как при этом не просто повторяются определенные биты, но и изменяется их порядок, эта операция называется **перестановкой с расширением.** У нее две задачи: привести размер правой половины в соответствие с ключом для операции XOR и получить более длинный результат, который можно будет сжать в ходе операции подстановки. Однако главный криптографический смысл совсем в другом. За счет влияния одного бита на две подстановки быстрее возрастает зависимость битов результата от битов исходных данных. Это называется **лавинным эффектом.** DES спроектирован так, чтобы как можно быстрее добиться зависимости каждого бита шифротекста от каждого бита открытого текста и каждого бита ключа.

Перестановка с расширением показана на 9-й. Иногда она называется **Е-блоком** (от expansion). Для каждого 4-битового входного блока первый и четвертый бит представляют собой два бита выходного блока, а второй и третий биты - один бит выходного блока. В 7-й показано, какие позиции результата соответствуют каким позициям исходных данных. Например, бит входного блока в позиции 3 переместится в позицию 4 выходного блока, а бит входного блока в позиции 21 - в позиции 30 и 32 выходного блока.

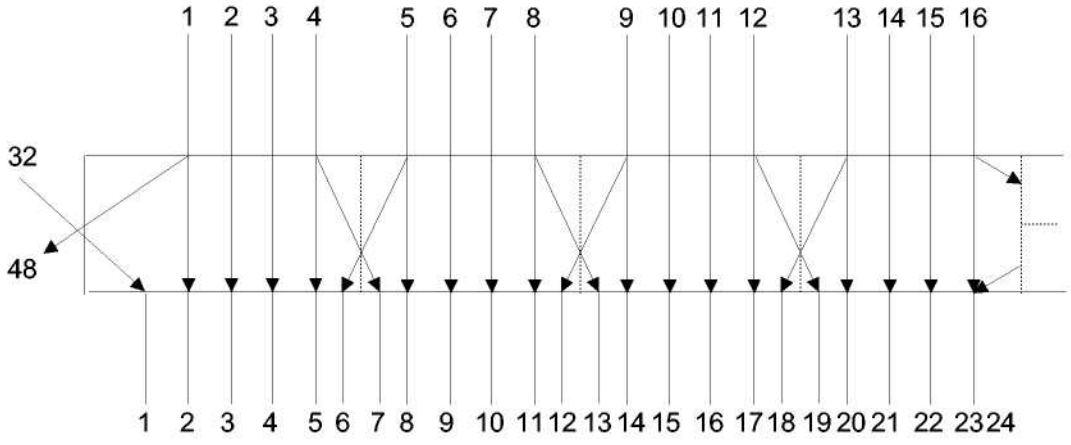


Рис. 17 Перестановка с расширением.

Хотя выходной блок больше входного, каждый входной блок генерирует уникальный выходной блок.

Табл. 5 Перестановка с расширением

32	1	2	3	4	5	4	5	6	7	8	9

8	9	10	14	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

Подстановка с помощью S-блоков

После объединения сжатого блока с расширенным блоком с помощью XOR над 48-битовым результатом выполняется операция подстановки. Подстановки производятся в восьми **блоках подстановки,** или **S-блоках** (от substitution). У каждого S-блока 6-битовый вход и 4-битовый выход, всего используется восемь различных S-блоков. (Для восьми S-блоков DES потребуется 256 байтов памяти.) 48 битов делятся на восемь 6-битовых подблока. Каждый отдельный подблок обрабатывается отдельным S-блоком: первый подблок - S-блоком 1, второй - S-блоком *2,* и так далее.

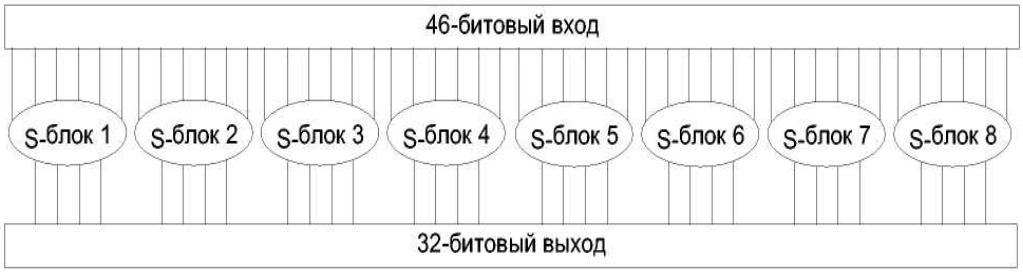


Рис. 18 Подстановка - S-блоки.

Каждый S-блок представляет собой таблицу из 2 строк и 16 столбцов. Каждый элемент в блоке является 4-битовым числом. По 6 входным битам S-блока определяется, под какими номерами столбцов и строк искать выходное значение. Все восемь S-блоков показаны в таблице.

Табл. 6 S-блоки

S-блок 1:															
14	4	13	1	2	15	14	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	14	9	5	3	8
4	1	14	8	13	6	2	14	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	14	3	14	10	0	6	13
S-блок 2:															
15	1	8	14	6	14	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	14	5
0	14	7	14	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	14	6	7	12	0	5	14	9
S-блок 3:															
10	0	9	14	6	3	15	5	1	13	12	7	14	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	14	15	1
13	6	4	9	8	15	3	0	14	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	14	5	2	12

S-блок 4:

7	13	14	3	0	6	9	10	1	2	8	5	14	12	4	15
13	8	14	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	14	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	14	12	7	2	14

S-блок 5:

2	12	4	1	7	10	14	6	8	5	3	15	13	0	14	9
14	14	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	14	10	13	7	8	15	9	12	5	6	3	0	14
14	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S-блок 6:

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	14
10	15	4	2	7	12	9	5	6	1	13	14	0	14	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	14	6
4	3	2	12	9	5	15	10	14	14	1	7	6	0	8	13

S-блок 7:

4	14	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	14	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	14	13	12	3	7	14	10	15	6	8	0	5	9	2
6	14	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S-блок 8:

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

Входные биты особым образом определяют элемент S-блока. Рассмотрим 6-битовый вход S-блока: b_1, b_2, b_3, b_4, b_5 и b_6. Биты b_1 и b_6 объединяются, образуя 2-битовое число от 0 до 3, соответствующее строке таблицы. Средние 4 бита, с b_2 по b_5, объединяются, образуя 4-битовое число от 0 до 15, соответствующее столбцу таблицы.

Например, пусть на вход шестого S-блока (т.е., биты функции XOR с 31 по 36) попадает 110011. Первый и последний бит, объединяясь, образуют 11, что соответствует строке 3 шестого S-блока. Средние 4 бита образуют 1001, что соответствует столбцу 9 того же S-блока. Элемент S-блока 6, находящийся на пересечении строки 3 и столбца 9, - это 14. (Не забывайте, что строки и столбцы нумеруются с 0, а не с 1.) Вместо 110011 подставляется 1110.

Конечно же, намного легче реализовать S-блоки программно в виде массивов с 64 элементами. Для этого потребуется переупорядочить элементы, что не является трудной задачей. (Изменить индексы, не изменяя порядок элементов, недостаточно. S-блоки спроектированы очень

тщательно.) Однако такой способ описания S-блоков помогает понять, как они работают. Каждый S-блок можно рассматривать как функцию подстановки 4-битового элемента: b_2 по b_5 являются входом, а некоторое 4-битовое число - результатом. Биты b_1 и b_6 определяются соседними блоками, они определяют одну из четырех функций подстановки, возможных в данном S-блоке.

Подстановка с помощью S-блоков является ключевым этапом DES. Другие действия алгоритма линейны и легко поддаются анализу. S-блоки нелинейны, и именно они в большей степени, чем все остальное, обеспечивают безопасность DES.

В результате этого этапа подстановки получаются восемь 4-битовых блоков, которые вновь объединяются в единый 32-битовый блок. Этот блок поступает на вход следующего этапа - перестановки с помощью P-блоков.

Перестановка с помощью P-блоков

32-битовый выход подстановки с помощью S-блоков, перетасовываются в соответствии с P-блоком. Эта перестановка перемещает каждый входной бит в другую позицию, ни один бит не используется дважды, и ни один бит не игнорируется. Этот процесс называется прямой перестановкой или просто перестановкой. Позиции, в которые перемещаются биты, показаны в 5-й. Например, бит 21 перемещается в позицию 4, а бит 4 - в позицию 31.

Табл. 7 Перестановка с помощью P-блоков

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,	И,	4,	25

Наконец, результат перестановки с помощью P-блока объединяется посредством XOR с левой половиной первоначального 64-битового блока. Затем левая и правая половины меняются местами, и начинается следующий этап.

Заключительная перестановка

Заключительная перестановка является обратной по отношению к начальной перестановки и описана в 4-й. Обратите внимание, что левая и правая половины не меняются местами после последнего этапа DES, вместо этого объединенный блок $R_{16}L_{16}$ используется как вход заключительной перестановки. В этом нет ничего особенного, перестановка половинок с последующим циклическим сдвигом привела бы к точно такому же результату. Это сделано для того, чтобы алгоритм можно было использовать как для шифрования, так и для дешифрирования.

Табл. 8 Заключительная перестановка

40	8	48	16	56	24	64	32	39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30	37	5	45	13	53	21	61	29

36	4	44	12	52	20	60	28	35	3	43	14	51	19	59	27
34	2	42	10	50	18	58	26	33	1	41	9	49	17	57	25

2.1.2. Режимы работы блочных шифров

Алгоритм DES является базовым строительным блоком защиты передачи данных. Для применения DES в различных приложениях были определены четыре режима его работы . Предполагается, что этих четырех режимов должно быть достаточно для того, чтобы использовать DES практически в любой области, для которой этот алгоритм подходит. Эти режимы представлены в Таблицаице, а в оставшейся части раздела даны их краткие описания. Эти режимы применимы и для других блочных шифров симметричной схемы.

Режим электронной шифровальной книги

Простейшим режимом является режим электронной шифровальной книги, (ECB), когда открытый текст обрабатывается блоками по 64 бита и каждый блок шифруется с одним и тем же ключом . Термин *шифровальная книга* объясняется тем, что при заданном ключе каждый 64-битовый блок открытого текста представляется уникальным блоком шифрованного текста. Такое соответствие вызывает аналогию с воображаемой гигантской шифровальной книгой, в которой для каждой 64-битовой последовательности открытого текста указана соответствующая последовательность шифрованного текста.

Если длина сообщения превышает 64 бита, оно разделяется на 64-битовые блоки с добавлением при необходимости заполнителей к последнему блоку. Дешифрование тоже выполняется поблочно на основе одного и того же ключа.

Метод ECB идеален для небольших объемов данных, например для ключей шифрования. Поэтому, если необходимо переслать ключ DES, обеспечив его защиту от разглашения, можно для этого воспользоваться режимом ECB.

Самой важной особенностью режима ECB является то, что одинаковые 64-битовые блоки открытого текста при условии, что таковые встречаются в исходном сообщении, в шифрованном тексте тоже будут представляться одинаковыми блоками

Таблица. Режимы работы DES

Режим	Описание	Типичные области применения
Электронная шифровальная книга (ECB - Electronic Code-book)	Каждый 64-битовый блок открытого текста шифруется независимо от других с одним и тем же ключом	Защищенная передача отдельных значений (например, ключа шифрования)
Сцепление шифрованных блоков (CBC — Cipher Block Chaining)	Входной блок данных для алгоритма шифрования вычисляется как XOR-разность текущего 64-битового блока открытого текста и предшествующего 64-битового блока шифрованного текста	Поблочная передача данных общего назначения Аутентификация
Шифрованная обратная связь (CFB - Cipher Feedback)	Входные данные обрабатываются порциями по J битов. Полученный на предыдущем шаге шифрованный текст используется как входные данные для алгоритма шифрования с целью получения псевдослучайной последовательности, XOR-разность которой и блока открытого текста определяет очередную порцию шифрованного текста	Потоковая передача данных общего назначения Аутентификация
Обратная связь по выходу (OFB - Output Feedback)	Подобна CFB, но в качестве входных данных для алгоритма шифрования используются ранее полученные выходные данные DES	Потоковая передача данных по каналам с помехами (например, по спутниковой связи)

Поэтому при передаче достаточно длинных сообщений режим ECB может не обеспечивать необходимый уровень защиты. Если сообщение имеет явно выраженную структуру, у криптоаналитика появляется возможность использовать регулярности текста. Например, если известно, что в начале сообщения всегда размещается определенный заголовок, криптоаналитик может получить в свое распоряжение целый набор соответствующих пар блоков открытого и шифрованного текста. Или если сообщение содержит

повторяющиеся элементы с периодом их повторения, кратным 64 битам, такие элементы тоже могут быть выявлены аналитиком. Подобные сведения упрощают задачу криптоаналитика или предоставляют возможность для замещения или реорганизации блоков текста в передаваемом сообщении.

Режим сцепления шифрованных блоков

Технология, свободная от недостатков режима ECB, должна в случае повторения в сообщении уже встречавшегося ранее блока открытого текста генерировать блок шифрованного текста, отличный от сгенерированного ранее. Проще всего добиться этого с помощью режима сцепления шифрованных блоков (CBC). В этом режиме входное значение алгоритма шифрования задается равным XOR-разности текущего блока открытого текста и полученного на предыдущем шаге блока шифрованного текста. Шифрование любого блока выполняется с одним и тем же ключом. В результате в процессе шифрования все блоки открытого текста оказываются связанными, а входные данные, поступающие на вход функции шифрования, уже не жестко связаны с блоками открытого текста. Поэтому повторяющиеся 64-битовые последовательности в шифрованном тексте не проявляются.

При дешифровании текст тоже проходит через алгоритм дешифрования поблочно. При этом соответствующий блок открытого текста получается как XOR-разность выходного блока алгоритма дешифрования и предыдущего блока шифрованного текста.' Чтобы понять, как это происходит, запишем процесс шифрования в виде формулы

$$C_n = E_K[C_{n-1} \oplus P_n].$$

Тогда

$$D_K[C_n] = D_K[E_K(C_{n-1} \oplus P_n)],$$

$$D_K[C_n] = (C_{n-1} \oplus P_n),$$

$$C_{n-1} \oplus D_K[C_n] = C_{n-1} \oplus C_{n-1} \oplus P_n = P_n.$$

Чтобы получить первый блок шифрованного текста, рассматривается XOR-разность некоторого инициализационного вектора (IV) и первого блока открытого текста. При дешифровании для восстановления первого блока открытого текста необходимо будет тоже выполнить операцию XOR по отношению к тому же вектору IV и первому блоку на выходе алгоритма дешифрования.

Значение IV должно быть известно и отправителю, и получателю сообщения. Для обеспечения максимальной безопасности значение IV должно быть защищено точно так же, как и ключ. Можно, например, отправить значение IV, зашифровав его в режиме ECB. Одной из причин, по

которым необходимо защищать IV, является следующая: если противник имеет возможность обмануть адресата сообщения, предоставив подложный IV, то противник получает возможность инвертировать избранные биты в первом блоке открытого текста. Чтобы убедиться в этом, рассмотрим следующие уравнения:

$$C_1 = E_K(IV \oplus P_n),$$
$$P_1 = IV \oplus D_K(C_1).$$

Обозначив i-й бит 64-битового значения X через X[i], получим

$$P_1[i] = IV[i] \oplus D_K(C_1)[i].$$

Поэтому, используя свойства операции XOR, можем заключить, что

$$P_1[i]' = IV[i]' \oplus D_K(C_1)[i],$$

где штрих означает дополнение бита. Как видите, если противник имеет возможность управляемым образом менять биты значения IV, он сможет поменять и соответствующие значения P,.

В заключение заметим, что благодаря механизму сцепления блоков CBC этот режим оказывается подходящим для шифрования сообщений, длина которых превышает 64 бита.

Помимо обеспечения конфиденциальности, режим CBC можно использовать для аутентификации .

Режим шифрованной обратной связи

Схема DES представляет собой блочный шифр с размером блока 64 бита. Но DES можно преобразовать и в потоковый шифр, используя либо режим шифрованной обратной связи (CFB), либо режим обратной связи по выходу (ОГВ). Использование поточного шифра избавляет от необходимости дополнять сообщение до целого числа блоков. Кроме того, с поточным шифром можно работать в режиме реального времени. Например, при передаче потока символов с помощью посимвольного поточного шифра каждый символ можно шифровать и сразу же передавать адресату, не дожидаясь окончания шифрования остальной части сообщения.

Для поточного шифра необходимо, чтобы длина шифрованного текста в точности соответствовала длине открытого. Так, при передаче 8-битовых символов следует обратиться к 8-битовому шифрованию. Если при этом использовать шифрование блоками длиной более 8 битов, часть ресурсов канала передачи данных будет расходоваться зря.

Единицей передачи данных являются j битов (обычноj = 8). Как и в режиме CBC, происходит сцепление элементов открытого текста, поэтому шифрованный текст, соответствующий любому элементу открытого текста, будет зависеть от всех предыдущих элементов открытого текста.

Сначала рассмотрим шифрование. На входе функции шифрования размещается 64-битовый регистр сдвига, в котором изначально размещается некоторое значение инициализационного вектора (IV). Крайние слева (главные) *j* битов этого значения связываются операцией XOR с первой порцией открытого текста Pt, в результате чего получается первая порция шифрованного текста C!, который подается на линию передачи данных. Содержимое регистра сдвига смещается влево на / битов, а в крайние справа (наименее значимые) *j* битов помещается значение C,. Затем весь процесс повторяется до тех пор, пока не будут зашифрованы все элементы открытого текста.

Для дешифрования используется та же схема, но для получения очередной порции открытого текста с помощью операции XOR связываются полученная по связи порция шифрованного текста и получаемые на выходе функции шифрования данные. Обратите внимание, что в данном случае используется функция *шифрования,* а не дешифрования. Объяснить это просто. Обозначим крайние слева ; битов значения X через S,(X). Тогда

$$C_1 = P_1 \oplus S_j(E(IV)),$$

и поэтому

$$P_1 = C_1 \oplus S_j(E(IV)).$$

Те же самые выкладки имеют место и для последующих шагов дешифрования. Режим CFB может использоваться как для обеспечения конфиденциальности, так и для аутентификации.

Режим обратной связи по выходу

Режим обратной связи по выходу (OFB), как видно из рис. 3.14, во многом подобен режиму CFB. В режиме OFB в регистр сдвига подается значение, получаемое на выходе функции шифрования, а в режиме CFB в этот регистр подается порция шифрованного текста.

Режим OFB обладает тем преимуществом, что влияние возможных искажений битов при передаче данных не распространяется на последующие порции данных. Например, если искаженные биты появились при передаче $C_{,,}$ это повлияет только на восстановленное из C_t значение $P_{,,}$ а все последующие порцииоткрытого текста из-за этой шибки передачи данных повреждены не будут. В случае CFB значение C_x используется в качестве входного для регистра сдвига, вследствие чего искажение d выльется в дополнительные искажения для всего потока принимаемых данных.

Недостатком режима OFB, с другой стороны, является то, что он обеспечивает меньшую, чем CFB, надежность в отношении нарушений типа модификации потока данных. Например, изменение одного бита

шифрованного текста отражается в изменении соответствующего бита восстановленного открытого текста. Это дает возможность осуществления контролируемых изменений в получаемом адресатом открытом тексте. Практически, чтобы программа коррекции ошибок не заметила подмены, противник должен внести необходимые ему изменения как в данные, представляющие собой порцию шифрованного текста, так и в данные, представляющие контрольную сумму для этой порции текста.

2.1.3. Расшифрование DES

После всех подстановок, перестановок, операций XOR и циклических сдвигов можно подумать, что алгоритм дешифрирования, резко отличаясь от алгоритма шифрования, точно также запутан. Напротив, различные компоненты DES были подобраны так, чтобы выполнялось очень полезное свойство: для шифрования и дешиф рирования используется один и тот же алгоритм.

DES позволяет использовать для шифрования или дешифрирования блока одну и ту же функцию. Единственное отличие состоит в том, что ключи должны использоваться в обратном порядке. То есть, если на этапах шифрования использовались ключи K_1, K_2, K_3, ..., K_{16}, то ключами дешифрирования будут K_{16}, K_{15}, K_{14}, ..., K_1. Алгоритм, который создает ключ для каждого этапа, также цикличен. Ключ сдвигается направо, а число позиций сдвига равно 0, 1,2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 1.

Режимы DES

FIPS PUB 81 определяет четыре режима работы: ECB, CBC, OFB и CFB. Банковские стандарты ANSI определяют для шифрования ECB и CBC, а для проверки подлинности - CBC и n-битовый CFB.

В мире программного обеспечения сертификация обычно не важна. Из-за своей простоты в большинстве существующих коммерческих программ используется ECB, хотя этот режим наиболее чувствителен к вскрытию. CBC используется редко несмотря на то, что он лишь незначительно сложнее, чем ECB, и обеспечивает большую безопасность.

Аппаратные и программные реализации DES

Утверждается, что самой быстрой является микросхема DES, разработанная в Digital Equipment Corporation. Она поддерживает режимы ECB и CBC и основана на вентильной матрице GaAs, состоящей из 50000 транзисторов. Данные могут зашифровываться и дешифрироваться со скоростью 1 гигабит в секунду, обрабатывая 16.8 миллионов блоков в секунду. Кажущиеся противоречия между тактовой частотой и скоростью обработки данных обусловлены конвейеризацией внутри микросхемы, в которой может быть реализовано несколько работающих параллельно DES-механизмов.

Наиболее выдающейся микросхемой DES является 6868 VLSI (ранее называвшаяся "Gatekeeper" - Вратарь). Она не только может выполнять шифрование DES за 8 тактов (лабораторные прототипы могут делать это за 4 такта), но также выполнять троекратный DES в режиме ECB за 25 тактов, а троекратный DES в режимах OFB или CBC - за 35 актов.

Программная реализация DES на мэйнфрейме IBM 3090 может выполнить 32000 шифрований DES в секунду. На других платформах скорость ниже, но все равно достаточно велика. В 2-й приведены действительные результаты и оценки для различных микропроцессоров Intel и Motorola.

Безопасность DES

Люди давно интересуются безопасностью DES. Было много рассуждений о длине ключа, количестве итераций и схеме S-блоков. S-блоки были наиболее таинственными - какие-то константы, без видимого объяснения для чего и зачем они нужны. Хотя IBM утверждала, что работа алгоритма была результатом 17 человеко-лет интенсивного криптоанализа, некоторые люди опасались, что NSA вставило в алгоритм лазейку, которая позволит агентству легко дешифрировать перехваченные сообщения.

Комитет по разведке Сената США чрезвычайно тщательно расследовал этот вопрос в 1978 году. Результаты работы комитета были засекречены, но в открытых итогах этого расследования с NSA были сняты все обвинения в неуместном вмешательстве в проектирование алгоритма. "Было сказано, что NSA убедило IBM в достаточности более короткого ключа, косвенно помогло разработать структуры S-блоков и подтвердило, что в окончательном варианте DES, с учетом всех знаний NSA, отсутствовали статистические или математические бреши".

Слабые ключи

Из-за того, что первоначальный ключ изменяется при получении подключа для каждого этапа алгоритма, определенные первоначальные ключи являются **слабыми**. Вспомните, первоначальное значение расщепляется на две половины, каждая из которых сдвигается независимо. Если все биты каждой половины равны 0 или 1, то для всех этапов алгоритма используется один и тот же ключ. Это может произойти, если ключ состоит из одних 1, из одних 0, или если одна половина ключа состоит из одних 1, а другая - из одних 0. Кроме того, у два слабых ключа обладают другими свойствами, снижающими их безопасность.

Четыре слабых ключа показаны в шестнадцатиричном виде в 1-й. (Не забывайте, что каждый восьмой бит - это бит четности.)

Табл. 9 Слабые ключи DES

Значение слабого ключа (с битами четности)	Действительный ключ
0101 0101 0101 0101	0000000 0000000

1F1F1F1F 0E0E0E0E	0000000 FFFFFFF
E0E0 E0E0 F1F1 F1F1	FFFFFFF 0000000
FEFE FEFE FEFE FEFE	FFFFFFF FFFFFFF

Кроме того, некоторые пары ключей при шифровании переводят открытый текст в идентичный шифротекст. Иными словами, один из ключей пары может расшифровать сообщения, зашифрованные другим ключом пары. Это происходит из-за метода, используемого DES для генерации подключей - вместо 16 различных подключей эти ключи генерируют только два различных подключа. В алгоритме каждый из этих подключей используется восемь раз. Эти ключи, называемые **полуслабыми ключами,** в шестнадцатиричном виде приведены в 0-й.

Табл.10 Полуслабые пары ключей DES

01FE 01FE 01FE 01FE	и	FE01 FE01 FE01 FE01
1FE0 1FE0 0EF1 0EF1	и	E01F E01F F10E F10E
01E0 01E0 01F1 01F1	и	E001 E001 F101 F101
1FFE 1EEE 0EFE 0EFE	и	FE1F FE1F FE0E FE0E
011F 011F 010E 010E	и	1F01 1F01 0E01 0E01
E0FE E0FE F1FE F1FE	и	FEE0 FEE0 FEE1 FEE1

Ряд ключей генерирует только четыре подключа, каждый из которых четыре раза используется в алгоритме.

Эти **возможно слабые ключи** перечислены в табл. 11.

Табл. 11 Возможно слабые ключи DES

1F	1F	01	01	0E	0E	01	01
01	1F	1F	01	01	0E	0E	01
1F	01	01	1F	0E	01	01	0E
01	01	1F	1F	01	01	0E	0E
E0	E0	01	01	F1	Fl	01	01
FE	FE	01	01	FE	FE	01	01
FE	E0	1F	01	FE	Fl	0E	01
E0	FE	1F	01	Fl	FE	0E	01
FE	E0	01	1F	FE	Fl	01	0E
E0	FE	01	1F	Fl	FE	01	0E
E0	E0	1F	1F	Fl	Fl	0E	0E
FE	FE	1F	1F	FE	FE	0E	0E
FE	IF	E0	01	FE	0E	Fl	01
E0	1F	FE	01	Fl	0E	FE	01
FE	01	E0	1F	FE	01	Fl	0E
E0	01	FE	1F	Fl	01	FE	0E
01	E0	E0	01	01	Fl	Fl	01
1F	FE	E0	01	0E	FE	F0	01
1F	E0	FE	01	0E	Fl	FE	01

E0	01	01	E0	Fl	01	01	Fl
FE	1F	01	E0	FE	0E	01	Fl
FE	01	1F	E0	FE	01	0E	Fl
E0	1F	IF	E0	Fl	0E	0E	Fl
FE	01	01	FE	FE	01	01	FE
E0	IF	01	FE	Fl	0E	01	FE
E0	01	1F	FE	Fl	01	0E	FE
FE	1F	1F	FE	FE	0E	0E	FE
IF	FE	01	E0	0E	FE	01	Fl
01	FE	1F	E0	01	FE	0E	Fl
IF	E0	01	FE	0E	Fl	01	FE
01	E0	IF	FE	01	Fl	0E	FE
01	01	E0	E0	01	01	Fl	Fl
IF	IF	E0	E0	0E	0E	Fl	Fl
IF	01	FE	E0	0E	01	FE	Fl
01	IF	FE	E0	01	0E	FE	Fl
IF	01	E0	FE	0E	01	Fl	FE
01	IF	E0	FE	01	0E	Fl	FE
01	01	FE	FE	01	01	FE	FE

01	FE	FE	01	01	FE	FE	01
1F	E0	E0	1F	0E	Fl	Fl	0E
01	FE	E0	1F	01	FE	Fl	0E
01	E0	FE	1F	01	Fl	FE	0E
1F	FE	FE	1F	0E	FE	FE	0E

IF	IF	FE	FE	0E	0E	FE	FE
FE	FE	E0	E0	FE	FE	Fl	Fl
E0	FE	FE	E0	Fl	FE	FE	Fl
FE	E0	E0	FE	FE	Fl	Fl	FE
E0	E0	FE	FE	Fl	Fl	FE	FE

Заметим, что эти 64 ключа - это крошечная часть полного набора из 72057594037927936 возможных ключей. Если вы выбираете ключ случайно, вероятность выбрать один из слабых ключей пренебрежимо мала. Можно всегда проверять "на слабость" сгенерированный ключ. Некоторые думают, что нечего и беспокоиться на этот счет. Другие утверждают, что проверка очень легка, почему бы ее и не выполнить.

Других слабых ключей в процессе исследований найдено не было.

Ключи-дополнения

Выполним побитное дополнение ключа, заменяя все 0 на 1 и все 1 - на 0. Теперь, если блок открытого текста зашифрован оригинальным ключом, то дополнение ключа при шифровании превратит дополнение блока открытого текста в дополнение блока шифротекста. Если x' обозначает дополнение x, то следующее верно:

$E_K(P) = C$

$E_{K'}(P) = C'$

В этом нет ничего таинственного. На каждом этапе после перестановки с расширением подключи подвергаются операции XOR с правой половиной. Прямым следствием этого факта и является приведенное свойство комплиментарности.

Это означает, что при выполнении вскрытия DES с выбранным открытым текстом нужно проверять только половину возможных ключей: 2^{55} вместо 2^{56}. Эли Бихам (Eli Biham) и Ади Шамир показали, что существует вскрытие с известным открытым текстом, имеющее ту же сложность, для которого нужно не меньше известных открытых текстов.

Остается вопросом, является ли такое свойство слабостью, так как в большинстве сообщений нет комплиментарных блоков открытого текста (для случайного открытого текста шансы "против" чрезвычайно велики), а пользователей можно предупредить не пользоваться дополняющими.

Алгебраическая структура

Все возможные 64-битовые блоки открытого текста можно отобразить на 64-битовые блоки шифротекста 2^{64}! Различными способами. Алгоритм DES, используя 56-битовый ключ, предоставляет нам 2^{56} (приблизительно 10^{17}) таких отображений. Использование многократного шифрования на первый взгляд позволяет значительно увеличить долю возможных отображений. Но это правильно только, если действие DES не обладает определенной алгебраической структурой.

Если бы DES был **замкнутым,** то для любых K_1 и K_2 всегда существовало бы такое K_3, что

$$E_{k_2}(E_{k_1}(P)) = E_{k_3}(p)$$

Другими словами, операция шифрования DES образовала бы группу, и шифрование набора блоков открытого текста последовательно с помощью К1 и К2 было бы идентично шифрованию блоков ключом К3. Что еще хуже, DES был бы чувствителен к вскрытию "встреча посередине" с известным открытым текстом, для которого потребовалось бы только 2^{28} этапов [807].

Если бы DES был **чистым,** то для любых K_1,K_2 *и* K_3 всегда существовало бы такое K_4, что

$$E_{k_3}(E_{k_2}(E_{k_1}(P))) = E_{k_4}(p)$$

Тройное шифрование было бы бесполезным. (Заметьте, что замкнутый шифр обязательно является и чистым, но чистый шифр не обязательно является замкнутым.)

Ряд подсказок можно найти в ранней теоретической работе Дона Копперсмита, но этого недостаточно. Различные криптографы пытались решить эту проблему. В повторяющихся экспериментах собирались "неопровержимые доказательства" того, что DES не является группой, но только в 1992 году криптографам удалось это доказать окончательно. Копперсмит утверждает, что команда IBM знала об этом с самого начала.

Длина ключа

В оригинальной заявке фирмы IBM в NBS предполагалось использовать 112-битовый ключ. К тому времени, когда DES стал стандартом, длина ключа уменьшилась до 56 бит. Многие криптографы настаивали на более длинном ключе. Основным их аргументом было вскрытие грубой силой.

В 1976 и 1977 гг. Диффи и Хеллман утверждали, что специализированный параллельный компьютер для вскрытия DES, стоящий 20 миллионов долларов, сможет раскрыть ключ за день. В 1981 году Диффи увеличил время поиска до двух дней, а стоимость - до 50 миллионов долларов. Диффи и Хеллман утверждали, что вскрытие в тот момент времени находилось за пределами возможностей любой организации, кроме подобных NSA, но что к 1990 году DES должен полностью утратить свою безопасность.

Хеллман продемонстрировал еще один аргумент против малого размера ключа: разменивая объем памяти на время, можно ускорить процесс поиска. Он предложил вычислять и хранить 2^{56} возможных результатов шифрования каждым возможным ключом единственного блока открытого текста. Тогда для взлома неизвестного ключа криптоаналитику потребуется только вставить блок открытого текста в шифруемый поток, вскрыть получившийся результат и найти ключ. Хеллман оценил стоимость такого устройства вскрытия в 5 миллионов долларов.

Шнайер отмечает в [17], что аргументы за и против существования в каком-нибудь тайном бункере правительственного устройства вскрытия DES

продолжают появляться. Многие указывают на то, что среднее время наработки на отказ для микросхем DES никогда не было большим настолько, чтобы обеспечивать работу устройства. Было показано, что этого возражения более чем достаточно. Другие исследователи предлагают способы еще больше ускорить процесс и уменьшить эффект отказа микросхем.

Между тем, аппаратные реализации DES постепенно приблизились к реализации требования о миллионе шифрований в секунду, предъявляемого специализированной машиной Диффи и Хеллмана. В 1984 году были выпущены микросхемы DES, способные выполнять 256000 шифрования в секунду. К 1987 году были разработаны микросхемы DES, выполняющие 512000 шифрований в секунду, и стало возможным появление варианта, способного проверять свыше миллиона ключей в секунду. А в 1993 Майкл Винер (Michael Wiener) спроектировал машину стоимостью 1 миллион долларов, которая может выполнить вскрытие DES грубой силой в среднем за 3.5 часа.

Никто открыто не заявил о создании этой машины, хоте разумно предположить, что кому-то это удалось. Миллион долларов – это не слишком большие деньги для большой и даже не очень большой страны.

В 1990 году два израильских математика, Бихам (Biham) и Шамир, открыли **дифференциальный криптоанализ,** метод, который позволил оставить в покое вопрос длины ключа. Прежде, чем мы рассмотрим этот метод, вернемся к некоторым другим критическим замечаниям в адрес DES.

Количество этапов

Почему 16 этапов? Почему не 32? После пяти этапов каждый бит шифротекста является функцией всех битов открытого текста и всех битов ключа, а после восьми этапов шифротекст по сути представляет собой случайную функцию всех битов открытого текста и всех битов ключа. (Это называется лавинным эффектом.) Так почему не остановиться после восьми этапов?

В течение многих лет версии DES с уменьшенным числом этапов успешно вскрывались. DES с тремя и четырьмя этапами был легко взломан в 1982 году. DES с шестью этапами пал несколькими годами позже. Дифференциальный криптоанализ Бихама и Шамира объяснил и это: DES с любым количеством этапов, меньшим 16, может быть взломан с помощью вскрытия с известным открытым текстом быстрее, чем с помощью вскрытия грубой силой. Конечно, грубый взлом является более вероятным способом вскрытия, но интересен тот факт, что алгоритм содержит ровно 16 этапов.

Проектирование S-блоков

Помимо уменьшения длины ключа NSA также обвиняют в изменении содержания S-блоков. Настаивая на подтверждении схемы S-блоков, NSA заявило, что детали алгоритма являются "чувствительными" и не могут быть опубликованы. Многие криптографы подозревали, что разработанные в NSA

S-блоки содержат лазейку, позволяющую NSA легко выполнять криптоанализ алгоритма.

С момента появления алгоритма для анализа схемы и работы S-блоков были предприняты значительные усилия. В середине 70-х Lexar Corporation и Bell Laboratories исследовали работу S-блоков. Ни одно из исследований не обнаружило никаких слабостей, хотя оба исследования обнаружили непонятный свойства. S-блоки имеют больше свойств, общих с линейным преобразованием, чем можно было ожидать при их формировании случайным образом. Команда Bell Laboratories констатировала, что S-блоки могут содержать скрытые лазейки, а доклад Lexar завершался следующей фразой:

В DES были найдены структуры, несомненно вставленные для повышения устойчивости системы к определенным типам вскрытия. Также были найдены структуры, которые, по-видимому, ослабили систему.

С другой стороны этот доклад также содержал следующее предупреждение:

... проблема [поиска структур в S-блоках] усложняется из-за способности человеческого сознания находить в случайных данных структуры, которые в действительности вовсе не являются структурами.

На втором симпозиуме по DES Агентство национальной безопасности раскрыло ряд критериев проектирования S-блоков. Но это не смогло снять всех подозрений, и спор продолжился.

В литературе про S-блоки писались удивительные вещи. Последние три бита результата четвертого S-блока могут быть получены тем же способом, что и первые, при помощи дополнения некоторых из входных битов. Различные, но тщательно подобранные входные данные для S-блоков могут давать одинаковый результат. Можно получить результат одного этапа DES, меняя биты только в трех соседних S-блоках. Шамир заметил, что элементы S-блоков, казалось, были несколько неустойчивы, но не собирался использовать эту неустойчивость для вскрытия. (Он упомянул об особенности пятого S-блока, но только спустя восемь лет линейный криптоанализ воспользовался этой особенностью.) Другие исследователи показали, что для получения S-блоков с наблюдаемыми характеристиками могли использоваться общеизвестные принципы проектирования.

Дополнительные результаты

Были предприняты и другие попытки криптоанализировать DES. Один из криптографов искал закономерности, используя спектральные тесты. Другие анализировали последовательность линейных множителей, но их вскрытие потерпело неудачу после восьми этапов. Неопубликованное вскрытие, выполненное в 1987 году Дональдом Дэвисом (Donald Davies), использовало способ, с помощью которого перестановка с расширением повторяет биты в соседних S-блоках, это вскрытие также оказалось бесполезным после восьми этапов.

2.1.4. Дифференциальный и линейный криптоанализ

Дифференциальный криптоанализ

Воспользуемся данными [16,17]. В 1990 году Эли Бихам и Ади Шамир ввели понятие **дифференциального криптоанализа**. Это был новый, ранее неизвестный метод криптоанализа. Используя этот метод, Бихам и Шамир нашли способ вскрытия DES с использованием выбранного открытого текста, который был эффективнее вскрытия грубой силой.

Дифференциальный криптоанализ работает с **парами шифротекстов,** открытые тексты которых содержат определенные отличия. Метод анализирует эволюцию этих отличий в процессе прохождения открытых текстов через этапы DES при шифровании одним и тем же ключом.

Просто выберем пару открытых текстов с фиксированным различием. Можно выбрать два открытых текста случайным образом, лишь бы они отличались друг от друга определенным образом, криптоаналитику даже не нужно знать их значений. (Для DES термин "различие" определяется с помощью XOR. Для других алгоритмов этот термин может определяться по другому.) Затем, используя различия в получившихся шифротекстах, присвоим различные вероятности различным ключам. В процессе дальнейшего анализа следующих пар шифротекстов один из ключей станет наиболее вероятным. Это и есть правильный ключ.

Подробнее см. раздел 2.

Реальные критерии проектирования

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

— 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.

Эта работа продолжала обсуждение критериев. Сегодня совсем нетрудно генерировать S-блоки, но в начале 70-х это было нелегкой задачей. Тачмен говорил, что программы, готовившие S-блоки, работали месяцами.

2.1.5. Варианты DES

Многократный DES

В ряде реализаций DES используется трехкратный DES. Так как DES е является группой, полученный шифротекст гораздо сложнее вскрыть, используя исчерпывающий поиск: 2^{112} попыток вместо 2^{56}.

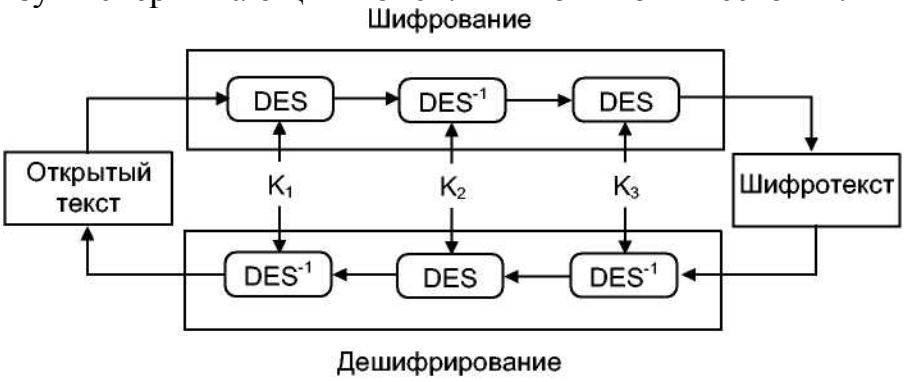


Рис. 19 Трехкратный DES.

DES с независимыми подключами

Другой возможностью является использование различных подключен на каждом этапе, не создавая их из одного 56-битового ключа. Так как на каждом из 16 этапов используется 48 битов ключа, то длина ключа для такого варианта составит 768 битов. Такой вариант резко увеличивает сложность вскрытия алгоритма грубой силой, сложность такого вскрытия составит 2^{768}.

Однако возможно использование вскрытия "встреча посередине". Сложность такого вскрытия уменьшается до 2^{384}, что, тем не менее, вполне достаточно для обеспечения любой мыслимой безопасности.

Хотя независимые подключи мешают линейному криптоанализу, этот вариант чувствителен к дифференциальному криптоанализу и может быть вскрыт с помощью 2^{61} выбранных открытых текстов. По видимому, никакая модификация распределения ключей не сможет намного усилить DES.

DESX

DESX - это вариант DES, разработанный RSA Data Security, Inc., и включенный в 1986 году в программу обеспечения безопасности электронной почты MailSafe, а в 1987 году в набор BSAFE. DESX использует метод, называемый отбеливанием, для маскировки входов и выходов DES. Кроме 56-битового ключа DES в DESX используется дополнительный 64-битовый ключ отбеливания. Эти 64 бита используются для выполнения операции XOR с блоком открытого текста перед первым этапом DES. Дополнительные 64 бита, являющиеся результатом применения однонаправленной функции к полному 120-битовому ключу DESX, используются для выполнения XOR с шифротекстом, полученным в результате последнего этапа. По сравнению с DES отбеливание значительно повышает устойчивость DESX к вскрытию грубой силой, вскрытие требует $(2^{120})/n$ операций при n известных открытых текстах. Также повышается устойчивость к дифференциальному и линейному криптоанализу, для вскрытия потребуется 2^{61} выбранных и 2^{60} известных открытых текстов, соответственно.

CRYPT(3)

CRYPT(3) представляет собой вариант DES, используемый в системах UNIX. Он в основном используется в качестве однонаправленной функции для паролей, но иногда может быть использован и для шифрования. Различие между CRYPT(3) и DES состоит в том, что в CRYPT(3) включена независимая от ключа перестановка с расширением с 2^{12} вариантами. Это сделано для того, чтобы для создания аппаратного устройства вскрытия паролей нельзя было использовать промышленные микросхемы DES.

Обобщенный DES

Обобщенный DES (Generalized DES, GDES) был спроектирован для ускорения DES и повышения устойчивости алгоритма. Общий размер блока увеличился, а количество вычислений осталось неизменным.

На рисунке 20 показана поблочная диаграмма GDES. GDES работает с блоками открытого текста переменной длины. Блоки шифрования делятся на q 32-битовых подблоков, точное число которых зависит от полного размера блока (который по идее может меняться, но фиксирован для конкретной реализации). В общем случае q равно размеру блока, деленному на 32.

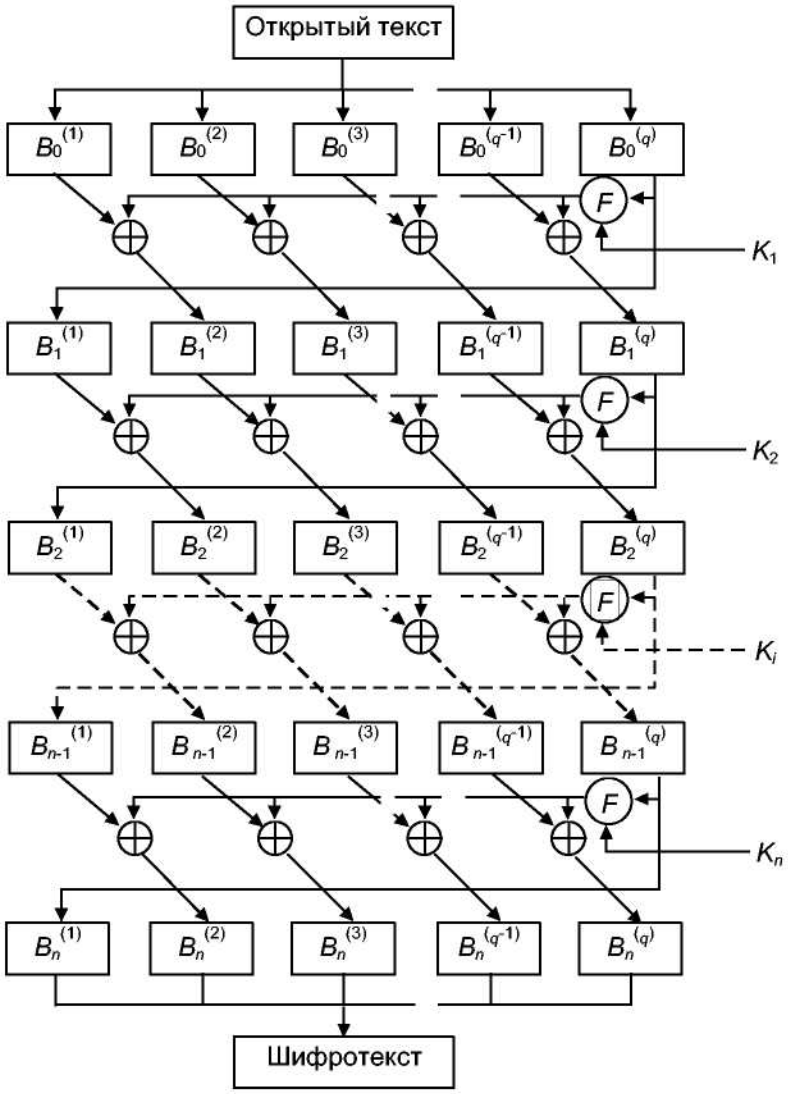


Рис. 20 GDES [17].

Функция f для каждого этапа рассчитывается один раз для крайнего правого блока. Результат при помощи операции XOR объединяется со всеми остальными частям, которые затем циклически смещаются направо. GDES использует переменное число этапов *n.* В последний этап внесено незначительное изменение, чтобы процессы шифрования и дешифрирования отличались только порядком подключей (точно также, как в DES). Действительно, если q = 2 и n = 16, то описанный алгоритм превращается в DES.

Бихам и Шамир показали, что дифференциальный криптоанализ вскрывает GDES с q = 8 и n=16 с помощью всего шести выбранных открытых текстов. При использовании независимых подключей требуется 16 выбранных открытых текстов. GDES с q = 8 и n = 22 вскрывается с помощью всего 48 выбранных открытых текстов, а для вскрытия GDES с q = 8 и n = 31 требуется всего 500000 выбранных открытых текстов. Даже GDES с q = 8 и n = 64 слабее, чем DES - для его вскрытия нужно только 249 выбранных открытых текстов. Действительно, любая более быстрая, чем DES, схема GDES является также и менее безопасной.

Недавно появился еще один вариант этой схемы. Возможно, он не более безопасен, чем оригинальный GDES. Общем случае любой вариант DES с большими блоками, который быстрее DES, скорее всего менее безопасен по сравнению с DES.

DES с измененными S-блоками

Другие модификации DES связаны с S-блоками. В некоторых проектах используется переменный порядок S-блоков. Другие разработчики меняют содержание самих S-блоков. Бихам и Шамир показали, что построение S-блоков и даже их порядок оптимальны с точки зрения устойчивости к дифференциальному криптоанализу:

Изменение порядка восьми S-блоков DES (без изменения их значений) также значительно ослабляет DES: DES с 16 этапами и конкретным измененным порядком вскрывается примерно за 2^{38} шагов. ... Доказано, что DES со случайными S-блоками вскрыть очень легко. Даже минимальное изменение одного из элементов S-блоков DES может снизить устойчивость DES к вскрытию.

S-блоки DES не были оптимизированы против линейного криптоанализа. Существуют и лучшие S-блоки, чем предлагаемые в DES, но бездумная замена S-блоков новыми - не самая лучшая идея.

RDES

RDES - это модификация, в которой в конце каждого этапа обмениваются местами правая и левая половины с использованием зависимой от ключа перестановки. Обмены местами фиксированы и зависят только от ключа. Это означает, что может быть 15 обменов, зависимых от ключа, и 2^{15} возможных вариантов, а также что эта модификация не устойчива по отношению к дифференциальному криптоанализу. У RDES большое количество слабых ключей. Действительно, почти каждый ключ слабее, чем типичный ключ DES. Использовать эту модификацию нельзя.

Лучшей является идея выполнять обмен местами только в пределах правой половины и в начале каждого этапа. Другой хорошей идеей является выполнение обмена в зависимости от входных данных, а не как статической функции ключа. Существует множество возможных вариантов. В RDES-1 используется зависящая от данных перестановка 16-битовых слов в начале каждого этапа. В RDES-2 применяется зависящая от данных перестановка байтов в начале каждого этапа после 16-битовых перестановок, аналогичных RDES-1. Развитием этой идеи является RDES-4, и т.д. RDES-1 устойчив и к дифференциальному, и к линейному криптоанализу. По видимому, RDES-2 и последующие варианты достаточно хороши.

Группа корейских исследователей под руководством Кванджо Кима (Kwangjo Kim) попыталась найти набор S-блоков, оптимально устойчивых и против дифференциального, и против линейного криптоанализа. Их первая попытка, известная как ,s^2DES, оказалась менее устойчивой, чем DES, против дифференциального криптоанализа. Следующий вариант, s^3DES, оказался

менее устойчив, чем DES, к линейному криптоанализу. Бихам предложил незначительно изменить алгоритм, чтобы сделать ,s^{3}DES безопасным по отношению и к дифференциальному, и к линейному криптоанализу. Исследователи вернулись к своим компьютерам и разработали улучшенную технику проектирования S-блоков. Они предложили ,s^{4}DES, а затем ,s^{5}DES.

DES с S-блоками, зависящими от ключа

Линейный и дифференциальный криптоанализ работают только, если аналитику известно строение S-блоков. Если S-блоки зависят от ключа и выбираются криптографически сильным методом, то линейный и дифференциальный криптоанализ значительно усложнятся. Хотя надо помнить, что даже у хранящихся в секрете случайно созданных S-блоков очень плохие дифференциальные и линейные характеристики.

Табл.12 S-блоки s3DES (с обращенными S-блоками 1 и 2)

S-блок 1:															
13	14		3	10	4	7	9	11	8	12	6	1	15	2	5
8	2	11	13	4	1	14	7	5	15	0	3	10	6	9	12
14	9	3	10	0	7	13	4	8	5	6	15	11	12	1	2
1	4	14	7	11	13	8	2	6	3	5	10	12	0	15	9
S-блок 2:															
15	8	3	14	4	2	9	5	0	11	10	1	13	7	6	12
6	15	9	5	3	12	10	0	13	8	4	11	14	2	1	7
9	14	5	8	2	4	15	3	10	7	6	13	1	11	12	0
10	5	3	15	12	9	0	6	1	2	8	4	11	14	7	13
S-блок З:															
13	3	11	5	14	8	0	6	4	15	1	12	7	2	10	9
4	13	1	8	7	2	14	11	15	10	12	3	9	5	0	6
6	5	8	11	13	14	3	0	9	2	4	1	10	7	15	12
1	11	7	2	8	13	4	14	6	12	10	15	3	0	9	5
S-блок 4:															
9	0	7	11	12,	5	10	6	15	3	1	14	2	8	4	13
5	10	12	6	0	15	3	9	8	13	11	1	7	2	14	4
10	7	9	12	5	0	6	11	3	14	4	2	8	13	15	1
3	9	15	0	6	10	5	12	14	2	1	7	13	4	8	11
S-блок 5:															
5	15	9	10	0	3	14	4	2	12	7	1	13	6	8	11
6	9	3	15	5	12	0	10	8	7	13	4	2	11	14	1
15	0	10	9	3	5	4	14	8	11	1	7	6	12	13	2
12	5	0	6	15	10	9	3	7	2	14	11	8	1	4	13
S-блок 6:															
4	3	7	10	9	0	14	13	15	5	12	6	2	11	1	8

14	13	11	4	2	7	1	8	9	10	5	3	15	0	12	6
13	0	10	9	4	3	7	14	1	15	6	12	8	5	11	2
1	7	4	14	11	8	13	2	10	12	3	5	6	15	0	9
S-блок 7:															
4	10	15	12	2	9	1	6	11	5	0	3	7	14	13	8
10	15	6	0	5	3	12	9	1	8	11	13	14	4	7	2
2	12	9	6	15	10	4	1	5	11	3	0	8	7	14	13
12	6	3	9	0	5	10	15	2	13	4	14	7	11	1	8
S-блок 8:															
13	10	0	7	3	9	14	4	2	15	12	1	5	6	11	8
2	7	13	1	4	14	11	8	15	12	6	1O	9	5	0	3
4	13	14	0	9	3	7	10	1	8	2	11	15	5	12	6
8	11	7	14	2	4	13	1	6	5	9	0	12	15	3	10

Вот как можно использовать 48 дополнительных битов ключа для создания S-блоков, устойчивых как к линейному, так и к дифференциальному криптоанализу.

(1) Изменить порядок S-блоков DES: 24673158.

(2) Выбрать 16 из оставшихся битов ключа. Если первый бит 1, обменять местами первые и последние два ряда S-блока 1. Если второй бит 1, обменять местами первые и последние восемь столбцов S-блока 1. Повторить то же самое для третьего и четвертого битов и S-блока 2. Повторить то же самое для S-блоков с 3 по 8.

(3) Взять оставшиеся 32 бита ключа. Выполнить XOR первых четырех битов с каждым элементом S-блока 1, XOR следующих четырех битов с каждым элементом S-блока 2, и так далее.

Сложность вскрытия такой системы с помощью дифференциального криптоанализа составит 251, с помощью линейного криптоанализа - 2^{53}. Сложность исчерпывающего перебора составит 2102.

Что хорошо в этом варианте DES так это то, что он может быть реализован в существующей аппаратуре. Различные поставщики микросхем DES продают микросхемы DES с возможностью загрузки S-блоков. Можно реализовать любой способ генерации S-блоков вне микросхемы и затем загрузить их в нее. Для дифференциального и линейного криптоанализа нужно так много известных или выбранных открытых текстов, что эти способы вскрытия становятся неосуществимыми. Вскрытие грубой силой также трудно себе представить, не поможет никакое увеличение скорости.

Насколько безопасен сегодня DES?

Ответ одновременно и прост, и труден. При простом ответе учитывается только длина ключа. Машина для вскрытия DES грубой силой, способная найти ключ в среднем за 3.5 часа, в 1993 году стоила 1 миллион долларов. DES используется очень широко, и наивно было бы предполагать,

что NSA и аналогичные организации в других странах не построили по такому устройству. Стоимость уменьшается в 5 раз каждые 10 лет. С течением времени DES будет становиться все менее и менее безопасным.

Для трудного ответа нужно попытаться оценить криптоаналитические методы. Дифференциальный криптоанализ был известен в NSA задолго до середины 70-х, когда DES впервые стал стандартом. Наивно считать, что с тех пор теоретики NSA ничего не делали, почти наверняка они разработали новые криптоаналитические методы, которые можно использовать против DES. Но фактов нет, одни слухи.

Винн Шварцтау (Winn Schwartau) пишет, что NSA построило огромную параллельную машину для вскрытия DES уже в середине 80-х. По крайней мере, одна такая машина была построена в Harris Corp. С использованием Cray Y-MP. Предположительно существует ряд алгоритмов, которые на несколько порядков уменьшают сложность вскрытия DES грубой силой. Контекстные алгоритмы, основанные на внутренней работе DES, позволяют отбросить ряд ключей, используя частичные решения. Статистические алгоритмы уменьшают эффективную длину ключа еще сильнее. Другие алгоритмы также проверяют вероятные ключи - слова, печатаемые последовательности ASCII, и т.д). По слухам NSA может вскрыть DES за время от 3 до 15 минут, в зависимости от того коков будет выполненный объем предварительной обработки. И каждая такая машина стоит порядка 50000 долларов.

Согласно другим слухам, если у NSA есть большое количество открытых текстов и шифротекстов, его эксперты могут выполнить некоторые статистические расчеты и затем считать ключ из архива на оптических дисках.

Рекомендуется использовать схему Бихама для зависящих от ключа S-блоков. Она может быть легко реализована программно или аппаратно (с помощью микросхем с загружаемыми S-блоками), и не приводит к потере эффективности по сравнению с DES. Эта схема повышает устойчивость алгоритма к вскрытию грубой силой, усложняет дифференциальный и линейный криптоанализ и заставляет NSA столкнуться с алгоритмом, по крайней мере таким же сильным как DES, но другим.

2.2. АЛГОРИТМ ШИФРОВАНИЯ ДАННЫХ ГОСТ 28147-89

Данный материал изложен в соответствии с источником [10]. Описание стандарта шифрования Российской Федерации содержится в документе, озаглавленном «Алгоритм криптографического преобразования данных ГОСТ 28147-89». То, что в его названии вместо термина «шифрование» фигурирует более общее понятие «криптографическое преобразование», вовсе не случайно. Помимо нескольких тесно связанных между собой процедур шифрования, в документе описан один построенный на общих

принципах с ними алгоритм выработки имитовставки. Последняя является не чем иным, как криптографической контрольной комбинацией, то есть кодом, вырабатываемым из исходных данных с использованием секретного ключа с целью имитозащиты, или защиты данных от внесения в них несанкционированных изменений.

На различных шагах алгоритмов ГОСТа данные, которыми они оперируют, интерпретируются и используются различным образом. В некоторых случаях элементы данных обрабатываются как массивы независимых битов, в других случаях – как целое число без знака, в третьих – как имеющий структуру сложный элемент, состоящий из нескольких более простых элементов. Поэтому во избежание путаницы следует договориться об используемых обозначениях. Элементы данных в данной статье 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 этот факт обозначается следующим образом: $X=(X_0,X_1,\ldots,X_{n-1})=X_0||X_1||\ldots||X_{n-1}$. Процедура объединения нескольких элементов данных в один называется конкатенацией данных и обозначается символом «||». Естественно, для размеров элементов данных должно выполняться следующее соотношение: $|X|=|X_0|+|X_1|+\ldots+|X_{n-1}|$. 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:

$(X_0,X_1,\ldots,X_{n-1})=X_0||X_1||\ldots||X_{n-1}=X_0+2^{|X0|}(X_1+2^{|X1|}(\ldots(X_{n-2}+2^{|Xn-2|}X_{n-1})\ldots))$.

В алгоритме элемент данных может интерпретироваться как массив отдельных битов, в этом случае биты обозначаем той же самой буквой, что и массив, но в строчном варианте, как показано на следующем примере:

$X=(x_0,x_1,\ldots,x_{n-1})= x_0+2^1\cdot x_1+\ldots+2^{n-1}\cdot x_{n-1}$.

Если над элементами данных выполняется некоторая операция, имеющая логический смысл, то предполагается, что данная операция выполняется над соответствующими битами элементов. Иными словами $A \bullet B =(a_0 \bullet b_0,a_1 \bullet b_1,\ldots,a_{n-1} \bullet b_{n-1})$, где $n=|A|=|B|$, а символом «•» обозначается произвольная бинарная логическая операция; как правило, имеется ввиду операция исключающего или, она же – операция суммирования по модулю 2: $a \oplus b=(a+b) mod 2$.

2.2.1. Логика построения и структура ключевой информации алгоритма

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 упоминаются в данной статье как базовые циклы, чтобы отличать их от всех прочих циклов. Они имеют следующие названия и обозначения, последние приведены в скобках и смысл их будет объяснен позже:

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

В свою очередь, каждый из базовых циклов представляет собой многократное повторение одной единственной процедуры, называемой для определенности далее в настоящей работе основным шагом криптопреобразования.

Таким образом, чтобы разобраться в ГОСТе, 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.

Прежде чем перейти к изучению этих вопросов, следует поговорить о ключевой информации, используемой алгоритмами ГОСТа. В соответствии с принципом Кирхгофа, которому удовлетворяют все современные известные широкой общественности шифры, именно ее секретность обеспечивает секретность зашифрованного сообщения. 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 в настоящей работе он обозначается символом K: $K=\{K_i\}_{0\le i\le 7}$. В ГОСТе элементы ключа используются как 32-разрядные целые числа без знака: $0\le K_i\le 2^{32}$. Таким образом, размер ключа составляет $32\cdot 8=256$ бит или 32 байта.

2. Таблица замен может быть представлена в виде матрицы размера 8x16, содержащей 4-битовые элементы, которые можно представить в виде целых чисел от 0 до 15. Строки таблицы замен называются узлами замен, они должны содержать различные значения, то есть каждый узел замен должен содержать 16 различных чисел от 0 до 15 в произвольном порядке. В настоящей статье таблица замен обозначается символом H: $H = \{H_{i,j}\}_{0 \le j \le 15}^{0 \le i \le 7}$. Таким образом, общий объем таблицы замен равен: 8 узловx16 элементов/узелx4 бита/элемент = 512 бит или 64 байта.

2.2.2. Основной шаг криптопреобразования

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

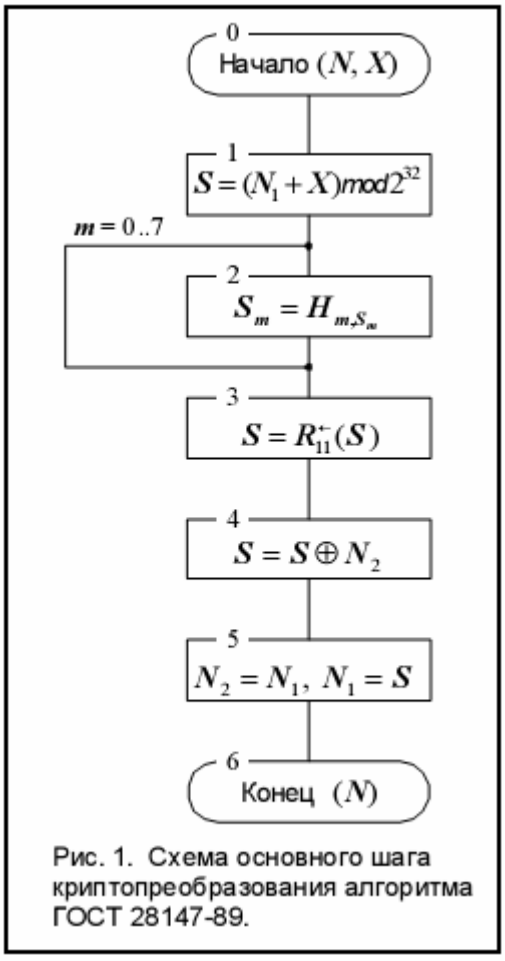


Рис. 1. Схема основного шага криптопреобразования алгоритма ГОСТ 28147-89.

Ниже даны пояснения к алгоритму основного шага:

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-тый по порядку элемент (нумерация с нуля) i-того узла замен (т.е. i-той строки таблицы замен, нумерация также с нуля). Другими словами, в качестве замены для значения блока выбирается элемент из таблицы замен с номером строки, равным номеру заменяемого блока, и номером столбца, равным значению заменяемого блока как 4-битового целого неотрицательного числа. Теперь становится понятным размер таблицы замен: число строк в ней равно числу 4-битовых элементов в 32-битовом блоке данных, то есть восьми, а число столбцов равно числу различных значений 4-битового блока данных, равному шестнадцати.

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.

2.2.3. Базовые циклы криптографических преобразований

Как отмечено в начале настоящей статьи, ГОСТ относится к классу блочных шифров, то есть единицей обработки информации в нем является

блок данных. Следовательно, вполне логично ожидать, что в нем будут определены алгоритмы для криптографических преобразований, то есть для зашифрования, расшифрования и «учета» в контрольной комбинации одного блока данных. Именно эти алгоритмы и называются базовыми циклами ГОСТа, что подчеркивает их фундаментальное значение для построения этого шифра.

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

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

1. Цикл зашифрования 32-З:

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

2. Цикл расшифрования 32-Р:

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

3. Цикл выработки имитовставки 16-З:

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

Каждый из циклов имеет собственное буквенно-цифровое обозначение, соответствующее шаблону «*n*-*X*», где первый элемент обозначения (*n*), задает число повторений основного шага в цикле, а второй элемент обозначения (*X*), буква, задает порядок зашифрования («З») или расшифрования («Р») в использовании ключевых элементов. Этот порядок нуждается в дополнительном пояснении:

Цикл расшифрования должен быть обратным циклу зашифрования, то есть последовательное применение этих двух циклов к произвольному блоку должно дать в итоге исходный блок, что отражается следующим

соотношением: $C_{32\text{-}Р}(C_{32\text{-}З}(T))=T$, где T – произвольный 64-битовый блок данных, $C_X(T)$ – результат выполнения цикла X над блоком данных T. 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 сказанного вытекает одно интересное следствие: свойство цикла быть обратным другому циклу является взаимным, то есть цикл 32-З является обратным по отношению к циклу 32-Р. Другими словами, зашифрование блока данных теоретически может быть выполнено с помощью цикла расшифрования, в этом случае расшифрование блока данных должно быть выполнено циклом зашифрования.

Из двух взаимно обратных циклов любой может быть использован для зашифрования, то 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, в чем нетрудно убедиться, рассмотрев приведенные выше последовательности, поэтому этот порядок в обозначении цикла кодируется той же самой буквой «З». Схемы базовых циклов приведены на рисунках 2а-в.

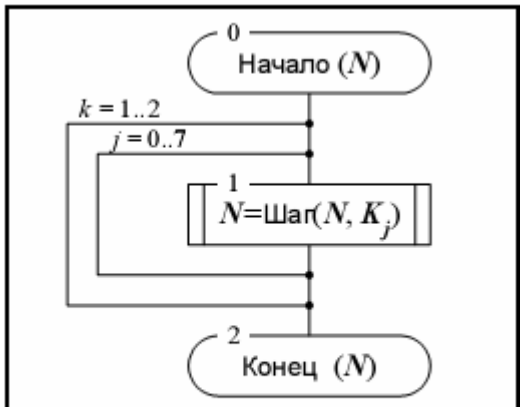


Рис. 2в. Схема цикла выработки имитовставки 16-З.

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, не упомянутое выше: в конце базовых циклов шифрования старшая и младшая часть блока результата меняются местами, это необходимо для их взаимной обратимости.

2.2.4. Основные режимы шифрования

ГОСТ 28147-89 предусматривает три следующих режима шифрования данных:

- простая замена,
- гаммирование,
- гаммирование с обратной связью,
- выработка имитовставки.

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

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

$T_o, T_ш$ – 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: $\left|T_i^o\right| = \left|T_i^ш\right| = 64$ при $1 \le i \le n$, $1 \le \left|T_n^o\right| = \left|T_n^ш\right| \le 64$;

n – число 64-битовых блоков в массиве данных;

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

Теперь опишем основные режимы шифрования:

1. Простая замена.

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

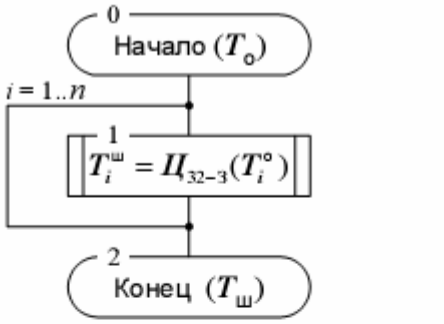


Рис. 3а. Алгоритм зашифрования данных в режиме простой замены.

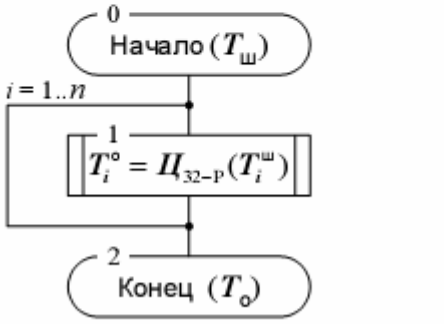


Рис. 3б. Алгоритм расшифрования данных в режиме простой замены.

Размер массива открытых или зашифрованных данных, подвергающийся соответственно зашифрованию или расшифрованию, должен быть кратен 64 битам: $\left|T_i^o\right| = \left|T_i^ш\right| = 64 \cdot n$, после выполнения операции размер полученного массива данных не изменяется.

Режим шифрования простой заменой имеет следующие особенности:

1. Так как блоки данных шифруются независимо друг от друга и от их позиции в массиве, при зашифровании двух одинаковых блоков открытого текста получаются одинаковые блоки 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. Эта задача не так проста, как кажется на первый взгляд. Очевидные решения типа «дополнить неполный блок нулевыми битами» или, более обще, «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.

На первый взгляд, перечисленные выше особенности делают практически невозможным использование режима простой замены, ведь он может применяться только для шифрования массивов данных с размером кратным 64 битам, не содержащим повторяющихся 64-битовых блоков. Кажется, что для любых реальных данных гарантировать выполнение указанных условий невозможно. Это почти так, но есть одно очень важное исключение: вспомните, что размер ключа составляет 32 байта, а размер таблицы замен – 64 байта. Кроме того, наличие повторяющихся 8-байтовых блоков в ключе или таблице замен будет говорить об их весьма плохом качестве, поэтому в реальных ключевых элементах такого повторения быть не может. Таким образом, мы выяснили, что режим простой замены вполне подходит для шифрования ключевой информации, тем более, что прочие режимы для этой цели менее удобны, поскольку требуют наличия дополнительного синхронизирующего элемента данных – синхропосылки (см. следующий раздел). Наша догадка верна, ГОСТ предписывает использовать режим простой замены исключительно для шифрования ключевых данных.

2. Гаммирование.

Как же можно избавиться от недостатков режима простой замены? Для этого необходимо сделать возможным шифрование блоков с размером менее 64 бит и обеспечить зависимость блока шифртекста от его номера, иными словами, рандомизировать процесс шифрования. В ГОСТе это достигается двумя различными способами в двух режимах шифрования, предусматривающих гаммирование. Гаммирование – это наложение (снятие) на открытые (зашифрованные) данные криптографической гаммы, то есть последовательности элементов данных, вырабатываемых с помощью некоторого криптографического алгоритма, для получения зашифрованных (открытых) данных. Для наложения гаммы при зашифровании и ее снятия при расшифровании должны использоваться взаимно обратные 64 для 64-битовых блоков бинарные операции, например, сложение и вычитание по модулю 2 данных. В ГОСТе для этой цели используется операция побитного сложения по модулю 2, поскольку она является обратной самой себе и, к тому же, наиболее просто реализуется аппаратно. 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.

Теперь перейдем непосредственно к описанию режима гаммирования. 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, то есть зашифрованию в режиме простой замены, в результате получаются блоки гаммы. Благодаря тому, что наложение и снятие гаммы осуществляется при помощи одной и той же операции побитового исключающего или, алгоритмы зашифрования и расшифрования в режиме гаммирования идентичны, их общая схема приведена на рисунке 4.

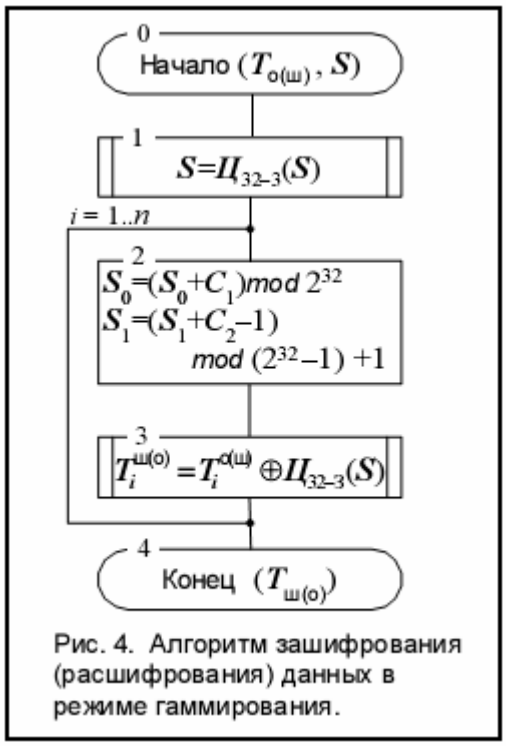


Рис. 4. Алгоритм зашифрования (расшифрования) данных в режиме гаммирования.

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 нашем ГОСТе – начальным заполнением одного из регистров шифрователя. По определенным соображениям разработчики ГОСТа решили использовать для инициализации РГПЧ не непосредственно синхропосылку, а результат ее преобразования по циклу 32-З: $\Omega_0=C_{32-З}(S)$. Последовательность элементов,

вырабатываемых РГПЧ, целиком зависит от его начального заполнения, то есть элементы этой последовательности являются функцией своего номера и начального заполнения РГПЧ:

$\Omega_i=f_i(\Omega_0)$, где $f_i\ (X)=f(f_{i-1}(X))$, $f_0(X)=X$. С учетом преобразования по алгоритму простой замены добавляется еще и зависимость от ключа:

$\Gamma i = C_{32\text{-}3}(\Omega_i)=C_{32\text{-}3}(f_i(\Omega_0))=C_{32\text{-}3}(f_i(C_{32\text{-}3}(S)))=\varphi_i(S,K)$, где Γ_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 подробно рассмотрим РГПЧ, используемый в ГОСТе для генерации элементов гаммы. Прежде всего, надо отметить, что к нему не предъявляются требования обеспечения каких-либо статистических характеристик вырабатываемой последовательности чисел. РГПЧ спроектирован разработчиками ГОСТа исходя из необходимости выполнения следующих условий:

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

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

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

$\Omega_i = (\Omega_i^0, \Omega_i^1), \left|\Omega_i^0\right| = \left|\Omega_i^1\right| = 32, \Omega_{i+1}^0 = \hat{f}(\Omega_i^0), \Omega_{i+1}^1 = \hat{f}(\Omega_i^1)$; фактически, существуют два независимых РГПЧ для старшей и младшей частей блока.

рекуррентные соотношения для старшей и младшей частей следующие:

$\Omega^0_{i+1} = (\Omega^1_i + C_1) \bmod 2^{32}$, где C_1=1010101_{16},

$\Omega^1_{i+1} = (\Omega^1_i + C_2 - 1) \bmod (2^{32} - 1) + 1$, где C_1=1010104_{16}.

Нижний индекс в записи числа означает его систему счисления, таким образом, константы, используемые на данном шаге, записаны в 16-ричной системе счисления.

Второе выражение нуждается в комментариях, так как в тексте ГОСТа приведено нечто другое: $\Omega^1_{i+1} = (\Omega^1_i + C_2) \bmod (2^{32} - 1)$ с тем же значением константы C_2. Но далее в тексте стандарта дается комментарий, что, оказывается, под операцией взятия остатка по модулю там понимается не то же самое, что и в математике. Отличие заключается в том, что согласно ГОСТу, $(2^{32} - 1) \bmod (2^{32} - 1) = 2^{32} - 1$, а не 0. На самом деле, это упрощает реализацию формулы, а математически корректное выражение для нее приведено выше.

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

Схема алгоритма шифрования в режиме гаммирования приведена на рисунке 4, ниже изложены пояснения к схеме:

Шаг 0. Определяет исходные данные для основного шага криптопреобразования: $T_{о(ш)}$– 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», то есть 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–З, 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, более экзотический.

Второй способ имеет гораздо меньшее применение, поскольку сделать синхропосылку предопределенной можно только в том случае, если на данном комплекте ключевой информации шифруется заведомо не более одного массива данных, что бывает в редких случаях. Генерировать синхропосылку синхронно у источника и получателя массива данных также не всегда представляется возможным, поскольку требует жесткой привязки к чему-либо в системе. Так, здравая на первый взгляд идея использовать в качестве синхропосылки в системе передачи зашифрованных сообщений номер передаваемого сообщения не подходит, поскольку сообщение может потеряться и не дойти до адресата, в этом случае произойдет десинхронизация систем шифрования источника и приемника. Поэтому в рассмотренном случае нет альтернативы передаче синхропосылки вместе с зашифрованным сообщением.

С другой стороны, можно привести и обратный пример. Допустим, шифрование данных используется для защиты информации на диске, и реализовано оно на низком уровне, для обеспечения независимого доступа

данные шифруются по секторам. В этом случае невозможно хранить синхропосылку вместе с зашифрованными данными, поскольку размер сектора нельзя изменить, однако ее можно вычислять как некоторую функцию от номера считывающей головки диска, номера дорожки (цилиндра) и номера сектора на дорожке. В этом случае синхропосылка привязывается к положению сектора на диске, которое вряд ли может измениться без переформатирования диска, то есть без уничтожения данных на нем.

Режим гаммирования имеет еще одну интересную особенность. В этом режиме биты массива данных шифруются независимо друг от друга. Таким образом, каждый бит 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 шифртекста на противоположное значение приведет к аналогичному изменению бита открытого текста на противоположный:

$\bar{t}_i^{ш} = t_i^{ш} \oplus 1 = (t_i^{o} \oplus \gamma_i) \oplus 1 = (t_i^{o} \oplus 1) \oplus \gamma_i = \bar{t}_i^{o} \oplus \gamma_i$, где $\bar{t}$ обозначает инвертированное по отношению к *t* значение бита.

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. Это иллюстрирует хорошо известный в криптологии факт, что секретность и аутентичность суть различные свойства криптографических систем. Иными словами, свойства криптосистемы обеспечивать защиту от несанкционированного ознакомления с содержимым сообщения и от несанкционированного внесения изменений в сообщение являются независимыми и лишь в отдельных случаях могут пересекаться. Сказанное означает, что существуют криптографические алгоритмы, обеспечивающие определенную секретность зашифрованных данных и при этом никак не защищающие от внесения изменений и наоборот, обеспечивающие аутентичность данных и никак не ограничивающие возможность ознакомления с ними. По этой причине рассматриваемое свойство режима гаммирования не должно рассматриваться как его недостаток.

3. Гаммирование с обратной связью.

Данный режим очень похож на режим гаммирования и отличается от него только способом выработки элементов гаммы – очередной элемент гаммы вырабатывается как результат преобразования по циклу 32-З предыдущего

блока зашифрованных данных, а для зашифрования первого блока массива данных элемент гаммы вырабатывается как результат преобразования по тому же циклу синхропосылки. Этим достигается зацепление блоков – каждый блок шифртекста в этом режиме зависит от соответствующего и всех предыдущих блоков открытого текста. Поэтому данный режим иногда называется гаммированием с зацеплением блоков. На стойкость шифра факт зацепления блоков не оказывает никакого влияния. Схема алгоритмов за- и расшифрования в режиме гаммирования с обратной связью приведена на рисунке 5 и ввиду своей простоты в комментариях не нуждается.

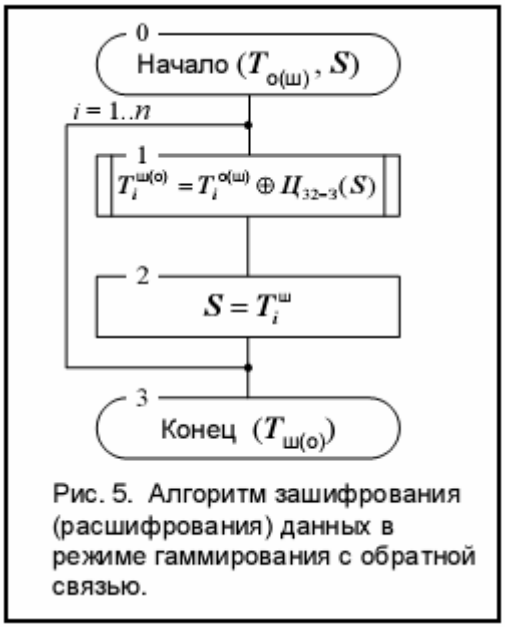


Рис. 5. Алгоритм зашифрования (расшифрования) данных в режиме гаммирования с обратной связью.

Шифрование в режиме гаммирования с обратной связью обладает теми же особенностями, что и шифрование в режиме обычного гаммирования, за исключением влияния искажений шифртекста на соответствующий открытый текст.

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

открытых данных все биты независимо друг от друга с вероятностью 1/2 изменят свои значения.

4. Выработка имитовставки к массиву данных.

В предыдущих разделах мы обсудили влияние искажения шифрованных данных на соответствующие открытые данные. Мы установили, что при расшифровании в режиме простой замены соответствующий блок открытых данных оказывается искаженным непредсказуемым образом, а при расшифровании блока в режиме гаммирования изменения предсказуемы. В режиме гаммирования с обратной связью искаженными оказываются два блока, один предсказуемым, а другой непредсказуемым образом.

При анализе данной ситуации необходимо учесть то, что непредсказуемые изменения в расшифрованном блоке данных могут быть обнаружены только в случае избыточности этих самых данных, причем чем больше степень избыточности, тем вероятнее обнаружение искажения. Очень большая избыточность имеет место, например, для текстов на естественных и искусственных языках, в этом случае факт искажения обнаруживается практически неизбежно. Однако в других случаях, например, при искажении сжатых звуковых образов, мы получим просто другой образ, который сможет воспринять наше ухо. Искажение в этом случае останется необнаруженным, если, конечно, нет априорной информации о характере звука. Вывод здесь такой: поскольку способность некоторых режимов шифрования обнаруживать искажения, внесенные в шифрованные данные, существенным образом опирается на наличие и степень избыточности шифруемых данных, эта способность не является имманентным свойством соответствующих режимов и не может рассматриваться как их достоинство.

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

- вычисление имитовставки для заданного открытого массива информации;

- подбор открытых данных под заданную имитовставку.

Схема алгоритма выработки имитовставки приведена на рисунке 6.

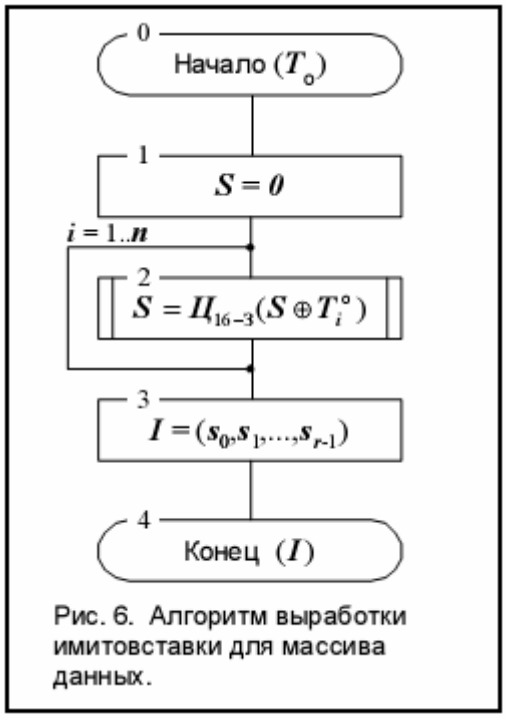


Рис. 6. Алгоритм выработки имитовставки для массива данных.

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

имитовставки надо принимать во внимание, что вероятность успешного навязывания ложных данных равна величине $2^{-|I|}$ на одну попытку подбора, если в распоряжении злоумышленника нет более эффективного метода подбора, чем простое угадывание. При использовании имитовставки

размером 32 бита эта вероятность равна $2^{-32}=0.23\cdot10^{-9}$.

2.2.5. Криптографическая стойкость ГОСТа.

При выборе криптографического алгоритма для использования в конкретной разработке одним из определяющих факторов является его стойкость, то есть устойчивость к попыткам противоположной стороны его раскрыть. Вопрос о стойкости шифра при ближайшем рассмотрении сводится к двум взаимосвязанным вопросам:

- можно ли вообще раскрыть данный шифр;
- если да, то насколько это трудно сделать практически.

Шифры, которые вообще невозможно раскрыть, называются абсолютно или теоретически стойкими. Существование подобных шифров доказывается теоремой Шеннона, однако ценой этой стойкости является необходимость использования для шифрования каждого сообщения ключа, не меньшего по размеру самого сообщения. Во всех случаях за исключением ряда особых эта цена чрезмерна, поэтому на практике в основном

используются шифры, не обладающие абсолютной стойкостью. Таким образом, наиболее употребительные схемы шифрования могут быть раскрыты за конечное время или, что точнее, за конечное число шагов, каждый из которых является некоторой операцией над числами. Для них наиважнейшее значение имеет понятие практической стойкости, выражающее практическую трудность их раскрытия. Количественной мерой этой трудности может служить число элементарных арифметических и логических операций, которые необходимо выполнить, чтобы раскрыть шифр, то есть чтобы для заданного шифртекста с вероятностью, не меньшей заданной величины, определить соответствующий открытый текст. При этом в дополнении к дешифруемому массиву данных криптоаналитик может

располагать блоками открытых данных и соответствующих им зашифрованных данных или даже возможностью получить для любых выбранных им открытых данных соответствующие зашифрованные данные – в зависимости от перечисленных и многих других неуказанных условий различают отдельные виды криптоанализа.

Все современные криптосистемы построены по принципу Кирхгоффа, то есть секретность зашифрованных сообщений определяется секретностью ключа. Это значит, что даже если сам алгоритм шифрования известен криптоаналитику, тот тем не менее не в состоянии расшифровать сообщение, если не располагает соответствующим ключом. Все классические блочные шифры, в том числе DES и ГОСТ, соответствуют этому принципу и спроектированы таким образом, чтобы не было пути вскрыть их более эффективным способом, чем полным перебором по всему ключевому пространству, т.е. по всем возможным значениям ключа. Ясно, что стойкость таких шифров определяется размером используемого в них ключа.

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.

Общеизвестно, что шифр ГОСТ 28147-89 является представителем целого семейства шифров, построенных на одних и тех же принципах. Самым известным его «родственником» является алгоритм DES. Все эти шифры, подобно ГОСТу, содержат алгоритмы трех уровней. В основе всегда лежит некий «основной шаг», на базе которого сходным образом строятся «базовые циклы», и уже на их основе построены практические процедуры шифрования и выработки имитовставки.

2.2.6. Требования к качеству ключевой информации и источники ключей

Не все ключи и таблицы замен обеспечивают максимальную стойкость шифра. Для каждого алгоритма шифрования существуют свои критерии оценки ключевой информации. Так, для алгоритма DES известно существование так называемых «слабых ключей», при использовании которых связь между открытыми и зашифрованными данными не маскируется достаточным образом, и шифр сравнительно просто вскрывается.

Исчерпывающий ответ на вопрос о критериях качества ключей и таблиц замен ГОСТа если и можно вообще где-либо получить, то только у разработчиков алгоритма. Соответствующие данные не были опубликованы в открытой печати. Однако согласно установленному порядку, для шифрования информации, имеющей гриф, должны быть использованы ключевые данные, полученные от уполномоченной организации. Косвенным образом это может свидетельствовать о наличии методик проверки ключевых данных на «вшивость». Сам факт существования слабых ключевых данных в Российском стандарте шифрования не вызывает сомнения. Очевидно, нулевой ключ и тривиальная таблица замен, по которой любое значение заменяется но него самого, являются слабыми, при использовании хотя бы одного из них шифр достаточно просто взламывается, каков бы ни был второй ключевой элемент.

Как уже было отмечено выше, критерии оценки ключевой информации недоступны, однако на их счет все же можно высказать некоторые общие соображения:

1. Ключ должен являться массивом статистически независимых битов, принимающих с равной вероятностью значения 0 и 1. При этом некоторые конкретные значения ключа могут оказаться «слабыми», то есть шифр может не обеспечивать заданный уровень стойкости в случае их использования. Однако, предположительно, доля таких значений в общей массе всех возможных ключей ничтожно мала. Поэтому ключи, выработанные с помощью некоторого датчика истинно случайных чисел, будут качественными с вероятностью, отличающейся от единицы на ничтожно малую величину. Если же ключи вырабатываются с помощью генератора псевдослучайных чисел, то используемый генератор должен обеспечивать указанные выше статистические характеристики, и, кроме того, обладать высокой криптостойкостью, не меньшей, чем у самого ГОСТа. Иными словами, задача определения отсутствующих членов вырабатываемой

генератором последовательности элементов не должна быть проще, чем задача вскрытия шифра. Кроме того, для отбраковки ключей с плохими статистическими характеристиками могут быть использованы различные статистические критерии. На практике обычно хватает двух критериев, – для проверки равновероятного распределения битов ключа между значениями 0 и 1 обычно используется критерий Пирсона («хи квадрат»), а для проверки независимости битов ключа – критерий серий. Об упомянутых критериях можно прочитать в учебниках или справочниках по математической статистике.

2. Таблица замен является долговременным ключевым элементом, то есть действует в течение гораздо более длительного срока, чем отдельный ключ. Предполагается, что она является общей для всех узлов шифрования в рамках одной системы криптографической защиты. Даже при нарушении конфиденциальности таблицы замен стойкость шифра остается чрезвычайно высокой и не снижается ниже допустимого предела. К качеству отдельных узлов замен можно предъявить приведенное ниже требование. Каждый узел замен может быть описан четверкой логических функций, каждая из которых имеет четыре логических аргумента. Необходимо, чтобы эти функции были достаточно сложными. Это требование сложности невозможно выразить формально, однако в качестве необходимого условия можно потребовать, чтобы соответствующие логические функции, записанные в минимальной форме (т.е. с минимально возможной длиной выражения) с использованием основных логических операций, не были короче некоторого необходимого минимума. В первом и очень грубом приближении это условие может сойти и за достаточное. Кроме того, отдельные функции в пределах всей таблицы замен должны отличаться друг от друга в достаточной степени. На практике бывает достаточно получить узлы замен как независимые случайные перестановки чисел от 0 до 15, это может быть практически реализовано, например, с помощью перемешивания колоды из шестнадцати карт, за каждой из которых закреплено одно из значений указанного диапазона.

Необходимо отметить еще один интересный факт относительно таблицы замен. Для обратимости циклов шифрования «32-З» и «32-Р» не требуется, чтобы узлы замен были перестановками чисел от 0 до 15. Все работает даже в том случае, если в узле замен есть повторяющиеся элементы, и замена, определяемая таким узлом, необратима, однако в этом случае снижается стойкость шифра. Почему это именно так, убедиться несложно.

2.3. АЛГОРИТМ IDEA

Первый вариант шифра IDEA, предложенный Ксуеджа Лай (Xuejia Lai) и Джеймсом Масси (James Massey), появился в 1990 году. Он назывался PES (Proposed Encryption Standard, предложенный стандарт шифрования). В следующем году, после демонстрации Бихамом и Шамиром возможностей дифференциального криптоанализа, авторы усилили свой шифр против такого вскрытия и назвали новый алгоритм IPES (Improved Proposed Encryption Standard, улучшенный предложенный стандарт шифрования). В 1992 году название IPES было изменено на IDEA (International Data Encryption Algorithm, международный алгоритм шифрования данных).

IDEA основывается на некоторых впечатляющих теоретических положениях и, хотя криптоанализ добился некоторых успехов в отношении вариантов с уменьшенным количеством этапов, алгоритм все еще кажется сильным.

Его сегодняшняя известность объясняется тем, что он является частью PGP.

Обзор IDEA

IDEA является блочным шифром, он работает с 64-битовыми блоками открытого текста. Длина ключа - 128 битов. Для шифрования и дешифрирования используется один и тот же алгоритм.

Как и другие, уже рассмотренные блочные шифры IDEA использует и запутывание, и рассеяние. Философия, лежащая в основе проекта, представляет собой "объединение операций из различных алгебраических групп". Смешиваются три алгебраические группы, и все они могут быть легко реализованы как аппаратно, так и программно:

— XOR

— Сложение по модулю 2^{16}

— Умножение по модулю $2^{16} + 1$. (Это операцию можно рассматривать как S-блок IDEA.)

Все эти операции (а в алгоритме используются только они, перестановки на битовом уровне не применяются) работают с 16-битовыми подблоками. Этот алгоритм даже эффективнее на 16-битовых процессорах.

Описание IDEA

Схема IDEA представлена на рисунке. 64-битовый блок данных делится на четыре 16-битовых подблока: X_1, X_2, X_3 и X_4. Эти четыре подблока становятся входными данными для первого этапа алгоритма. Всего в алгоритме восемь этапов. На каждом этапе четыре подблока подвергаются операциям XOR, сложениям и умножениям друг с другом и с шестью 16-битовыми подключами. Между этапами обмениваются местами второй и третий подблоки. Наконец четыре подблока объединяются с четырьмя подключами в окончательном преобразовании.

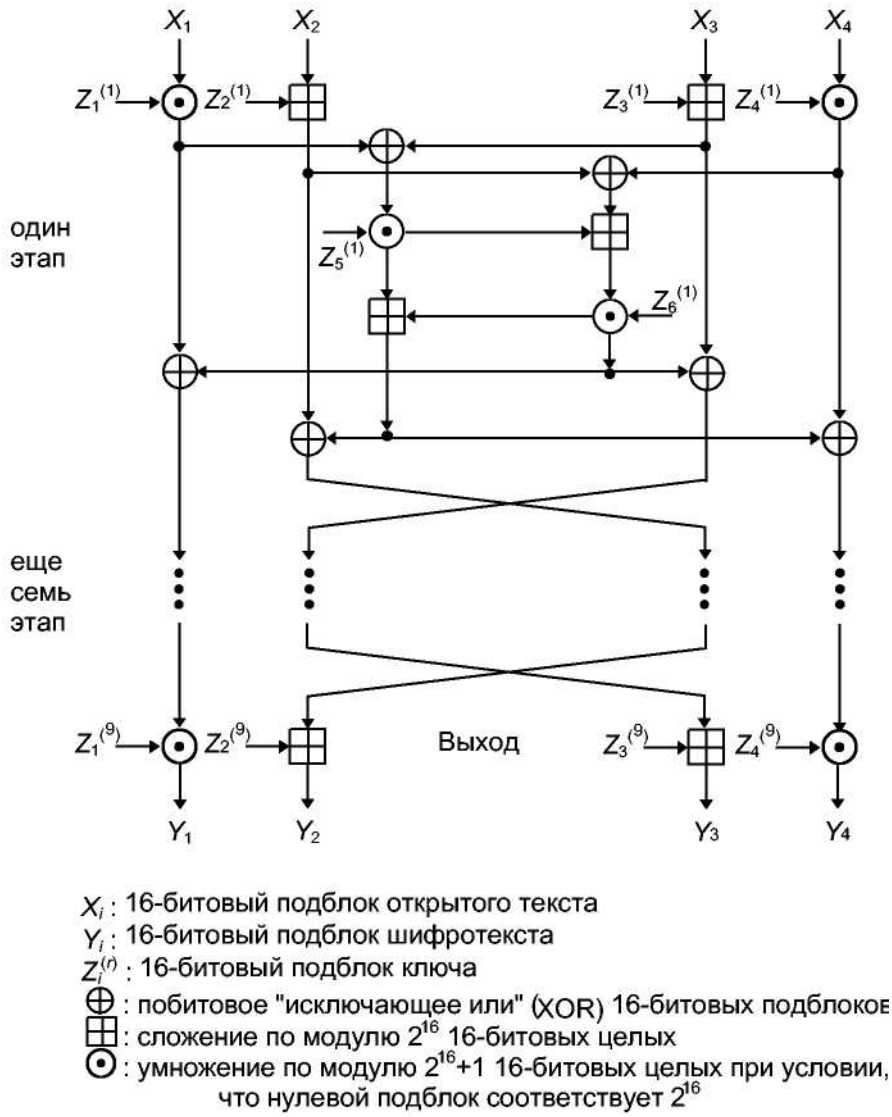


Рис. 23 IDEA.

На каждом этапе события происходят в следующей последовательности:

(1) Перемножаются X_1 и первый подключ.
(2) Складываются X_2 и второй подключ.
(3) Складываются X_3 и третий подключ.
(4) Перемножаются X_4 и четвертый подключ.
(5) Выполняется XOR над результатами этапов (1) и (3).
(6) Выполняется XOR над результатами этапов (2) и (4).
(7) Перемножаются результаты этапа (5) и пятый подключ.
(8) Складываются результаты этапов (6) и (7).
(9) Перемножаются результаты этапа (8) и шестой подключ.
(10) Складываются результаты этапов (7) и (9).
(11) Выполняется XOR над результатами этапов (1) и (9).
(12) Выполняется XOR над результатами этапов (3) и (9).
(13) Выполняется XOR над результатами этапов (1) и (10).
(14) Выполняется XOR над результатами этапов (4) и (10).

Выходом этапа являются четыре подблока - результаты действий (11), (12), (13) и (14). Поменяйте местами два внутренних подблока (но не в последнем этапе), и вы получите исходные данные для следующего этапа.

После восьмого этапа выполняется заключительное преобразование:

(1) Перемножаются X_1 и первый подключ.

(2) Складываются X_2 и второй подключ.

(3) Складываются X_3 и третий подключ.

(4) Перемножаются X_4 и четвертый подключ.

Наконец четыре подблока снова соединяются, образуя шифротекст.

Также несложно создавать подключи. Алгоритм использует 52 из них (шесть для каждого из восьми этапов и еще четыре для заключительного преобразования). Сначала 128-битовый ключ делится на восемь 16-битовых подключей. Это первые восемь подключей алгоритма (шесть для первого этапа и два - для второго). Затем ключ циклически сдвигается налево на 25 битов и снова делится на восемь подключей. Первые четыре используются на этапе 2, а оставшиеся четыре - на этапе 3. Ключ циклически сдвигается налево на 25 битов для получения следующих восьми подключей, и так до конца алгоритма.

Дешифрирование выполняется точно также за исключением того, что подключи инвертируются и слегка изменяются. Подключи при дешифрировании представляют собой обратные значения ключей шифрования по отношению к операциям либо сложения, либо умножения. (Для IDEA подблоки, состоящие из одних нулей, считаются равными $2^{16} = -1$ для умножения по модулю $2^{16} + 1$, следовательно, обратным значением 0 относительно умножения является 0.) Эти вычисления могут занять некоторое время, но их нужно выполнить один раз для каждого ключа дешифрирования.

Скорость IDEA

Программные реализации IDEA примерно в два раза быстрее, чем DES. На компьютере с i386/33 МГц IDEA шифрует данные со скоростью 880 Кбит/с, а на компьютере с i486/33 МГц - со скоростью 2400 Кбит/с. IDEA должен был быть побыстрее, но умножения - недешевое удовольствие. Умножение двух 32-битовых чисел на процессоре i486 занимает 40 тактов (10 на процессоре Pentium).

Реализация PES на базе СБИС шифрует данные со скоростью 55 Мбит/с при тактовой частоте 25 МГц. Другая СБИС, разработанная ETH Zurich и состоящая из of 251000 транзисторов на кристалле площадью 107.8 мм^2, шифрует данные с помощью алгоритма IDEA со скоростью 177 Мбит/с при тактовой частоте 25 МГц.

Криптоанализ IDEA

Длина ключа IDEA равна 128 битам - более чем в два раза длиннее ключа DES. При условии, что наиболее эффективным является вскрытие грубой силой, для вскрытия ключа потребуется 2^{128} (10^{38}) шифрований. Соз-

дайте микросхему, которая может проверять миллиард ключей в секунду, объедините миллиард таких микросхем, и вам потребуется 10^{13} лет для решения проблемы - это больше, чем возраст вселенной. 10^{24} таких микросхем могут найти ключ за день, но во вселенной не найдется столько атомов кремния, чтобы построить такую машину.

Может быть вскрытие грубой силой - не лучший способ вскрытия IDEA. Алгоритм все еще слишком нов, чтобы можно было говорить о каких-то конкретных криптографических результатах. Разработчики сделали все возможное, чтобы сделать алгоритм устойчивым к дифференциальному криптоанализу. Они определили понятие марковского шифра и продемонстрировали, что устойчивость к дифференциальному криптоанализу может быть промоделирована и оценена количественно. Лай (Lai) утверждал (он привел подтверждение, но не доказательство), что IDEA устойчив к дифференциальному криптоанализ уже после 4 из 8 этапов. Согласно Бихаму, его попытка вскрыть IDEA с помощью криптоанализа со связанными ключами также не увенчалась успехом.

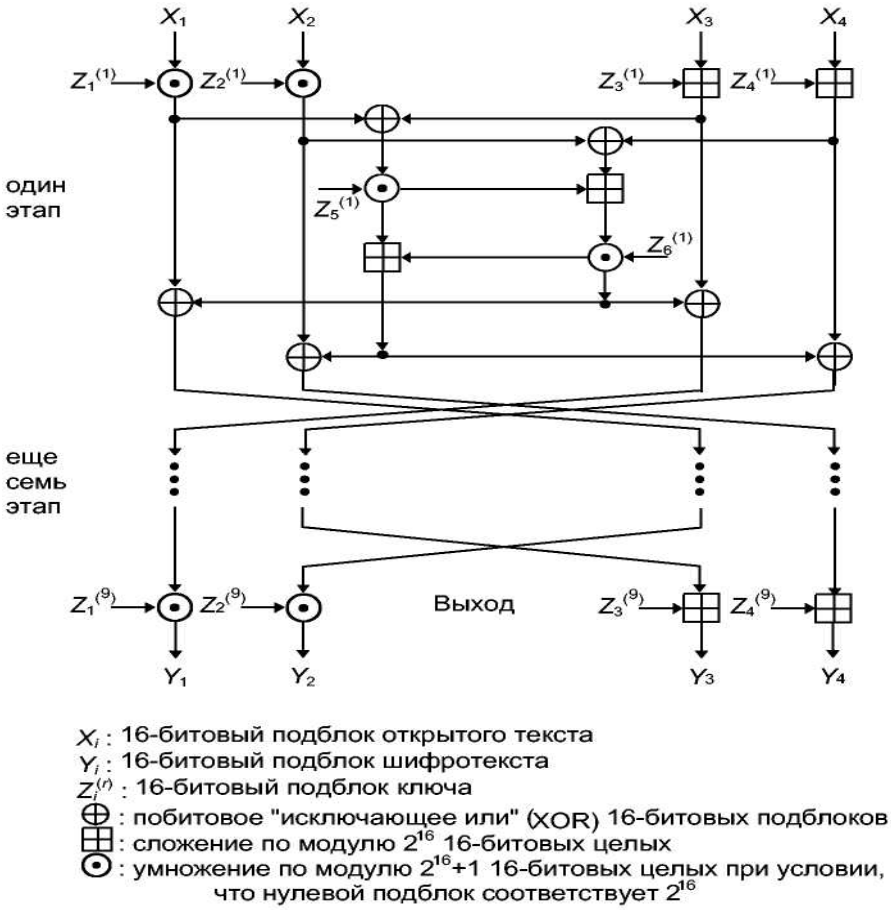


Рис. 24 PES.

Вилли Майер (Willi Meier) исследовал три алгебраических операции IDEA и показал, что, хотя они несовместимы, есть случаи, когда эти операции можно упростить так, чтобы в некоторой степени облегчить. Его вскрытие 2-этапного IDEA оказалось эффективнее вскрытия грубой силой (2^{42} операций), но для IDEA с 3 и более этапами эффективность этого

вскрытия была ниже вскрытия грубой силой. Безопасность полного 8-этапного IDEA осталась непоколебимой.

Джоан Дэймен (Joan Daemen) открыла класс слабых ключей IDEA. Эти ключи не являются слабыми в том смысле, в котором слабы некоторые ключи DES, для которых функция шифрования обратна самой себе. Слабость этих ключей состоит в том, что взломщик может легко определить их с помощью вскрытия с выбранным открытым текстом. Например, слабым является следующий ключ (в шестнадцатиричной записи):

0000, 0000, 0x00, 0000, 0000, 000x, xxxx, x000

В позиции *"x"* может стоять любая цифра. При использовании такого ключа побитовое XOR определенных пар открытых текстов равно побитовому XOR получившихся пар шифротекстов.

В любом случае вероятность случайной генерации одного из таких слабых ключей очень мала: $1/2^{96}$. Опасность случайно выбрать такой ключ практически не существует. К тому же, несложно модифицировать IDEA так, чтобы исключить наличие слабых ключей - достаточно выполнить XOR каждого подключа с числом 0x0dae.

Режимы работы и варианты IDEA

IDEA может работать в любом из режимов работы блочного шифра. Против двойных реализаций IDEA может быть предпринято то же вскрытие "встреча посередине", что и против DES.

Однако, так как ключ IDEA более чем в два раза длиннее ключа DES, это вскрытие непрактично. Объем нужной для такого вскрытия памяти составит $64*2^{128}$ битов, или 10^{39} байтов.

Можно увеличить вдвое размер блока. Алгоритм также прекрасно работал бы с 32-битовыми подблоками вместо 16-битовых и с 256-битовым ключом. Шифрование выполнялось бы быстрее, и безопасность возросла бы в 2^{32} раза. Но теория, на которой основан алгоритм, опирается на то, что $2^{16}+1$ является простым числом. А $2^{32} + 1$ простым числом не является. Может быть, алгоритм и можно изменить так, чтобы он работал, но его безопасность будет совсем иной. Некоторые эксперты считают, что заставить работать такой алгоритм будет нелегко.

2.4. АЛГОРИТМ AES

Изложено по статье [Панасенко]. Алгоритм шифрования AES (Advanced Encryption Standard – улучшенный стандарт шифрования) в настоящее время является стандартом блочного симметричного шифрования США.

Конкурс AES

В отличие от многих других криптографических стандартов (например, предыдущего стандарта шифрования США DES или отечественного

стандарта ГОСТ 28147-89), AES не был разработан «в недрах спецслужб», а выбран на открытом конкурсе из относительно большого числа алгоритмов-претендентов.

До принятия в качестве стандарта алгоритма AES в данном качестве использовался алгоритм DES (Data Encryption Standard – стандарт шифрования данных). Несмотря на множество найденных криптоаналитических методов, применимых к DES, фактически, не было обнаружено каких-либо серьезных уязвимостей в данном алгоритме, за исключением одного, но весьма принципиального недостатка: весьма короткого ключа, реальный размер которого составляет всего 56 бит. Насколько это мало, можно судить из следующей весьма распространенной оценки: для полного перебора ключей DES достаточно примерно 20 часов работы миллиона процессоров, каждый из которых перебирает миллион ключей DES в секунду. Ясно, что такой стойкости против вскрытия алгоритма методом «грубой силы» категорически недостаточно. Тем не менее, DES оставался стандартом шифрования до начала XXI века.

В 1997 году Институт Стандартов и Технологий США (NIST – National Institute of Standards and Technology) объявил о проведении конкурса по замене стандарта DES. На конкурс могли быть присланы алгоритмы шифрования, разработанные как организациями, так и частными лицами в любой стране. Алгоритм-победитель этого конкурса должен был стать новым стандартом блочного симметричного шифрования США.

Для участия в конкурсе (которому было присвоено название AES) алгоритм шифрования должен был соответствовать всего двум обязательным требованиям:

- 128-битный размер блока шифруемых данных,
- не менее трех поддерживаемых алгоритмом размеров ключей шифрования: 128, 192 и 256 бит.

Кроме того, NIST предъявил большое число требований, носивших рекомендательный характер. Именно по соответствию этим требованиям, фактически, и был выбран алгоритм-победитель конкурса. Рекомендации NIST были таковы:

- Алгоритм должен быть стойким против криптоаналитических атак, известных на время проведения конкурса.
- Структура алгоритма должна быть ясной, простой и обоснованной. Прежде всего, такая структура облегчила бы анализ алгоритма в рамках конкурса. Кроме того, ясность и простота структуры давала бы некоторую гарантию отсутствия в алгоритме специально внедренных авторами «закладок», т.е. некоторых недокументированных особенностей алгоритма, которые могли бы быть использованы авторами для его вскрытия.
- Должны отсутствовать слабые и эквивалентные ключи (т.е. ключи, являющиеся различными, но приводящие к одному и тому же результату шифрования).

- Скорость шифрования данных должна быть высокой на всех потенциальных аппаратных платформах – от 8-битных до 64-битных.
- Структура алгоритма должна позволять распараллеливание операций в многопроцессорных системах и аппаратных реализациях.
- Алгоритм должен предъявлять минимальные требования к оперативной и энергонезависимой памяти.
- Не должно быть ограничений для использования алгоритма; в частности, алгоритм не должен ограничивать свое использование в различных стандартных режимах работы, в качестве основы для построения хэш-функций, генераторов псевдослучайных последовательностей и т.д.

В конкурсе AES приняли участие 15 алгоритмов шифрования:

Алгоритм	*Автор*	*Достоинства и недостатки алгоритма*
CAST-256	Entrust Technologies, Inc.	В алгоритме не обнаружено уязвимостей. Однако, скорость шифрования у данного алгоритма невысока; кроме того, у него высокие требования к оперативной и энергонезависимой памяти.
Crypton	Future Systems, Inc.	Алгоритм без явных недостатков. Однако, Crypton уступает по всем характеристикам похожему на него алгоритму Rijndael. Эксперты конкурса сочли, что в финале конкурса Crypton однозначно проиграет Rijndael, поэтому не выбрали его во второй раунд конкурса.
DEAL	Richard Outerbridge, Lars Knudsen	Множество недостатков: наличие подмножеств слабых ключей, подверженность дифференциальному криптоанализу, отсутствие усиления при использовании 192-битных ключей по сравнению с 128-битными.
DFC	CNRS – Centre National pour la Recherche Scientifique – Ecole Normale Superieure	Достоинство алгоритма: очень высокая скорость шифрования на 64-битных платформах. При этом алгоритм весьма медленно шифрует на остальных платформах, а также имеет классы слабых ключей.
E2	NTT – Nippon Telegraph and Telephone Corporation	Аналогично алгоритму CAST-256, в E2 не найдено уязвимостей, но требования к энергонезависимой памяти слишком высоки.

FROG	TecApro Internacional S.A.	Алгоритм с наиболее оригинальной структурой и с наибольшим количеством недостатков: наличие уязвимостей, низкая скорость шифрования и высокие требования к ресурсам.
HPC	Richard Schroeppel	Аналогично алгоритму DFC, HPC очень быстро шифрует на 64-битных платформах, но весьма медленно на остальных. Кроме того, сложная и медленная процедура расширения ключа ограничивает возможные применения алгоритма, а наиболее сложная из всех представленных на конкурс алгоритмов структура раунда усложняет анализ алгоритма и делает недоказуемым отсутствие закладок.
LOKI97	Lawrie Brown, Josef Pieprzyk, Jennifer Seberry	Низкая скорость шифрования, высокие требования к ресурсам, наличие уязвимостей.
Magenta	Deutsche Telekom AG	Алгоритм подвержен атакам методами линейного и дифференциального криптоанализа; медленная скорость шифрования.
MARS	IBM	У алгоритма отсутствуют серьезные недостатки, за исключением низкой скорости шифрования на ряде платформ, не имеющих аппаратной поддержки требуемых операций и некоторых других незначительных недостатков. Алгоритм был незначительно модифицирован в течение первого раунда; измененный вариант вышел в финал конкурса.
RC6	RSA Laboratories	RC6 имеет весьма похожие на MARS проблемы с реализацией на некоторых платформах. Эксперты посчитали это незначительным недостатком – алгоритм вышел в финал конкурса.
Rijndael	Joan Daemen, Vincent Rijmen	Каких-либо недостатков у алгоритма не обнаружено; алгоритм вышел в финал конкурса.
SAFER+	Cylink Corporation	Алгоритм подвержен ряду атак и имеет низкую скорость выполнения.
Serpent	Ross Anderson, Eli Biham, Lars Knudsen	Выявлены некоторые незначительные недостатки, алгоритм вышел в финал конкурса.

Twofish	Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, Niels Ferguson	Из недостатков эксперты отметили сложность алгоритма, затрудняющую его анализ. Алгоритм вышел в финал конкурса.

Как видно из данной таблицы, список участников конкурса оказался весьма разнообразен, среди них можно отметить:

- всемирно известных криптологов, например, «звездный» коллектив авторов алгоритма Twofish;
- организации, давно работающие в данной области и обладающие как богатым опытом разработки криптоалгоритмов, так и сильнейшими специалистами в данной области, например, американская фирма RSA – автор алгоритма RC6;
- всемирно известные корпорации, обладающие большим научным потенциалом, например японский телекоммуникационный гигант NTT с алгоритмом E2;
- образовательные учреждения, известные своими достижениями в области криптографии, например Ecole Normale Superieure (Париж, Франция) с алгоритмом DFC;
- и наоборот, организации, весьма малоизвестные до проведения конкурса AES, например, фирма Tecnologia Apropriada (автор алгоритма FROG) из латиноамериканского государства Коста-Рика.

Анализ алгоритмов-претендентов проводился как специалистами института NIST, так и различными независимыми экспертами – на конкурс было прислано множество материалов, посвященных участвовавшим в конкурсе алгоритмам; все эти материалы можно найти на сайте института NIST http://csrc.nist.gov. По результатам анализа, как видно из приведенной выше таблицы, во второй раунд конкурса вышли следующие 5 алгоритмов: MARS, RC6, Rijndael, Serpent и Twofish.

Второй раунд конкурса был посвящен анализу уже этих пяти алгоритмов. В результате победителем конкурса был выбран алгоритм Rijndael, который по большинству критериев оказался лучше остальных четырех алгоритмов-финалистов.

Здесь подробно рассмотрим алгоритм шифрования Rijndael, которому (как победителю конкурса) было присвоено название AES.

Структура алгоритма

Алгоритм AES представляет блок данных в виде двумерного байтового массива размером 4 X 4. Все операции производятся над отдельными байтами массива, а также над независимыми столбцами и строками.

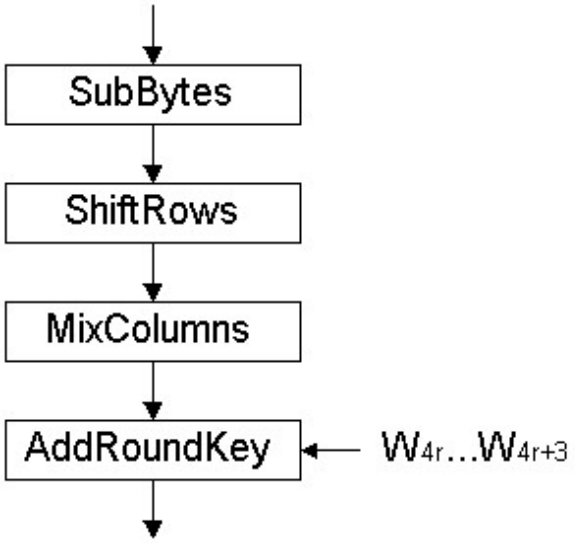


Рис.1. Раунд алгоритма.

В каждом раунде алгоритма выполняются следующие преобразования (см. рис. 1):

1. Операция SubBytes, представляющая собой табличную замену каждого байта массива данных согласно следующей таблице (см. рис. 2):

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

Таблица меняет входное значение 0 на 63 (шестнадцатеричное значение), 1 – на 7C, 2 – на 77 и т.д.

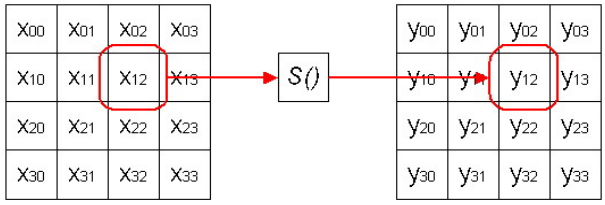


Рис. 2. Операция SubBytes.

Вместо данной табличной замены можно выполнить эквивалентную ей комбинацию двух операций:

1.1. Вычисление мультипликативной обратной величины от входного значения в конечном поле GF(2^8); обратной величиной от 0 является 0.

1.2. Выходное значение **b** вычисляется следующим образом:

$b_i = a_i \oplus a_{i+4 \bmod 8} \oplus a_{i+5 \bmod 8} \oplus a_{i+6 \bmod 8} \oplus a_{i+7 \bmod 8} \oplus c_i$,

где n_i обозначает i-й бит величины n,

a – результат предыдущей операции,

c – шестнадцатеричная константа 63.

2. Операция ShiftRows, которая выполняет циклический сдвиг влево всех строк массива данных, за исключением нулевой (см. рис. 3). Сдвиг i-й строки массива (для $i = 1, 2, 3$) производится на i байт.

3. Операция MixColumns. Выполняет умножение каждого столбца массива данных (см. рис. 4), который рассматривается как полином в конечном поле $GF(2^8)$, на фиксированный полином $a(x)$: $a(x) = 3x^3 + x^2 + x + 2$.

Умножение выполняется по модулю $x^4 + 1$.

4. Операция AddRoundKey выполняет наложение на массив данных материала ключа. А именно, на i-й столбец массива данных ($i = 0…3$) побитовой логической операцией «исключающее или» (XOR) накладывается определенное слово расширенного ключа W_{4r+i}, где r – номер текущего раунда алгоритма, начиная с 1 (процедура расширения ключа будет описана ниже). Операция AddRoundKey представлена на рис. 5.

Количество раундов алгоритма R зависит от размера ключа следующим образом:

Размер ключа, бит	R
128	10
192	12
256	14

Перед первым раундом алгоритма выполняется предварительное наложение материала ключа с помощью операции AddRoundKey, которая выполняет наложение на открытый текст первых четырех слов расширенного ключа $W_0 \ldots W_3$.

Последний же раунд отличается от предыдущих тем, что в нем не выполняется операция MixColumns.

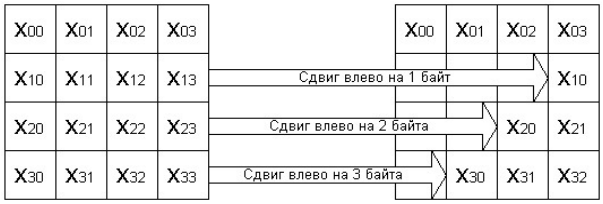


Рис. 3. Операция ShiftRows.

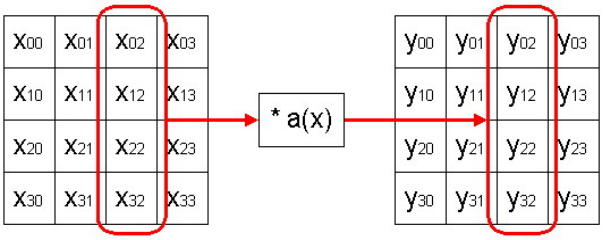


Рис. 4. Операция MixColumns.

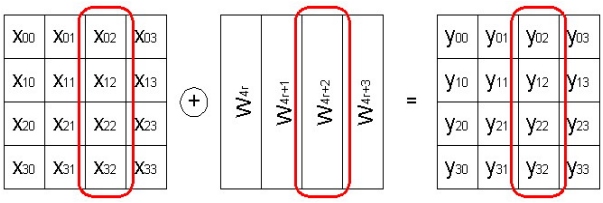


Рис. 5. Операция AddRoundKey.

Расширение ключа

AES использует ключи шифрования трех фиксированных размеров: 128, 192, и 256 бит. В зависимости от размера ключа, конкретный вариант алгоритма AES может обозначаться как AES-128, AES-192 и AES-256 соответственно.

Задача процедуры расширения ключа состоит в формировании нужного количество слов расширенного ключа для их использования в операции AddRoundKey. Как было сказано выше, под «словом» здесь понимается 4-байтный фрагмент расширенного ключа, один из которых используется в первичном наложении материала ключа и по одному – в

каждом раунде алгоритма. Таким образом, в процессе расширения ключа формируется 4*(*R*+1) слов.

Расширение ключа выполняется в два этапа, на первом из которых производится инициализация слов расширенного ключа (обозначаемых как W_i): первые *Nk* (*Nk* – размер исходного ключа шифрования *K* в словах, т.е. 4, 6 или 8) слов W_i (т.е. $i = 0 \ldots Nk\text{-}1$) формируются их последовательным заполнением байтами ключа (см. рис. 6).

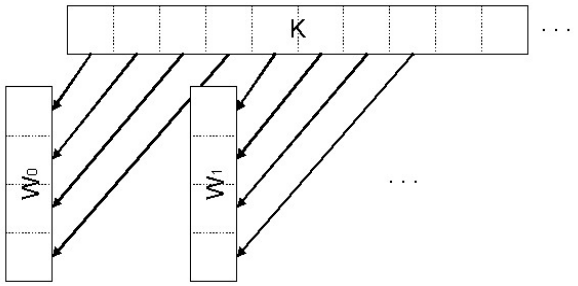


Рис. 6. Инициализация первых слов расширенного ключа.

Последующие слова W_i формируются следующей последовательностью операций для каждого $i = Nk \ldots 4*(R+1)-1$:

Шаг 1. Инициализируется временная переменная *T*: $T = W_{i-1}$.

Шаг 2. Данная переменная модифицируется следующим образом:

- если *i* кратно *Nk*, то: T = SubWord(RotWord(T)) $\oplus$ $RC_{i/Nk}$; операции SubWord, RotWord будут описаны ниже, а константы RC_n представляют собой слова, в которых все байты, кроме первого являются нулевыми, а первый байт имеет значение $2^{n-1} \bmod 256$;
- если $Nk = 8$ и $(i \bmod Nk) = 4$, то: T = SubWord(T);
- в остальных случаях модификация переменной *T* не выполняется.

Шаг 3. Формируется i-е слово расширенного ключа: $W_i = W_{i-Nk} \oplus T$.

Операция SubWord выполняет над каждым байтом входного значения табличную замену, которая была описана выше – см. операцию SubBytes.

Операция RotWord побайтно вращает входное слово на 1 байт влево.

Как видно, процедура расширения ключа является достаточно простой по сравнению со многими другими современными алгоритмами шифрования. Процедура расширения ключа имеет также несомненное достоинство в том, что расширение ключа может быть выполнено «на лету» (on-the-fly), т.е. параллельно с зашифрованием данных.

Авторы алгоритма указывают также, что не следует задавать напрямую

расширенный ключ – программная или аппаратная реализация алгоритма должна именно получать исходный ключ шифрования K и выполнять процедуру расширения ключа. Здесь стоит снова вспомнить алгоритм DES – известно, что DES с независимо задаваемыми ключами раундов оказался слабее против некоторых атак, чем исходный алгоритм DES.

Расшифрование

Расшифрование выполняется применением обратных операций в обратной последовательности. Соответственно, перед первым раундом расшифрования выполняется операция AddRoundKey (которая является обратной самой себе), выполняющая наложение на шифртекст четырех последних слов расширенного ключа, т.е. $W_{4R}...W_{4R+3}$.

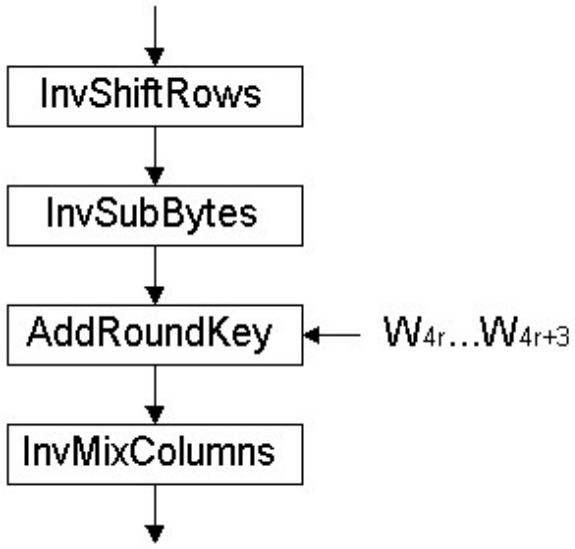


Рис. 7. Раунд расшифрования.

Затем выполняется R раундов расшифрования, каждый из которых выполняет следующие преобразования (см. рис. 7):

1. Операция InvShiftRows выполняет циклический сдвиг вправо трех последних строк массива данных на то же количество байт, на которое выполнялся сдвиг операцией ShiftRows при зашифровании.
2. Операция InvSubBytes выполняет побайтно обратную табличную замену, которая определена следующей таблицей:

52	09	6A	D5	30	36	A5	38	BF	40	A3	9E	81	F3	D7	FB
7C	E3	39	82	9B	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
54	7B	94	32	A6	C2	23	3D	EE	4C	95	0B	42	FA	C3	4E
08	2E	A1	66	28	D9	24	B2	76	5B	A2	49	6D	8B	D1	25
72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	B6	92
6C	70	48	50	FD	ED	B9	DA	5E	15	46	57	A7	8D	9D	84
90	D8	AB	00	8C	BC	D3	0A	F7	E4	58	05	B8	B3	45	06

D0	2C	1E	8F	CA	3F	0F	02	C1	AF	BD	03	01	13	8A	6B
3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
60	51	7F	A9	19	B5	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
A0	E0	3B	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
17	2B	04	7E	BA	77	D6	26	E1	69	14	63	55	21	0C	7D

Данную табличную замену можно выполнить, применив к входному байту преобразование, обратное операции 1.2 (см. описание операции SubBytes), после чего вычислить мультипликативную обратную величину от результата предыдущей операции в конечном поле $GF(2^8)$.

3. Операция AddRoundKey, как и при зашифровании, выполняет наложение на обрабатываемые данные четырех слов расширенного ключа $W_{4r}\ldots W_{4r+3}$. Однако, нумерация раундов r при расшифровании производится в обратную сторону – от R-1 до 0.
4. Операция InvMixColumns выполняет умножение каждого столбца массива данных аналогично прямой операции MixColumns, однако, умножение производится на полином $a^{-1}(x)$, определенный следующим образом: $a^{-1}(x) = Bx^3 + Dx^2 + 9x + E$.

Аналогично зашифрованию, последний раунд расшифрования не содержит операцию InvMixColumns.

Отличия AES от исходного алгоритма Rijndael

Алгоритм Rindael позволяет шифровать данные не только 128-битными блоками, но и блоками по 192 или 256 бит. Таким образом, AES, фактически, имеет лишь одно принципиальное отличие от Rijndael: он предусматривает использование только 128-битных блоков данных. Рассмотрим изменения в приведенном выше описании алгоритма AES, связанные с другими размерами блоков:

1. Обрабатываемые данные могут представляться не только в виде массива размером 4 X 4, но и 4 X 6 или 4 X 8 для 192- и 256-битных блоков соответственно.
2. Количество раундов R алгоритма Rijndael определяется следующей таблицей в зависимости не только от размера ключа, но и от размера блока:

Размер ключа, бит	Размер блока, бит		
	128	192	256
128	10	12	14
192	12	12	14

256	14	14	14

3. Количество бит сдвига строк таблицы также зависит от размера блока:

Номер строки	Размер блока, бит		
	128	192	256
1	1	1	1
2	2	2	3
3	3	3	4

4. Поскольку для 192- и 256-битного блоков увеличивается количество столбцов массива данных до 6 и 8 соответственно, в операции AddRoundKey участвуют уже 6 или 8 слов расширенного ключа вместо четырех. Следовательно, в r-м раунде алгоритма выполняется наложение слов расширенного ключа $W_{Nb*r}…W_{Nb*r+3}$, где *Nb* – количество столбцов массива данных.
5. В связи с вышесказанным, изменяется и процедура расширения ключа, однако, изменение состоит лишь в том, что данная процедура должна выработать *Nb**(*R*+1)-1 слов расширенного ключа, а не 4*(*R*+1)-1 (что, впрочем, остается справедливым для 128-битного блока).

Первичная оценка криптостойкости

Первичная оценка криптостойкости алгоритма Rijndael была приведена авторами алгоритма в его спецификации, представленной на конкурс AES. Согласно оценкам авторов, Rijndael не подвержен следующим видам криптоаналитических атак:

1. У алгоритма отсутствуют слабые ключи, а также возможности его вскрытия с помощью атак на связанных ключах (описание криптоаналитических атак, упомянутых в данной статье, можно найти в статьях «Современные методы вскрытия алгоритмов шифрования», «Атаки на шифраторы с использованием утечек данных по побочным каналам» и «Криптоаналитические атаки на связанных ключах»).
2. К алгоритму не применим дифференциальный криптоанализ.
3. Алгоритм не атакуем с помощью линейного криптоанализа и усеченных дифференциалов.
4. Square-атака (специфичная атака на алгоритмы со структурой «квадрат», к которым относится и AES) также не применима к алгоритму Rijndael.
5. Алгоритм не вскрывается методом интерполяции.

Исследования в рамках конкурса AES

Сначала рассмотрим те результаты криптоанализа, которые были учтены экспертами при выборе алгоритма-победителя конкурса AES. Прежде всего, было найдено несколько атак на усеченные (с уменьшенным количеством раундов) версии алгоритма Rijndael:

- В упомянутой выше первичной оценке криптостойкости были приведены некоторые атаки на усеченные версии Rijndael, из

которых стоит отметить Square-атаку на 6-раундовую версию алгоритма, для которой необходимо 2^{32} выбранных открытых текстов и 2^{72} операций шифрования.

- Square-атака на 6-раундовую версию Rijndael была незначительно усилена Эли Бихамом и Натаном Келлером, которые также предложили атаку методом невозможных дифференциалов на 5-раундовую версию алгоритма; для нее требуется $2^{29,5}$ выбранных открытых текстов и 2^{31} операций. Атака с помощью невозможных дифференциалов рядом корейских специалистов была распространена на 6-раундовую версию Rijndael: для данной атаки требуется $2^{91,5}$ выбранных открытых текстов и 2^{122} операций шифрования.
- Впоследствии Square-атака была расширена на 7 раундов алгоритма Rijndael. Однако, данная атака весьма непрактична: для ее реализации требуется 2^{32} выбранных открытых текстов и от 2^{184} до 2^{208} операций шифрования.
- Авторы алгоритма Twofish (финалиста конкурса AES) с участием ряда других специалистов продолжили усиление Square-атаки против алгоритма Rijndael: новая Square-атака позволяла вскрыть 6-раундовый Rindael выполнением 2^{44} операций, а 7-раундовый – от 2^{155} до 2^{172} операций шифрования при том же требуемом количестве выбранных открытых текстов. Кроме того, новая атака позволяет вскрыть 7- и 8-раундовые (последнюю – только при использовании 256-битного ключа) версии Rijndael при наличии от 2^{119} до 2^{128} выбранных открытых текстов выполнением, соответственно, 2^{120} или 2^{204} операций.
- Та же команда криптологов предложила атаку на связанных ключах на 9-раундовую версию Rijndael с 256-ключом, которой необходимо 2^{77} выбранных открытых текстов, зашифрованных на 256 связанных ключах, и 2^{224} операций шифрования.

С учетом того, что в алгоритме Rijndael выполняется, как минимум, 10 раундов, запас криптостойкости алгоритма экспертами был признан адекватным. С этим, однако, согласны далеко не все специалисты. Утверждается, что алгоритм, выбранный в качестве стандарта AES, должен оставаться криптографически стойким до 2100 года. Ясно, что за почти сто лет будет сделано огромное количество попыток вскрытия AES, некоторые из которых приведут к увеличению количества вскрываемых раундов. Судя по последнему десятилетию, появятся и принципиально новые виды криптоаналитических атак. Поэтому, считая формально, Rijndael должен выполнять, как минимум, 18 раундов.

В рамках исследований в течение первого раунда конкурса AES было также отмечено, что Rijndael имеет ряд потенциальных уязвимостей при реализации данного алгоритма в смарт-картах:

- Rijndael может быть подвержен атаке по потребляемой мощности, нацеленной на его процедуру расширения ключа, что, однако, не доказано Эли Бихамом и Эди Шамиром.
- Весьма интересное исследование на данную тему было проведено специалистами исследовательского центра компании IBM. Они выполнили реализацию алгоритма Twofish для типичной смарт-карты с CMOS-архитектурой и проанализировали возможность атаки на данный алгоритм с помощью дифференциального анализа потребляемой мощности (DPA – Differential Power Analysis). Оказалось, что атаке подвержена процедура входного отбеливания алгоритма Twofish, в результате чего можно вычислить ключ шифрования данного алгоритма. Атака была (теоретически) распространена на остальные алгоритмы-участники конкурса AES. Аналогично алгоритму Twofish, с помощью DPA атакуется предварительное наложение материала ключа, выполняемое в алгоритме Rijndael, после чего вычисляется ключ шифрования целиком.

Данные потенциальные уязвимости не повлияли на выбор алгоритма Rijndael в качестве стандарта AES, поскольку, во-первых, остальные алгоритмы-финалисты не принципиально лучше в данном контексте, а во-вторых, существуют различные методы противодействия атакам по потребляемой мощности, которые, однако усложняют реализацию и ухудшают быстродействие алгоритма.

Исследования после конкурса AES

Как и предполагали эксперты, после принятия алгоритма Rijndael в качестве стандарта AES попытки вскрытия данного алгоритма существенно усилились.

Можно сказать, что криптоанализ алгоритма AES стал развиваться, в основном, в следующих четырех направлениях:

Во-первых, попытки усиления «классических» атак или применения других известных атак к данному алгоритму. Например, атаку методом бумеранга на 6-раундовую версию алгоритма со 128-битным ключом, для выполнения которой требуется 2^{39} выбранных открытых текстов, 2^{71} шифртекстов с адаптивным выбором и 2^{71} операций шифрования.

Во-вторых, применение различных методов криптоанализа на связанных ключах, в частности:

- Предложено несколько вариантов атак на связанных ключах на 7- и 8-раундовый AES-192 с использованием невозможных дифференциалов.
- Комбинация метода бумеранга и связанных ключей предложена в работе [3]: 9-раундовый AES-192 атакуется при наличии 2^{79} выбранных открытых текстов, каждый из которых шифруется на 256 связанных ключах, выполнением 2^{125} операций шифрования;

для атаки на 10-раундовый AES-256 требуется $2^{114,9}$ выбранных открытых текстов (включая зашифрование на 256 связанных ключах) и $2^{171,8}$ операций. Данная атака использует слабость процедуры расширения ключа, состоящую в ее недостаточной нелинейности.

- Эта атака была усилена в работе [17], в которой, в частности, предлагается атака на 10-раундовый алгоритм AES-192, для которой требуется 2^{125} выбранных открытых текстов (на 256 связанных ключах) и $2^{146,7}$ операций.

Несмотря на то, что предложенные атаки на связанных ключах являются весьма непрактичными, настораживает тот факт, что атаке подвержены уже 10 из 12 раундов алгоритма AES-192 (и это после всего 5 лет после принятия стандарта AES!) – возникает опасение, что эксперты (указывающие на недостаточность раундов в алгоритме Rijndael) были правы и полнораундовый алгоритм AES будет вскрыт существенно раньше, чем предполагали эксперты института NIST.

В-третьих, многие исследования посвящены алгебраической структуре алгоритма Rijndael, например:

- Найдены линейные соотношения в таблице замен Rijndael (т.е. в единственном нелинейном элементе алгоритма). Однако, как и в других аналогичных работах, каких-либо практических возможностей использования данного свойства не предложено.
- Зашифрование с помощью Rijndael можно выразить относительно (особенно по сравнению с другими «серьезными» алгоритмами шифрования) простой формулой. Авторы не нашли практического применения данной формулы, но предположили, что она будет использована в реальных атаках в течение ближайших примерно 20 лет.
- Показано, что вскрытие алгоритма AES эквивалентно решению системы квадратичных уравнений в конечном поле $GF(2^8)$.

Попытки использования алгебраических свойств алгоритма для его вскрытия были названы «алгебраическими атаками». Стоит отметить, что были и работы с попытками доказательства того факта, что простая структура алгоритма AES не ухудшает его криптостойкости.

В-четвертых, больше всего исследований посвящено атакам, использующим информацию, полученную по побочным каналам:

- Много работ содержат примеры успешного вскрытия различных реализаций полнораундового алгоритма AES с помощью атак по времени исполнения, потребляемой мощности и атак на основе сбоев.
- Не меньше исследований посвящено безопасным (т.е. защищенным от утечек данных по побочным каналам) программным или аппаратным реализациям AES.

Заключение

Итак, на май 2007 г., по прошествии всего 6 лет после принятия алгоритма Rijndael в качестве стандарта AES, криптоаналитики весьма серьезно продвинулись во вскрытии данного алгоритма:

- Предложена теоретическая атака уже на 10 раундов (из 12) алгоритма AES-192.
- Существует множество примеров вскрытия реализаций алгоритма AES с помощью side-channel-атак.

2.5. ОБЗОР НЕКОТОРЫХ СОВРЕМЕННЫХ БЛОЧНЫХ ШИФРОВ

Со временем DES все-таки устаревал. Кроме того, возникла необходимость в специализированных алгоритмах. Как следствие, появились новые блочные шифры. Основой для некоторых таких алгоритмов послужил хорошо известный Люцифер.

Совершенствование средств нападения (линейного и дифференциального методов криптоанализа) поставило перед разработчиками новые задачи.

2.5.1. Алгоритм LUCIFER

В конце 60-х IBM начала выполнение исследовательской программы по компьютерной криптографии, названной Люцифером (Lucifer) и руководимой сначала Хорстом Фейстелем (Horst Feistel), а затем Уолтом Тачменом (Walt Tuchman). Это же название - Lucifer - получил блочный алгоритм, появившийся в результате этой программыв начале 70-х. В действительности существует по меньшей мере два различных алгоритма с таким именем содержит ряд пробелов в спецификации алгоритма. Все это привело к заметной путанице.

Lucifer - это набор перестановок и подстановок, его блоки похожи на блоки DES. В DES результат функции f объединяется с помощью XOR со входом предыдущего этапа, образуя вход следующего этапа. У S-блоков алгоритма Lucifer 4-битовые входы и 4-битовые выходы, вход S-блоков представляет собой перетасованный выход S-блоков предыдущего этапа, входом S-блоков первого этапа является открытый текст. Для выбора используемого S-блока из двух возможных применяется бит ключа. (Lucifer реализует это, как один T-блок с 9 битами на входе и 8 битами на выходе.) В отличие от DES половины блока между этапами не переставляются и вообще понятие половины блока не используется в алгоритме Lucifer. У этого алгоритма 16 этапов, 128-битовые блоки и более простое, чем в DES, распределение ключей.

Применив дифференциальный криптоанализ к первой реализации Lucifer'a, Бихам и Шамир показали, что Lucifer с 32-битовыми блоками и 8 этапами может быть взломан с помощью 40 выбранных открытых текстов за 2^{39} шагов, тот же способ позволит вскрыть Lucifer с 128-битовыми блоками и 8 этапами с помощью 60 выбранных открытых текстов за 2^{53} шагов. 18-этапный, 128-битовый Lucifer вскрывается дифференциальным криптоанализом с помощью 24 выбранных открытых текстов за 2^{21} шагов. Все эти вскрытия использовали сильные S-блоки DES. Применив дифференциальный криптоанализ против второй реализации Lucifer, Бихам и Шамир обнаружили, что S-блоки намного слабее, чем в DES. Дальнейший анализ показал, что более половины возможных ключей не являются безопасными. Криптоанализ со связанными ключами может взломать 128-

битовый Lucifer с любым числом этапов с помощью 2^{33} выбранных открытых текстов для выбранных ключей или 2^{65} известных открытых текстов для выбранных ключей. Вторая реализация Lucifer еще слабее.

Lucifer является объектом нескольких патентов США. Сроки действия всех этих патентов истекли.

2.5.2. Алгоритм MADRYGA

В.Е. Мадрига (W. E. Madryga) предложил этот блочный алгоритм в 1984 году. Он может быть эффективно реализован как программа: в нем нет надоедливых перестановок, и все операции выполняются над байтами. Стоит перечислить задачи, которые решал автор при проектировании алгоритма:

1. Открытый текст нельзя получить из шифротекста без помощи ключа. (Это означает только то, что алгоритм безопасен.)

Количество операций, нужное для определения ключа по имеющимся шифротексту и открытому тексту, должно быть статистически равно произведению количества операций при шифровании на число возможных ключей. (Это означает, что никакое вскрытие с открытым текстом не может быть лучше, чем вскрытие грубой силой.)

Известность алгоритма не влияет на силу шифра. (Безопасность полностью определяется ключом.)

Изменение одного бита ключа должно вызывать для того же открытого текста радикальное изменение шифротекста, и изменение одного бита открытого текста должно вызывать для того же ключа радикальное изменение шифротекста. (Это лавинный эффект.)

Алгоритм должен содержать некоммутативную комбинацию подстановок и перестановок.

Подстановки и перестановки, используемые в алгоритме, должны определяться и входными данными, и ключом.

Избыточные группы битов открытого текста должны быть полностью замаскированы в шифротексте.

Длина шифротекста должна равняться длине открытого текста.

9. Не должно быть простых взаимосвязей между любыми возможными ключами и особенностями шифротекста.

10. Все возможные ключи должны давать сильный шифр. (Не должно быть слабых ключей.)

11. Длина ключа и текста могут регулироваться для реализации различных требований к безопасности.

12. Алгоритм должен позволять эффективную программную реализацию на больших мэйнфреймах, миникомпьютерах, микрокомпьютерах и с помощью дискретной логики. (По сути используемые в алгоритме функции ограничены XOR и битовым сдвигом.)

DES удовлетворял первым девяти требованиям, но последние три были новыми. В предположении, что лучшим способом вскрытия алгоритма является грубая сила, переменная длина ключа, конечно же, заставит

замолчать тех, кто считает, что 56 битов - это слишком мало. Такие люди могут реализовать этот алгоритм с л гобой нужной им длиной ключа. А любой, кто когда-нибудь пытался реализовать DES программно, обрадуется алгоритму, который учитывает возможности программных реализаций.

Описание Madryga

Madryga состоит из двух вложенных циклов. Внешний цикл повторяется восемь раз (но это количество может быть увеличено для повышения) и содержит применение внутреннего цикла к открытому тексту. Внутренний цикл превращает открытый текст в шифротекст, повторяясь для каждого 8-битового блока (байта) открытого текста. Следовательно, весь открытый текст восемь раз последовательно обрабатывается алгоритмом.

Итерация внутреннего цикла оперирует с 3-байтовым окном данных, называемым рабочим кадром. Это окно смещается на 1 байт за итерацию. (При работе с последними 2 байтами данные считаются циклически замкнутыми.) Первые два байта рабочего кадра циклически сдвигаются на переменное число позиций, а для последнего байта выполняется XOR с некоторыми битами ключа. По мере продвижения рабочего кадра все байты последовательно "вращаются" и подвергаются операции XOR с частями ключа. Последовательные вращения перемешивают результаты предыдущих операций XOR и вращения, а результат XOR влияет на вращение. Это делает весь процесс обратимым.

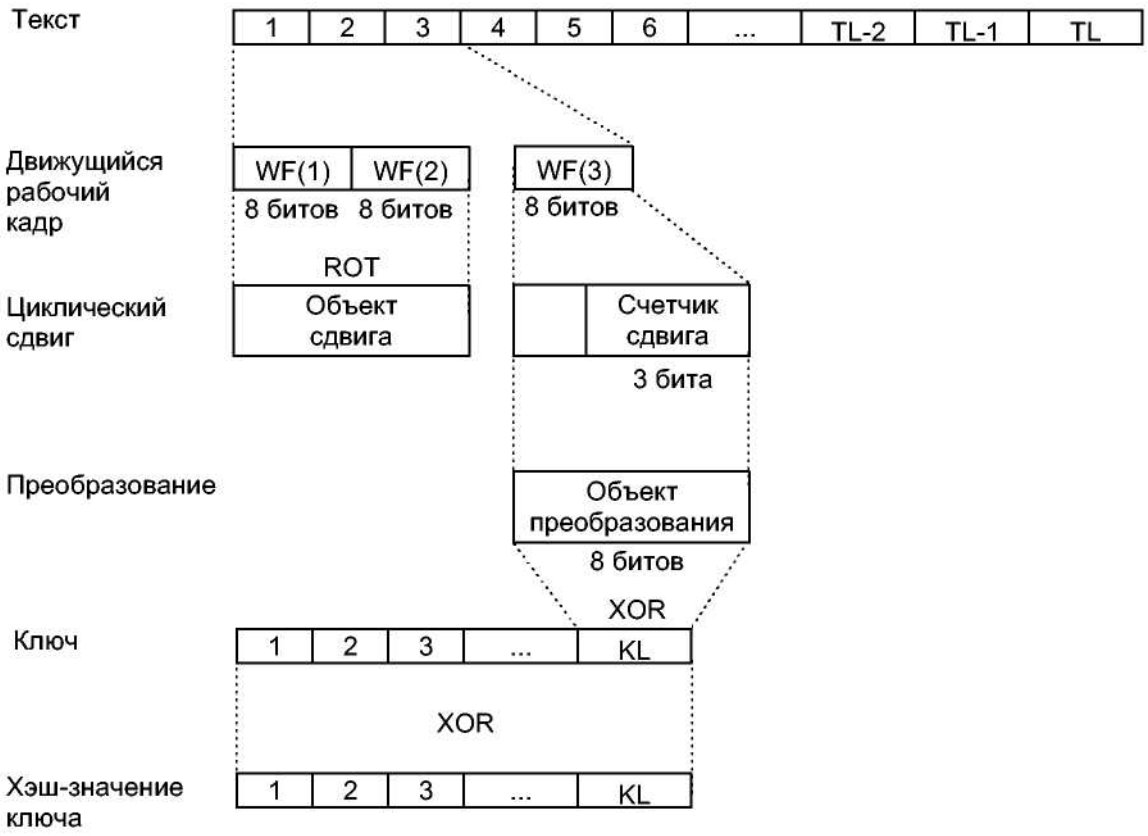


Рис. 22 Одна итерация Madryga.

Так как каждый байт данных влияет на два байта слева от себя и на один байт справа, после восьми проходов каждый байт шифротекста зависит от 16 байтов слева и от восьми байтов справа.

При шифровании каждая итерация внутреннего цикла устанавливает рабочий кадр на предпоследний байт открытого текста и циклически перемещает его к байту открытого текста, третьему слева от последнего. Сначала весь ключ подвергается операции XOR со случайной константой и затем циклически смещается влево на 3 бита. Младшие три бита младшего байта рабочего кадра сохраняются, они определяют вращение остальных двух байтов. Затем для младшего байта рабочего кадра выполняется операция XOR с младшим байтом ключа. Далее объединение двух старших байтов циклически смещается влево на переменное число битов (от 0 до 7). Наконец рабочий кадр смещается вправо на один байт и весь процесс повторяется.

Смысл случайной константы в том, чтобы превратить ключ в псевдослучайную последовательность. Длина константы должна быть равна длине ключа. При обмене данными абоненты должны пользоваться константой одинаковой длины. Для 64-битового ключа Мадрига рекомендует константу 0x0fle2d3c4b5a6978.

При дешифрировании процесс инвертируется. При каждой итерации внутреннего цикла рабочий кадр устанавливается на байт, третий слева от последнего байта шифротекста, и циклически перемещается в обратном направлении до байта, который находится на 2 байта левее последнего байта шифротекста. И ключ, и 2 байта шифротекста в процессе циклически смещаются направо, а XOR выполняется перед циклическими сдвигами.

Криптоанализ Madryga

Исследователи из Технического университета в Квинсланде (Queensland University of Technology) исследовали Madryga вместе с некоторыми другими блочными шифрами. Они обнаружили, что в этом алгоритме не проявляется лавинный эффект для преобразования открытого текста в шифротекст. Кроме того, во многих шифротекстах процент единиц был выше, чем процент нулей.

Хотя нет сведений о проведении формального анализа этого алгоритма, он не производит впечатление супернадежного. При поверхностном знакомстве с ним Эли Бихам пришел к следующим выводам:

Алгоритм состоит только из линейных операций (циклическое смещение и XOR), незначительно изменяемых в зависимости от данных.

В этом нет ничего похожего на мощь S-блоков DES.

Четность всех битов шифротекста и открытого текста неизменна и зависит только от ключа. Поэтому, обладая открытым текстом и соответствующим шифротекстом, можно предсказать четность шифротекста для любого открытого текста.

2.5.3. Алгоритмы KHUFU и KHAFRE

В 1990 году Ральф Меркл (Ralph Merkle) предложил два алгоритма. В основе их проектирования лежали следующие принципы:

1. 56-битовый размер ключа DES слишком мал. Так как стоимость увеличения размера ключа пренебрежимо мала (компьютерная память недорога и доступна), он должен быть увеличен.
2. Интенсивное использование перестановок в DES хотя и удобно для аппаратных реализаций, чрезвычайно затрудняет программные реализации. Наиболее быстрые реализации DES выполняют перестановки табличным образом. Просмотр таблицы может обеспечить те же характеристики "рассеяния", что и собственно перестановки, и может сделать реализацию намного более гибкой.
3. S-блоки DES, всего с 64 4-битовыми элементами, слишком малы. Теперь с увеличением памяти должны увеличиться и S-блоки. Более того, все восемь S-блоков используются одновременно. Хотя это и удобно для аппаратуры, для программной реализации это кажется ненужным ограничением. Должны быть реализованы больший размер S-блоков и последовательное (а не параллельное) их использование.
4. Широко признано, что начальная и заключительная перестановки криптографически бессмысленны, поэтому они должны быть устранены.
5. Все быстрые реализации DES заранее рассчитывают ключи для каждого этапа. При данном условии нет смысла усложнять эти вычисления.
6. В отличие от DES критерии проектирования S-блоков должны быть общедоступны.

К этому перечню Меркл, возможно, теперь добавил бы "устойчивость к дифференциальному и линейному криптоанализу", ведь в то время эти способы вскрытия не были известны.

Khufu

Khufu - это 64-битовый блочный шифр. 64-битовый открытый тест сначала разбивается на две 32-битовые половины, *L* и *R*. Над обеими половинами и определенными частями ключа выполняется операция XOR. Затем, аналогично DES, результаты проходят через некоторую последовательность этапов. На каждом этапе младший значащий байт *L* используется в качестве входных данных S-блока. У каждого S-блока 8 входных битов и 32 выходных бита. Далее выбранный в S-блоке 32-битовый элемент подвергается операции XOR с *R*. Затем *L* циклически сдвигается не несколько из восьми битов, *L* и *R* меняются местами, и этап заканчивается. Сам S-блок не является статическим, но меняется каждые восемь этапов. Наконец после последнего этапа над *L* и *R* выполняется операция XOR с другими частями ключа, и половины объединяются, образуя блок шифротекста.

Хотя части ключа используются для XOR с блоком шифрования в начале и в конце алгоритма, главная цель ключа - генерация S-блоков. Эти S-

блоки - секретны, по сути являются они являются частью ключа. Полный размер ключа Khufu равен 512 битам (64 байтам), алгоритм предоставляет способ генерации S-блоков по ключу. Количество этапов алгоритма остается открытым. Меркл упомянул, что 8-этапный Khufu чувствителен к вскрытию с выбранным открытым текстом и рекомендует 16, 24 или 32 этапа. (Он ограничивает выбор количества этапов числами, кратными восьми.)

Так как в Khufu используются зависимые от ключа и секретные S-блоки, он устойчив к дифференциальному криптоанализу. Существует дифференциальное вскрытие 16-этапного Khufu, которое раскрывает ключ после 2^{31} выбранных открытых текстов, но его не удалось расширить на большее количество этапов. Если лучшим способом вскрыть Khufu является грубая сила, то его надежность производит сильное впечатление. 512-битовый ключ обеспечивает сложность 2^{512} - огромное число при любых условиях.

Khafre

Khafre - это вторая из криптосистем, предложенных Мерклом. (Khufu (Хуфу) и Khafre (Хафр) - это имена египетских фараонов.) По конструкции этот алгоритм похож на Khufu, но он спроектирован для приложений, не использующих предварительных вычислений. S-блоки не зависят от ключа. Вместо этого Khafre использует фиксированные S-блоки. Блок шифрования подвергается операции XOR с ключом не только перед первым этапом и после последнего, но и после каждых 8 этапов шифрования.

Меркл предположил, что с Khafre должны использоваться 64- или 128-битовые ключи, и что для Khafre потребуется больше этапов, чем для Khufu. Это наряду с тем, что каждый этап Khafre сложнее этапа Khufu, делает Khafre более медленным. Зато для Khafre не нужны никакие предварительны расчеты, что позволяет быстрее шифровать небольшие порции данных.

В 1990 году Бихам и Шамир применили свой метод дифференциального анализа против Khafre. Им удалось взломать 16-этапный Khafre с помощью вскрытия с выбранным открытым текстом после 1500 различных шифрований. На их персональном компьютере это заняло около часа. Преобразование этого вскрытия во вскрытие с известным открытым текстом потребует около 238 шифрований. Khafre с 24 этапами может быть вскрыт с помощью вскрытия с выбранным открытым текстом за 253 шифрования, а с помощью вскрытия с известным открытым текстом - за 259 шифрования.

2.5.4. Алгоритм RC2

RC2 представляет собой алгоритм с переменной длиной ключа, спроектированный Роном Ривестом (Ron Rivest) для RSA Data Security, Inc. (RSADSI). Очевидно "RC" - это сокращенное "Ron's Code" ("Код Рона"), хотя официально это "Rivest Cipher" ("Шифр Ривеста"). (RC3 был взломан в RSADSI в процессе разработки, RC1 не вышел за пределы записной книжки

Ривеста.) Он представляет собой частную собственность, и его детали не были опубликованы. RC2 уже появился в коммерческих продуктах. RC2 не был запатентован и защищен только как торговый секрет.

RC2 - это шифр с 64-битовым блоком и переменной длиной ключа, предназначенный заменить DES. В соответствии с утверждениями компании программные реализации RC2 в три раза быстрее DES. Алгоритм может использовать ключ переменной длины, от 0 байтов до максимальной длины строки, поддерживаемой компьютерной системой, скорость шифрования не зависит от размера ключа. Этот ключ предварительно используется для заполнения 128-байтовой таблицы, зависящей от ключа. Поэтому множество действительно различных ключей составляет 21024. RC2 не использует S-блоков, используются две операции - "смешивание" и "перемешивание" ("mix" и "mash"), для каждого этапа выбирается одна из них. В соответствии с литературой:

. . . RC2 не является итеративным блочным шифром. Это предполагает, что RC2 более устойчив к дифференциальному и линейному криптоанализу, чем другие блочные шифры, безопасность которых опирается на копирование схемы DES.

Отказ RSADSI опубликовать RC2 заставляет сомневаться в намерениях этой компании. Она обещает предоставить детали алгоритма всем, кто подпишет соглашение о нераспространении информации, и утверждает, что позволит криптоаналитикам опубликовать любые обнаруженные негативные результаты.

RC4, также являющийся интеллектуальной собственностью RSADSI, был опубликован в Internet, и, вероятно, опубликование RC2 является только вопросом времени.

По соглашению между Ассоциацией издателей программного обеспечения (Software Publishers Association, SPA) и правительством США RC2 и RC4 получили специальный экспортный статус. Процесс получения разрешения на экспорт продуктов, реализующих один из этих двух алгоритмов, значительно упрощен при условии, что длина ключа не превышает 40 битов.

Достаточен ли 40-битовый ключ? Существует всего один триллион возможных ключей. При условии, что наиболее эффективным методом криптоанализа является вскрытие грубой силой (большое допущение, ведь алгоритм никогда не был опубликован), и что микросхема грубого вскрытия может проверить миллион ключей в секунду, поиск правильного ключа займет 12.7 дней. Тысяча машин, работающих параллельно, смогут раскрыть ключ за двадцать минут.

RSA Data Security, Inc., утверждает, что, хотя шифрование и дешифрирования выполняются для быстро, исчерпывающего поиска потребуется намного больше времени. Заметное количество времени тратится на формирование плана использования ключа. Хотя это время пренебрежимо мало при шифровании и дешифрировании сообщений, это не так при проверке каждого возможного ключа.

Правительство США никогда не позволило бы экспортировать любой алгоритм, который оно, по крайней мере в теории, не смогло бы вскрыть. Оно может создать магнитную ленту или CD с конкретным блоком открытого текста, зашифрованным каждым возможным ключом. Для вскрытия сообщения остается только вставить ленту и сравнить блоки шифротекста в сообщении с блоками шифротекста на ленте. При совпадении можно проверить возможный ключ и посмотреть, имеет ли сообщение какой-нибудь смысл. Если они выберут часто встречающийся блок (все нули, ASCII-символы пробела, и т.д.), этот метод будет работать. Объем данных, нужный для хранения результатов шифрования 64-битового блока открытого текста всеми 10^{12} возможными ключами, составляет 8 терабайтов - вполне реально.

2.5.5. Алгоритм MMB

Недовольство использованием в IDEA 64-битового блока шифрования привело к созданию Джоном Дэймоном алгоритма под названием MMB (Modular Multiplication-based Block cipher, модульный блочный шифр, использующий умножения). В основе MMB лежит теория, используемая и в IDEA: перемешивающие операции из различных групп. MMB - это итеративный алгоритм, главным образом состоящий из линейных действий (XOR и использование ключа) и параллельное использование четырех больших нелинейных изменяющих обычный порядок подстановок. Эти подстановки определяются с помощью умножения по модулю 2^{32}-1 с постоянными множителями. Результатом применения этих действий является алгоритм, использующий и 128-битовый ключ и 128-битовый блок.

MMB оперирует 32-битовыми подблоками текста (X_0, X_1, x_2, x_3) и 32-битовыми подблоками ключа (k_0, k_1, k_2, k_3). Это делает удобным реализацию алгоритма на современных 32-битовых процессорах. Чередуясь с XOR, шесть раз используется нелинейная функция f. Вот этот алгоритм (все операции с индексами выполняются по модулю 3):

$x_i = x_i \oplus k_i$, для $i = 0$ до 3
$f(x_0, x_1 x_2, x_3)$
$x_i = x_i \oplus k_{i+1}$, для $i = 0$ до 3
$f(x_0, x , x_2, x_3)$
$x_i = x_i \oplus k_{i+2}$, для $i = 0$ до 3
$f(x_0, x , x_2, x_3)$
$x_i = x_i \oplus k_i$ для $i = 0$ до 3
$f(x_0, x_1, x_2, x_3)$
$x_i = x_i \oplus k_{i+1}$, для $i = 0$ до 3
$f(x_0, x_1, x_2, x_3)$
$x_i = x_i \oplus k_{i+2}$, для $i = 0$ до 3
$f(x_0, x_1, x_2, x_3)$

У функции f три этапа:

(1) $x_1 = c_i * x_i$, для $i = 0$ до 3 (Если на входе умножения одни единицы, то на выходе - тоже одни единицы.)

(2) Если младший значащий бит $x_0 = 1$, то $x_0 = x_0 \oplus C$. Если младший значащий бит $x_3 = 0$, то $x_3 = x_3 \oplus C$.

(3) $x_i = x_{i-1} \oplus x_i \oplus x_{i+1}$, для $i = 0$ до 3

Все операции с индексами выполняются по модулю 3. Операция умножения на этапе (1) выполняется по модулю 2^{32}-1. В данном алгоритме если второй операнд - это 2^{32}-1, то результат также равен 2^{32}-1. В алгоритме используются следующие константы:

C = 2aaaaaaa

c_0= 025flcdb

$c_1 = 2 * c_0$

$c_2 = 2^3 * c_0$ $c_3 = 2^7 * c_0$

Константа C - это "простейшая" константа с высоким троичным весом, нулевым младшим значащим битом и без круговой симметрии. У константы c_0 несколько иные характеристики. Константы c_1, c_2 и c_3 являются смещенными версиями c_0, и используются для предотвращения вскрытий основанных на симметрии.

Дешифрирование является обратным процессом. Этапы (2) и (3) заменяются на свою инверсию. На этапе (1) вместо c_i^{-1} используется c_i c_i^{-1} = 0dad4694.

Безопасность MMB

Схема MMB обеспечивает на каждом этапе значительное и независимое от ключа рассеяние. В IDEA ра с-сеяние до определенной степени зависит от конкретных подключей. В отличие от IDEA у MMB нет слабых ключей.

К сожалению MMB - это умерший алгоритм. Это утверждение справедливо по многим причинам, хотя криптоанализ MMB и не был опубликован. Во первых, он проектировался без учета требований устойчивости к линейному криптоанализу. Выбор мультипликативных множителей обеспечил устойчивость к дифференциальному криптоанализу, но о линейном криптоанализе авторам алгоритма было еще неизвестно.

Во вторых, Эли Бихам реализовал эффективное вскрытие с выбранным ключом, использующеее тот факт, что все этапы идентичны, а ключ при использовании просто циклически сдвигается на 32 бита. В третьих, несмотря на то, что программные реализации MMB были бы очень эффективны, в аппаратном исполнении алгоритм менее эффективен, чем DES.

Дэймон предлагает, что тот, кто захочет улучшить MMB, должен сначала проанализировать умножение по модулю с помощью линейного криптоанализа и подобрать новый множитель, а затем сделать константу C различной для каждого этапа. Затем, улучшив использование ключа, добавляя к ключам этапов константы с целью устранения смещения. Но сам

не стал заниматься этим и разработал 3-Way.

2.5.6. Алгоритм CA-1.1

CA - это блочный шифр, основанный на клеточных автоматах и разработанный Говардом Гутовицом (Howard Gutowitz). Он шифрует 384-битовые блоки открытого текста 1088-битовым ключом (на самом деле используется два ключа - 1024-битовый и 64- битовый). Из-за природы клеточных автоматов алгоритм наиболее эффективен при реализации в больших параллельных интегрированных схемах.

CA-1.1 использует как обратимые, так и необратимые правила клеточного автомата. При обратимом правиле каждое состояние структуры получается из единственного предшествующего состояния, а при необратимом правиле у каждого состояния может быть несколько предшественников. При шифровании необратимые правила пошагово обращаются во времени. Для продвижения обратно от текущего состояния случайным образом должно выбираться одно из состояний-предшественников. Этот процесс многократно повторяется. Таким образом, обратная итерация служит для смешивания случайной информации с информацией сообщения. CA-1.1 использует особый сорт частично линейного необратимого правила, такого, что для любого данного состояния может быть быстро построено случайное состояние-предшественник. На некоторых стадиях шифрования используются и обратимые правила.

Обратимые правила (простые параллельные перестановки подблоков состояния) нелинейны. Необратимые правила полностью определяются ключом, а обратимые зависят как от ключа, так и от случайной информации, вставленной в ходе шифрования необратимыми правилами.

CA-1.1 основан на структуре блочных связей. То есть, обработка блока сообщения частично отделена от обработки потока случайной информации, вставленной при шифровании. Эта случайная информация служит для связи друг с другом стадий шифрования. Она также может быть использована для связи с потоком шифротекста. Информация связи генерируется как часть шифрования.

Так как CA-1.1 представляет собой новый алгоритм, слишком рано делать какие-либо заявления о его безопасности. Гутовиц упоминает некоторые возможные вскрытия, включая дифференциальный криптоанализ, но ему не удалось вскрыть алгоритм. В качестве стимула Гутовиц предложил награду в 1000 долларов для "первого человека, который разработает доступную процедуру вскрытия CA-1.1."

CA-1.1 запатентован, но доступен для некоммерческого использования.

2.5.7. Алгоритм SKIPJACK

Skipjack разработан NSA в качестве алгоритма шифрования для микросхем Clipper и Capstone. Так как этот алгоритм объявлен секретным,

его подробности никогда не публиковались. Он будет реализован только как защищенная от взлома аппаратура.

Этот алгоритм объявлен секретным не потому, что это повышает его надежность, а потому что NSA не хочет, чтобы Skipjack использовался без механизма условного вручения ключей Clipper. Агентство не хочет, чтобы программные реализации алгоритма распространились по всему миру.

Безопасен ли Skipjack? Если NSA захочет создать безопасный алгоритм, оно, скорее всего, это сделает. С другой стороны, если NSA захочет создать алгоритм с лазейкой, то оно сможет сделать и это. Вот что было опубликовано.

— Это итеративный блочный шифр.

— Размер блока - 64 бита.

— Алгоритм использует 80-битовый ключ.

— Он может быть использован в режимах ECB, CBC, 64-битовый OFB, либо 1-, 8-, 16-, 32- или 64-битовый CFB.

— Операция шифрования или дешифрирования состоит из 32 этапов.

— NSA начало работу над ним в 1985 и завершило проверку в 1990.

В документации на микросхему Mykotronx Clipper утверждается, что задержка в выдаче результата, присущая алгоритму Skipjack, составляет 64 такта. Это означает, что на каждый этап приходится два такта: один предположительно для подстановки с помощью S-блока, а другой - для заключительного XOR в конце каждого этапа. (Не забывайте, перестановки при аппаратных реализациях не занимают времени.) В документации Mykotronx эта двухтактная операция называется "G-блоком", а все вместе - "сдвигом". (Часть G-блока носит название "F-таблицы" и является таблицей констант, а может быть таблицей функций.)

По одним слухам Skipjack использует 16 S-блоков, а по другим для хранения S-блоков нужно всего 128 байт памяти. Непохоже, чтобы оба этих слуха были правдой.

Еще один слух утверждает, что этапы Skipjack, в отличие от DES, работают не с половиной блока. Это вместе с замечанием о "сдвигах" и случайном заявлении на Crypto '94 о том, что в Skipjack применяется "48-битовая внутренняя структура", позволяет сделать вывод, что алгоритм по своей схеме похож на SHA, но использует четыре 16-битовых подблока. Три подблока, обработанные зависящей от ключа однонаправленной функцией, дают 16 битов, которые подвергаются операции XOR с оставшимся подблоком. Затем весь блок циклически сдвигается на 16 битов и поступает на вход следующего этапа, или сдвига. При этом также используются 128 байтов данных S-блока.

По своей структуре Skipjack вероятно похож на DES. NSA понимает, что его защищенная от взлома аппаратура в конце концов будет вскрыта и исследована, они не будут рисковать никакими передовыми криптографическими методами.

То, что NSA планирует использовать алгоритм Skipjack для шифрования своей Системы защиты сообщений (Defense Messaging System,

DMS), свидетельствует о безопасности алгоритма. Чтобы убедить скептиков, NIST разрешил комиссии "уважаемых неправительственных экспертов . . . получить доступ к конфиденциальным подробностям алгоритма, чтобы они исследовали его возможности и опубликовали результаты своих исследований".

В предварительном отчете этой комиссии экспертов (окончательного отчета не было, и возможно никогда не будет) сообщалось:

Принимая во внимание, что стоимость вычислительных мощностей уменьшается в два раза каждые 18 месяцев, сложность вскрытия Skipjack сравняется с сегодняшней сложностью вскрытия DES только через 36 лет. Следовательно, риск, что Skipjack будет взломан в ближайшие 30-40 лет, незначителен.

Незначителен и риск взлома Skipjack с помощью более быстрых способов вскрытия, включая дифференциальный криптоанализ. У алгоритма не слабых ключей, отсутствует и свойство комплиментарности. Эксперты в отсутствие времени для самостоятельного большого исследования алгоритма изучили представленное NSA описание разработки и проверки алгоритма

Устойчивость Skipjack к криптоанализу не зависит от хранения в тайне самого алгоритма.

Итак, участники дискуссии не смогли поработать с алгоритмом достаточно долго, чтобы прийти к каким-нибудь выводам самостоятельно. Все, что они смогли сделать - это взглянуть на результаты, показанные им NSA.

Остался без ответа вопрос, является ли плоским пространство ключей Skipjack. Даже если у Skipjack нет ключей, слабых в смысле DES, ряд особенностей процесса использования ключа может сделать одни ключи сильнее других. У Skipjack может быть 2^{70} сильных ключей, гораздо больше чем у DES, вероятность случайно выбрать один из этих сильных ключей будет приблизительно 1 к 1000.

Skipjack запатентован, но в соответствии с соглашением о секретности патента этот патент хранится в тайне. Патент будет опубликован тогда и только тогда, когда алгоритм Skipjack будет успешно восстановлен кем-то посторонним. Это дает возможность правительству воспользоваться и преимуществом защиты патентом, и преимуществом конфиденциальности торгового секрета.

2.6. ОБЪЕДИНЕНИЕ БЛОЧНЫХ ШИФРОВ

Существует множество способов объединять блочные алгоритмы для получения новых алгоритмов. Стимулом создавать подобные схемы является желание повысить безопасность, не пробираясь через тернии создания нового алгоритма. DES является безопасным алгоритмом, он подвергался криптоанализу добрых 20 лет и, тем не менее, наилучшим способом вскрытия остается грубая сила. Однако ключ слишком короток. Разве не хорошо было бы использовать DES в качестве компонента другого алгоритма с более длинным ключом? Это позволило бы получить преимущества длинного ключа с гарантией двух десятилетий криптоанализа.

Одним из способов объединения является **многократное шифрование** - для шифрования одного и того же блока открытого текста алгоритм шифрования используется несколько раз с несколькими ключами. Шифрование каскадом похоже на многократное шифрование, но использует различные алгоритмы . Существуют и другие методы.

Повторное шифрование блока открытого текста одним и тем же ключом с помощью того же или другого алгоритма неразумно. Повторное использование того же алгоритма не увеличивает сложность вскрытия грубой силой. (Мы предполагаем, что алгоритм, включая количество шифрований, известен криптоаналитику.) При различных алгоритмах сложность вскрытия грубой силой может возрастать, а может и остаться неизменной. Если вы собираетесь использовать методы, описанные в этой главе, убедитесь, что ключи для последовательных шифрований различны и независимы.

2.6.1.Двойное шифрование

Наивным способом повысить безопасность алгоритма является шифрование блока дважды с двумя различными ключами. Сначала блок шифруется первым ключом, а затем получившийся шифротекст шифруется вторым ключом. Дешифрирование является обратным процессом.

$C = E_{K2}(E_{K1}(P))$

$P = D_{K1}(D_{K2}(C))$

Если блочный алгоритм образует группу, то всегда существует K_3, для которого

$C = E_{K2}(E_{K1}(P)) = E_{K3}(P)$

Если алгоритм не образует группу, то при помощи исчерпывающего поиска взломать получающийся дважды зашифрованный блок шифротекста намного сложнее. Вместо 2^n (где n - длина ключа в битах), потребуется 2^{2n} попыток. Если алгоритм использует 64-битовый ключ, для обнаружения ключей, которыми дважды зашифрован шифротекст, потребуется 2^{128} попыток.

Но при вскрытии с известным открытым текстом это не так. Меркл и

Хеллман придумали способ обменять память на время, который позволяет вскрыть такую схему двойного шифрования за 2^{n+1} шифрований, а не за 2^{2n}. (Они использовали эту схему против DES, но результаты можно обобщить на все блочные алгоритмы.) Это вскрытие называется **"встреча посередине",** с одной стороны выполняется шифрование а с другой - дешифрирование, получившиеся посередине результаты сравниваются.

В этом вскрытии криптоаналитику известны P_1, C_1, P_2 и C_2, такие что

$C_1=E_{K2}(E_{K1}(P_1))$

$C_2=E_{K2}(E_{K1}(P_2))$

Для каждого возможного K (или K_1, или K_2), криптоаналитик рассчитывает $E_K(P_1)$ и сохраняет результат в памяти. Собрав все результаты, он для каждого K вычисляет $D_K(C_1)$ и ищет в памяти такой же результат. Если такой результат обнаружен, то возможно, что текущий ключ - K_2, а ключ для результата в памяти - K_1. Затем криптоаналитик шифрует P_1 с помощью K_1 и K_2. Если он получает C_2, то он может гарантировать (с вероятностью успеха 1 к 2^{2n-2m}, где m - размер блока), что он узнал и K_1, и K_2. Если это не так, он продолжает поиск. Максимальное количество попыток шифрования, которое ему, возможно, придется предпринять, равно $2*2^n$, или 2^{n+1}. Если вероятность ошибки слишком велика, он может использовать третий блок шифротекста, обеспечивая вероятность успеха 1 к 2^{2n-3m}. Существуют и другие способы оптимизации.

Для такого вскрытия нужен большой объем памяти: 2^n блоков. Для 56-битового ключа нужно хранить 2^{56} 64-битовых блоков, или 10^{17} байтов. Такой объем памяти пока еще трудно себе представить, но этого хватает, чтобы убедить криптографов в том, что двойным шифрованием пользоваться не стоит.

При 128-битовом ключе для хранения промежуточных результатов потребуется 10^{39} байтов. Вскрытие "встреча посередине" кажется невозможным для ключей такого размера.

Другим способом двойного шифрования, который иногда называют Davies-Price, является вариантом CBC.

$C_i=E_{K1}(P \oplus E_{Kг}(C_{i-1}))$ $P_i=D_{K1}(C_i) \oplus E_{K2}(C_{i-1}))$

Утверждается, что "у этого режима нет никаких особых достоинств", к тому же он, по видимому, так же чувствителен ко вскрытию "встреча посередине" как и другие режимы двойного шифрования.

2.6.2.Тройное шифрование

Тройное шифрование с двумя ключами. Воспользуемся [17]. В более интересном методе, предложенном Тачменом, блок обрабатывается три раза с помощью двух ключей: первым ключом, вторым ключом и снова первым ключом. Он предлагает, чтобы отправитель сначала шифровал первым ключом, затем дешифрировал вторым, и окончательно шифровал первым ключом. Получатель расшифровывает первым ключом, затем шифрует вторым и, наконец, дешифрирует первым.

$C = E_{K1}(D_{K2}(E_{K1}(P)))$ $P = D_{K1}(E_{K2}(D_{K1}(C)))$

Иногда такой режим называют **шифрование-дешифрирование-шифрование** (encrypt-decrypt-encrypt, EDE). Если блочный алгоритм использует n-битовый ключ, то длина ключа описанной схемы составляет 2n бит. Любопытный вариант схемы шифрование-дешифрирование-шифрование был разработан в IBM для совместимости с существующими реализациями алгоритма: задание двух одинаковых ключей эквивалентно одинарному шифрованию, этим ключом. Схема шифрование-дешифрирование-шифрование сама по себе не обладает никакой безопасностью, но этот режим был использован для улучшения алгоритма DES в стандартах X9.7 и ISO 8732.

Ключи K_1 и K_2 чередуются для предотвращения описанного выше вскрытия "встреча посередине". Если $C = E_{K1}(E_{K1}(E_{K1}(P)))$, то криптоаналитик для любого возможного K_1 может заранее вычислить $E_{K1}(E_{K1}(P))$ и затем выполнить вскрытие. Для этого потребуется только 2^{n+2} шифрований.

Тройное шифрование с двумя ключами устойчиво к такому вскрытию. Но Меркл и Хеллман разработали другой способ размена памяти на время, который позволяет взломать этот метод шифрования за 2^{n-1} действий, используя 2^n блоков памяти.

Для каждого возможного K_2 расшифруйте 0 и сохраните результат. Затем расшифруйте 0 для каждого возможного K_1, чтобы получить P. Выполните тройное шифрование P, чтобы получить C, и затем расшифруйте C ключом K_1. Если полученное значение совпадает с значением (хранящемся в памяти), полученным при дешифрировании 0 ключом K_2, то пара K_1 K_2 является возможным результатом поиска. Проверьте, так ли это. Если нет, продолжайте поиск.

Выполнение этого вскрытия с выбранным открытым текстом требует огромного объема памяти. Понадобится 2^n времени и памяти, а также 2^m выбранных открытых текстов. Вскрытие не очень практично, но все же чувствительность к нему является слабостью алгоритма.

Пауль ван Оорсчот (Paul van Oorschot) и Майкл Винер (Michael Wiener) преобразовали это вскрытие ко вскрытию с известным открытым текстом, для которого нужно p известных открытых текстов. В примере предполагается, что используется режим EDE.

(1) Предположить первое промежуточное значения a.

(2) Используя известный открытый текст, свести в таблицу для каждого возможного K_1 второе промежуточное значение b, при первом промежуточном значении, равном a:

$b=D_{K1}(C)$

где C - это шифротекст, полученный по известному открытому тексту.

(3) Для каждого возможного K_2 найти в таблице элементы с совпадающим вторым промежуточным значение

b: $b=E_{K2}(a)$

(4) Вероятность успеха равно p/m, где p - число известных открытых текстов, а m - размер блока. Если совпадения не обнаружены, выберите другое a и начните сначала.

Вскрытие требует $2^{n+m}/p$ времени и p - памяти. Для DES это равно $2^{120}/p$. Для p, больших 256, это вскрытие быстрее, чем исчерпывающий поиск.

Тройное шифрование с тремя ключами

Если вы собираетесь использовать тройное шифрование, рекомендуются три различных ключа. Общая длина ключа больше, но хранение ключа обычно не является проблемой. Биты дешевы.

$C = E_{K3}(D_{K2}(E_{K1}(P)))$

$P = D_{K1}(E_{K2}(D_{K3}(C))$

Для наилучшего вскрытия с разменом памяти на время, которым является "встреча посередине", потребуется 2^{2n} действий и 2^n блоков памяти. Тройное шифрование с тремя независимыми ключами безопасно настолько, насколько на первый взгляд кажется безопасным двойное шифрование.

Тройное шифрование с минимальным ключом (ТЕМК)

Существует безопасный способ использовать тройное шифрование с двумя ключами, противостоящий описанному вскрытию и называемый Тройным шифрованием с минимальным ключом (Triple Encryption with Minimum Key, ТЕМК). Фокус в той, чтобы получить три ключа из: X_1 *и* X_2.

$K_1=E_{X1}(D_{X2}(E_{X1}(T_1)))$ $K_2=E_{X1}(D_{X2}(E_{X2}(T_2)))$ $K_3 = E_{X1}(D_{X2}(E_{X1}(T_3)))$

T_1, T_2*и* T_3 представляют собой константы, которые необязательно хранить в секрете. Эта схема гарантирует, что для любой конкретной пары ключей наилучшим будет вскрытие с известным открытым текстом .

Режимы тройного шифрования

Недостаточно просто определить тройное шифрование, нужно выбрать один из способов его использования. Решение зависит от требуемых безопасности и эффективности. Вот два возможных режима тройного шифрования:

Внутренний CBC: Файл три раза шифруется в режиме CBC. Для этого нужно три различных IV.

C_0, S_0 и T_0 являются IV.

Внешний CBC: Файл троекратно шифруется в режиме CBC. Для этого нужен один IV.

$C_i = E_{K3}(D_{K2}(E_{K1}(P_i \oplus C_{i-1})))$

$P_i = C_{i-1} \oplus D_{K1}(E_{K2}(D_{K3}(C_i)))$

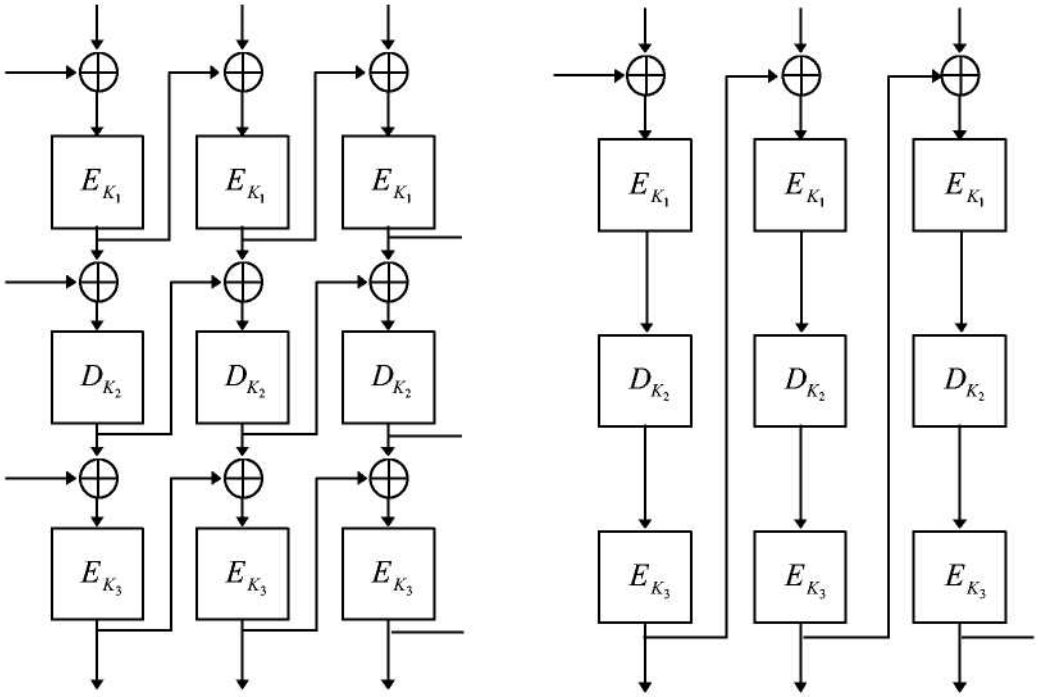


Рис. 25 Тройное шифрование в режиме CBC [17].

Для обоих режимов нужно больше ресурсов, чем для однократного шифрования: больше аппаратуры или больше времени. Однако при трех шифрующих микросхемах производительность внутреннего CBC не меньше, чем при однократном шифровании. Так как три шифрования CBC независимы, три микросхемы могут быть загружены постоянно, подавая свой выход себе на вход.

Напротив, во внешнем CBC обратная связь находится снаружи по отношению к трем шифрованиям. Это означает, что даже с тремя микросхемами производительность будет равна только одной трети производительности при однократном шифровании. Чтобы получить ту же производительность для внешнего CBC, потребуется чередование IV:

$C_i=E_{K3}\{D_{K2}(E_{K1}(P_i \oplus C_{i-3}))$

В этом случае C_0, C_{-1} и C_{-2} являются IV. Это не поможет при программной реализации, разве только при использовании параллельного компьютера.

К сожалению, менее сложный режим является также и менее безопасным. Бихам проанализировал различные режимы по отношению к дифференциальному криптоанализу и обнаружил, что безопасность внутреннего CBC по сравнению с однократным шифрованием увеличивается незначительно. Если рассматривать тройное шифрование как единый большой алгоритм, то внутренние обратные связи позволяют вводить внешнюю и известную информацию внутрь алгоритма, что облегчает криптоанализ . Для дифференциальных вскрытий нужно огромное количество выбранных шифротекстов, что делает эти вскрытия не слишком практичными, но этих результатов должно хватить, чтобы насторожить

пользователей . Анализ устойчивости алгоритмов к вскрытиям грубой силой и "встречей посередине" показал, что оба варианта одинаково безопасны.

Кроме этих существуют и другие режимы. Можно зашифровать файл один раз в режиме ECB, затем дважды в CBC, или один раз в CBC, один в ECB и еще раз в CBC, или дважды в CBC и один раз в ECB. Бихам показал, что эти варианты не безопаснее, чем однократный DES, против вскрытия дифференциальным криптоанализом с выбранным открытым текстом. Он не оставил больших надежд и для других вариантов.

Варианты тройного шифрования

Прежде, чем появились доказательства того, что DES не образует группу, для многократного шифрования предлагались различные схемы. Одним из способов обеспечить то, что тройное шифрование не выродится в однократное, было изменение эффективной дины блока. Простым методом является добавление бита-заполнителя. Между первым и вторым, а также между вторым и третьим шифрованиями текст дополняется строкой случайных битов. Если *PP* - это функция дополнения, то: $C = E_{K3}(PP(E_{K2}(PP(E_{K1}(P)))))$.

Это дополнение не только разрушает шаблоны, но также обеспечивает перекрытие блоков шифрования, как кирпичей в стене. К длине сообщения добавляется только один блок.

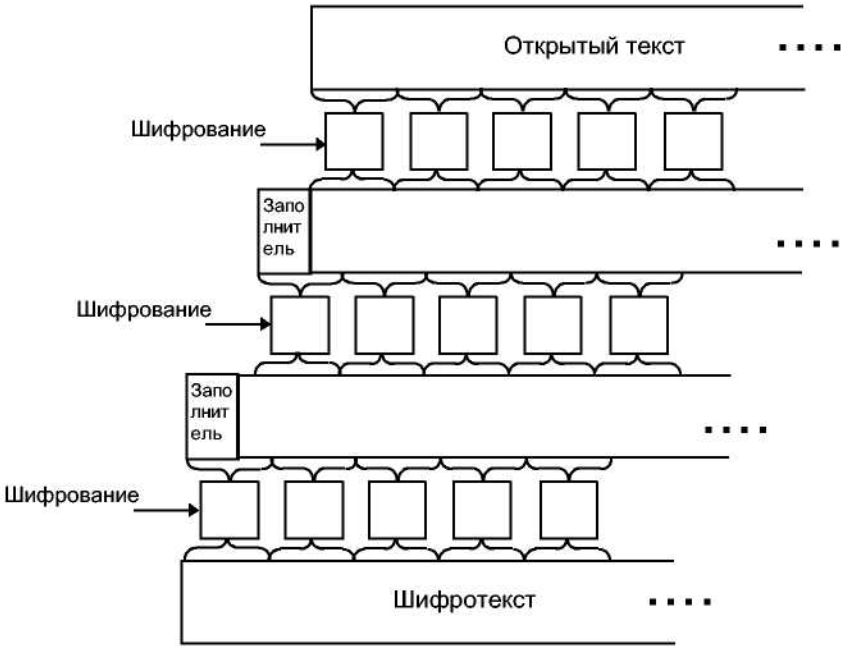


Рис. 26 Тройное шифрование с заполнением.

Другой метод, предложенный Карлом Эллисоном (Carl Ellison), использует некоторую функцию независимой от ключа перестановки между тремя шифрованиями. Перестановка должна работать с большими блоками - 8 Кбайт или около этого, что делает эффективный размер блока для этого варианта равным 8 Кбайтам. При условии, что перестановка выполняется

быстро, этот вариант ненамного медленнее, чем базовое тройное шифрование.

$C = E_{K3}(T(E_{K2}(T(E_{K1}(P)))))$

T собирает входные блоки (до 8 Кбайт в длину) и использует генератор псевдослучайных чисел для их перемешивания. Изменение одного бита входа приводит к изменению 8 байтов результата первого шифрования, к изменению до 64 байтов результата второго шифрования и к изменению до 512 байтов результата третьего шифрования. Если каждый блочный алгоритм работает в режиме CBC, как было первоначально предложено, то изменение единичного бита входа скорее всего приведет к изменению всего 8-килобайтового блока, даже если этот блок не является первым.

Самый последний вариант этой схемы отвечает на вскрытие внутреннего CBC, выполненное Бихамом, добавлением процедуры отбеливания, чтобы замаскировать структуру открытых текстов. Эта процедура представляет собой потоковую операцию XOR с криптографически безопасным генератором псевдослучайных чисел и ниже обозначена как *R. T* мешает криптоаналитику определить *a priori,* какой ключ используется для шифрования любого заданного байта входа последнего шифрования. Второе шифрование обозначено *nE* (шифрование с циклическим использованием *n* различных ключей):

$C = E_{K3}(R(T(nE_{K2}(T(E_{K1}(P))))))$

Все шифрования выполняются в режиме ECB, используется не меньше *n+2* ключей шифрования и криптографически безопасный генератор псевдослучайных чисел.

Эта схема была предложена для использования вместе с DES, но она работает с любым блочным алгоритмом.

2.6.3. Другие схемы многократного шифрования

Проблемой тройного шифрования с двумя ключами является то, что для увеличения вдвое пространства ключей нужно выполнять три шифрования каждого блока открытого текста. От этого недостатка свободна следующая схема.

Удвоение длины блока. Существуют предложения удваивать длину блока алгоритма с помощью многократного шифрования. Прежде, чем реализовывать одно из них, оцените возможность вскрытия "встреча посередине". Схема Ричарда Аутбриджа (Richard Outerbridge), не более безопасна, чем тройное шифрование с одинарным блоком и двумя ключами.

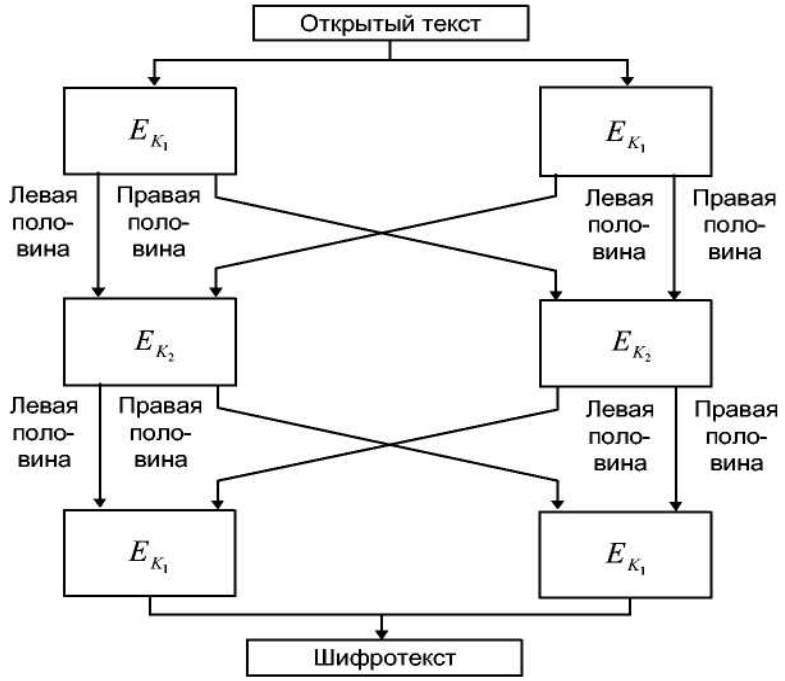


Рис. 27 Удвоение длины блока.

Однако он не быстрее обычного тройного шифрования: для шифрования двух блоков данных все также нужно шесть шифрований. Характеристики обычного тройного шифрования известны, а за новыми конструкциями часто прячутся новые проблемы.

Двойной OFB/счетчик

Этот метод использует блочный алгоритм для генерации двух потоков ключей, которые используются для шифрования открытого текста.

$S_i=E_{K1}(S_{i-1} \oplus I_1)$; $I_1=I_1+1$ $T_i = E_{K2}(T_{i-1} \oplus I_2)$; $I_2=I_2+1$

$C_i= P_i \oplus S_i \oplus T_i$

S_i и T_i - внутренние переменные, а I_1 и I_2 - счетчики. Две копии блочного алгоритма работают в некотором гибридном режиме OFB/счетчик, а открытый текст, S_i и T_i, объединяются с помощью XOR. Ключи K_1 и K_2 независимы.

ECB + OFB

Этот метод был разработан для шифрования нескольких сообщений фиксированной длины, например, блоков диска. Используются два ключа: K_1 и K_2. Сначала для генерации маски для блока нужной длины используется выбранный алгоритм и ключ. Эта маска будет использована повторно для шифрования сообщений теми же ключами. Затем выполняется XOR открытого текста сообщения и маски. Наконец результат XOR шифруется с помощью выбранного алгоритма и ключа K_2 в режиме ECB.

Как отмечает Шнайер в [17], анализ этого метода проводился только в той работе, в которой он и был опубликован. Он не слабее одинарного шифрования ECB и возможно также силен, как и двойное применение алгоритма. Вероятно, криптоаналитик может выполнять поиск ключей

независимо, если он получит несколько открытых текстов файлов, зашифрованных одним ключом.

Чтобы затруднить анализ идентичных блоков в одних и тех же местах различных сообщений, можно использовать IV. В отличии от использования IV в других режимах в данном случае перед шифрованием ECB выполняется XOR каждого блока сообщения с IV.

Мэтт Блэйз (Matt Blaze) разработал этот режим для своей UNIX Cryptographic File System (CFS, криптографическая файловая система). Это хороший режим, поскольку скрытым состоянием является только одно шифрование в режиме ECB, маска может быть сгенерирована только один раз и сохранена. В CFS в качестве блочного алгоритма используется DES.

xDES[1]

DES используется как компонент ряда блочных алгоритмов с увеличенными размерами ключей и блоков. Эти схемы никак не зависят от DES, и в них может использоваться любой блочный алгоритм.

Первый, xDES[1], представляет собой просто схему Luby-Rackoff с блочным шифром в качестве базовой функции. Размер блока в два раза больше размера блока используемого блочного фильтра, а размер ключа в три раза больше, чем у используемого блочного фильтра. В каждом из 3 этапов правая половина шифруется блочным алгоритмом и одним из ключей, затем выполняется XOR результата и левой половины, и половины переставляются.

Это быстрее, чем обычное тройное шифрование, так как тремя шифрованиями шифруется блок, длина которого в два раза больше длины блока используемого блочного алгоритма. Но при этом существует простое вскрытие "встреча посередине", которое позволяет найти ключ с помощью таблицы размером 2^k, где k - это размер ключа блочного алгоритма. Правая половина блока открытого текста шифруется с помощью всех возможных значений K_1, выполняется XOR с левой половиной открытого текста и полученные значения сохраняются в таблице. Затем правая половина шифротекста шифруется с помощью всех возможных значений K_3, и выполняется поиск совпадений в таблице. При совпадении пара ключей K_1 и K_3 - возможный вариант правого ключа. После нескольких повторений вскрытия останется только один кандидат. Таким образом, xDES[1] не является идеальным решением. Существует вскрытие с выбранным открытым текстом, доказывающее, что xDES[1] не намного сильнее используемого в нем блочного алгоритма.

В xDES[2] эта идея расширяется до 5-этапного алгоритма, размер блока которого в 4 раза, а размер ключа в 10 раз превышают размеры блока и ключа используемого блочного шифра. Каждый из четырех подблоков по размеру равен блоку используемого блочного шифра, а все 10 ключей независимы.

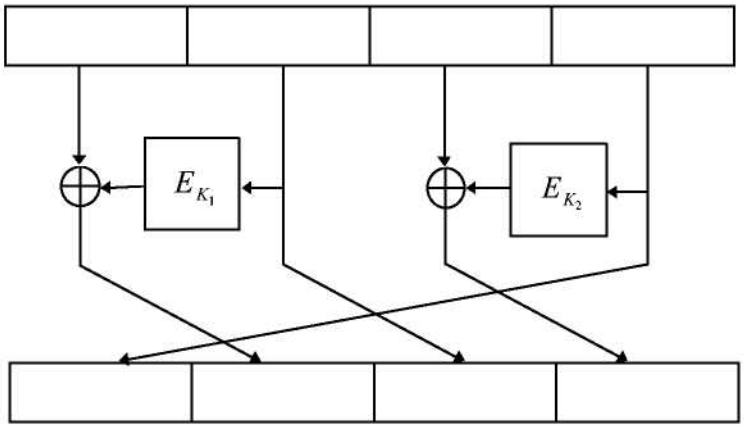


Рис. 28 Один 3TanxDES[2].

К тому же, эта схема быстрее, чем тройное шифрование: для шифрования блока, который в четыре раза больше блока используемого блочного шифра, нужно 10 шифрований. Однако этот метод чувствителен к дифференциальному криптоанализу и использовать его не стоит. Такая схема остается чувствительной к дифференциальному криптоанализу, даже если используется DES с независимыми ключами этапов.

Для $i \geq 3$ xDES' вероятно слишком велик, чтобы использовать его в качестве блочного алгоритма. Например, размер блока для $xDES^3$ в 6 раз больше, чем у лежащего в основе блочного шифра, ключ в 21 раз длиннее, а для шифрования блока, который в 6 раз длиннее блока лежащего в основе блочного шифра, нужно 21 шифрование. Это медленнее, чем тройное шифрование.

Пятикратное шифрование

Если тройное шифрование недостаточно безопасно - может быть, вам нужно шифровать ключи тройного шифрования, используя еще более сильный алгоритм - то кратность шифрования можно увеличить. Очень устойчиво к вскрытию "встреча посередине" пятикратное шифрование . (Аргументы, аналогичные рассмотренным для двойного шифрования, показывают, что четырехкратное шифрование по сравнению с тройным лишь незначительно повышает надежность.)

$C = E_{K1}(D_{K2}(E_{K3}(D_{K2}(E_{K1}(P)))))$

$P = D_{K1}(E_{K2}(D_{K3}(E_{K2}(D_{K1}(C)))))$

Эта схема обратно совместима с тройным шифрованием, если $K_1 = K_2$, и с однократным шифрованием, если $K_1 = K_2 = K_3$. Конечно, она будет еще надежней, если использовать пять независимых ключей.

Уменьшение длины ключа в CDMF

Этот метод был разработан IBM для продукта CDMF (Commercial Data Masking Facility, Коммерческое средство маскирования данных), чтобы превратить 56-битовый ключ DES в 40-битовый, разрешенный для экспорта. Предполагается, что первоначальный ключ DES содержит биты четности.

(1) Обнуляются биты четности: биты 8, 16, 24, 32,40, 48, 56, 64.

(2) Результат этапа (1) шифруется с помощью DES ключом 0xc408b0540bale0ae, результат шифрования объединяется посредством XOR с результатом этапа (1).

(3) В результате этапа (2) обнуляются следующие биты: 1, 2, 3, 4, 8, 16, 17, 18, 19, 2.0, 2.4, 32, 33, 34, 35, 36, 40,48,49,50,51,52,56,64.

(4) Результат этапа (3) шифруется с помощью DES ключом 0xef2c041ce6382fe6. Полученный ключ используется для шифрования сообщения.

Однако этот метод укорачивает ключ и, следовательно, ослабляет алгоритм.

Отбеливание

Отбеливанием (whitening) называется способ, при котором выполняется XOR части ключа с входом блочного алгоритма и XOR другой части ключа с выходом блочного алгоритма. Впервые этот метод был применен для варианта DESX, разработанного RSA Data Security, Inc., а затем (по-видимому, независимо) в Khufu и Khafre. (Ривест и дал имя этому методу)

Смысл этих действий в том, чтобы помешать криптоаналитику получить пару "открытый текст/шифротекст" для лежащего в основе блочного алгоритма. Метод заставляет криптоаналитика угадывать не только ключ алгоритма, но и одно из значений отбеливания. Так как XOR выполняется и перед, и после блочного алгоритма, считается, что этот метод устойчив против вскрытия "встреча посередине".

$C = K_3 \oplus E_{K2}(P \oplus K_1)$

$P = K_1 \oplus D_{K2}(C \oplus K_3)$

Если $K_1 = K_2$, то для вскрытия грубой силой потребуется $2^{n+m/p}$ действий, где n - размер ключа, m - размер блока, и p - количество известных открытых текстов. Если K_1 и K_2 различны, то для вскрытия грубой силой с тремя известными открытыми текстами потребуется 2^{n+m+1} действий. Против дифференциального и линейного криптоанализа, такие меры обеспечивают защиту только для нескольких битов ключа. Но с вычислительной точки зрения это очень дешевый способ повысить безопасность блочного алгоритма.

2.6.4. Многократное последовательное использование блочных алгоритмов

А как насчет шифрования сначала алгоритмом А и ключом K_A, а затем еще раз алгоритмом В и ключом K_B? Может быть у А и В различные представления о том, какой алгоритм безопаснее: А хочет пользоваться алгоритмом А, а В - алгоритмом В. Этот прием, иногда называемый **последовательным использованием** (cascading), можно распространить и на большее количество алгоритмов и ключей.

Пессимисты утверждали, что совместное использование двух алгоритмов не гарантирует повышения безопасности. Алгоритмы могут взаимодействовать каким-то хитрым способом, что на самом деле даже *уменьшит.* Даже тройное шифрование тремя различными алгоритмами может не быть настолько безопасным, насколько это кажется.

Упомянутые предостережения верны, только если различные ключи зависят друг от друга. Если все используемые ключи независимы, то сложность взлома последовательности алгоритмов по крайней мере не меньше, чем сложность взлома первого из применяемых алгоритмов. Если второй алгоритм чувствителен к вскрытию с выбранным открытым текстом, то первый алгоритм может облегчить это вскрытие и при последовательном использовании сделать второй алгоритм чувствительным к вскрытию с известным открытым текстом. Такое возможное облегчение вскрытия не ограничивается только алгоритмами шифрования: если вы позволите кому-то другому определить любой из алгоритмов, делающих что-то с вашим сообщением до шифрования, стоит удостовериться, что ваше шифрование устойчиво по отношению к вскрытию с выбранным открытым текстом.

Это можно сформулировать и иначе: При использовании вскрытия с выбранным открытым текстом последовательность шифров взломать не легче, чем любой из шифров последовательности. Ряд результатов показал, что последовательное шифрование взломать по крайней мере не легче, чем самый сильный из шифров последовательности, но в основе этих результатов лежат некоторые несформулированные предположения. Только если алгоритмы коммутативны, как в случае каскадных потоковых шифров (или блочных шифров в режиме OFB), надежность их последовательности не меньше, чем у сильнейшего из используемых алгоритмов.

Если A и B не доверяют алгоритмам друг друга, они могут использовать их последовательно. Для потоковых алгоритмов их порядок не имеет значения. При использовании блочных алгоритмов A может сначала использовать алгоритм A, а затем алгоритм B. B, который больше доверяет алгоритму B, может использовать алгоритм B перед алгоритмом A. Между алгоритмами они могут вставить хороший потоковый шифр. Это не причинит вреда и может значительно повысить безопасность.

Ключи для каждого алгоритма последовательности должны быть независимыми . Если алгоритм A использует 64-битовый ключ, а алгоритм B - 128-битовый ключ, то получившаяся последовательность должна использовать 192-битовый ключ. При использовании зависимых ключей у пессимистов гораздо больше шансов оказаться правыми.

Объединение нескольких блочных алгоритмов

Вот другой способ объединить несколько блочных алгоритмов, безопасность которого гарантировано будет, по крайней мере, не меньше, чем безопасность обоих алгоритмов. Попытка, на наш взгляд, достаточно успешная, прояснить этот вопрос предпринята в статье [18] A.M.

Кукарцевым с соавторами. Для двух алгоритмов (и двух независимых ключей):

(1) Генерируется строка случайных битов R того же размера, что и сообщение M.

(2) R шифруется первым алгоритмом.

(3) $M \oplus R$ шифруется вторым алгоритмом.

(4) Шифротекст является объединением результатов этапов (2) и (3).

При условии, что строка случайных битов действительно случайна, этот метод шифрует M с помощью одноразового блокнота, а затем содержимое блокнота и получившееся сообщение шифруются каждым из двух алгоритмов. Так как и то, и другое необходимо для восстановления M, криптоаналитику придется взламывать оба алгоритма. Недостатком является удвоение размера шифротекста по сравнению с открытым текстом.

Этот метод можно расширить для нескольких алгоритмов, но добавление каждого алгоритма увеличивает шифротекст.

2.7. АЛГОРИТМЫ С ОТКРЫТЫМИ КЛЮЧАМИ

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, через несколько месяцев была опубликована их основополагающая работа "New Directions in Cryptography" ("Новые направления в криптографии"). (Из-за бесстрастного процесса публикации первый вклад Меркла в эту область появился только в 1978 году.)

С 1976 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. Обычно эти алгоритмы основаны на одной из трудных проблем. 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 и Rabin. Все эти алгоритмы медленны. Они шифруют и дешифрируют данные намного медленнее, чем симметричные алгоритмы. Обычно их скорость недостаточна для шифрования больших объемов данных.

Гибридные криптосистемы позволяют ускорить события: для шифрования сообщения используется симметричный алгоритм со случайным ключом, а алгоритм с открытым ключом применяется для шифрования случайного сеансового ключа.

Так как у криптоаналитика есть доступ к открытому ключу, он всегда может выбрать для шифрования любое сообщение. Это означает, что криптоаналитик при заданном $C=E_K(P)$ может попробовать угадать значение P и легко проверить свою догадку. Это является серьезной проблемой, если количество возможных открытых текстов настолько мало, что делает возможным исчерпывающий поиск, но эту проблему легко можно решить, дополняя сообщения строкой случайных битов. Это приводит к тому, что идентичным открытым текстам соответствуют различные шифротексты.

Это особенно важно, если алгоритм с открытым ключом используется для шифрования сеансового ключа. Ева может создать базу данных всех возможных сеансовых ключей, зашифрованных открытым ключом Боба . Конечно, это потребует много времени и памяти, но взлом грубой силой

разрешенного к экспорту 40-битового ключа или 56-битового ключа DES потребует намного больше времени и памяти. Как только Ева создаст такую базу данных, она получит ключ Боба и сможет читать его почту.

Алгоритмы с открытыми ключами спроектированы так, чтобы противостоять вскрытиям с выбранным открытым текстом. Их безопасность основана как на трудности получения секретного ключа по открытому, так и на трудности получить открытый текст по шифротексту. Однако большинство алгоритмов с открытым ключом особенно чувствительны к вскрытию с выбранным шифротекстом.

В системах, в которых операция, обратная шифрованию, используется для цифровой подписи, это вскрытие невозможно предотвратить, если для шифрования и подписей использовать одинаковые ключи.

Следовательно, важно увидеть всю систему целиком, а не только составные части. Хорошие протоколы с открытыми ключами спроектированы таким образом, чтобы различные стороны не могли расшифровать произвольные сообщения, генерированные другими сторонами, - хорошим примером являются протоколы доказательства идентичности.

2.7.1. Алгоритмы рюкзака

Первым алгоритмом для обобщенного шифрования с открытым ключом стал алгоритм рюкзака, разработанный Ральфом Мерклом и Мартином Хеллманом. Он мог быть использован только для шифрования, хотя позднее Ади Шамир адаптировал систему для цифровой подписи. Безопасность алгоритмов рюкзака опирается на проблему рюкзака, **NP-полную** проблему. Хотя позже было обнаружено, что этот алгоритм небезопасен, его стоит изучить, так как он демонстрирует возможность применения **NP-полной** проблемы в криптографии с открытыми ключами.

Проблема рюкзака несложна. Дана куча предметов различной массы, можно ли положить некоторые из этих предметов в рюкзак так, чтобы масса рюкзака стала равна определенному значению? Более формально, дан набор значений $M_1, M_2, \ldots, M_n$ и сумма S, вычислить значения b_i, такие что

$S = b_1M_1 + b_2M_2 + \ldots + b_nM_n$

b_i может быть либо нулем, либо единицей. Единица показывает, что предмет кладут в рюкзак, а ноль - что не кладут.

Например, массы предметов могут иметь значения 1, 5, 6, 11, 14 и 20. Вы можете упаковать рюкзак так, чтобы его масса стала равна 22, использовав массы 5, 6 и 11. Невозможно упаковать рюкзак так, чтобы его масса была равна 24. В общем случае время, необходимое для решения этой проблемы, с ростом количества предметов в куче растет экспоненциально.

В основе алгоритма рюкзака Меркла-Хеллмана лежит идея шифровать сообщение как решение набора проблем рюкзака. Предметы из кучи выбираются с помощью блока открытого текста, по длине равного количеству предметов в куче (биты открытого текста соответствуют

значениям *b*), а шифротекст является полученной суммой. Пример шифротекста, зашифрованного с помощью проблемы рюкзака.

Открытый текст	111001	010110	000000	011000
Рюкзак	1 5 6 11 14 20	1 5 6 11 14 20	1 5 6 11 14 20	1 5 6 11 14 20
Шифротекст	1+5+6+20=32	5+11+14=30	0=0	5+6=11

Рис. 13 Шифрование с рюкзаками

Фокус в том, что на самом деле существуют две различные проблемы рюкзака, одна решается за линейное время, а другая, как считается, - нет. Легкую проблему можно превратить в трудную. Открытый ключ представляет собой трудную проблему, которую легко использовать для шифрования, но невозможно для дешифрирования сообщений. Закрытый ключ является легкой проблемой, давая простой способ дешифрировать сообщения . Тому, кто не знает закрытый ключ, придется попытаться решить трудную проблему рюкзака .

Сверхвозрастающие рюкзаки

Что такое легкая проблема рюкзака? Если перечень масс представляет собой **сверхвозрастающую последовательность,** то полученную проблему рюкзака легко решить. Сверхвозрастающая последовательность - это последовательность, в которой каждой член больше суммы всех предыдущих членов . Например, последовательность {1,3,6,13,27,52} является сверхвозрастающей, а {1,3,4,9, 15,25} - нет.

Решение **сверхвозрастающего рюкзака** найти легко. Возьмите полный вес и сравните его с самым большим числом последовательности. Если полный вес меньше, чем это число, то его не кладут в рюкзак. Если полный вес больше или равен этому числу, то оно кладется в рюкзак. Уменьшим массу рюкзака на это значение и перейдем к следующему по величине числу последовательности. Будем повторять, пока процесс не закончится. Если полный вес уменьшится до нуля, то решение найдено. В противном случае нет.

Например, пусть полный вес рюкзака - 70, а последовательность весов {2,3,6, 13,27,52}. Самый большой вес, 52, меньше 70, поэтому кладем 52 в рюкзак. Вычитая 52 из 70, получаем 18. Следующий вес, 27, больше 18, поэтому 27 в рюкзак не кладется, вес, 13,меньше 18, поэтому кладем 13 в рюкзак. Вычитая 13 из 18, получаем 5. Следующий вес, 6, больше 5, поэтому 6 не кладется в рюкзак. Продолжение этого процесса покажет, что и 2, и 3 кладутся в рюкзак, и полный вес уменьшается до 0, что сообщает о найденном решении. Если бы это был блок шифрования методом рюкзака Меркла-Хеллмана, открытый текст, полученный из значения шифротекста 70, был бы равен 110101.

Не сверхвозрастающие, или нормальные, рюкзаки представляют собой трудную проблему - быстрого алгоритма для них не найдено. Единственным известным способом определить, какие предметы кладутся в рюкзак, является методическая проверка возможных решений, пока вы не наткнетесь на правильное . Самый быстрый алгоритм, принимая во внимание различную эвритсику, имеет экспоненциальную зависимость от числа возможных предметов. Добавьте к последовательности весов еще один член, и найти решение станет вдвое труднее. Это намного труднее сверхвозрастающего рюкзака, где, если вы добавите один предмет к последовательности, поиск решения увеличится на одну операцию.

Алгоритм Меркла-Хеллмана основан на этом свойстве. Закрытый ключ является последовательностью весов проблемы сверхвозрастающего рюкзака. Открытый ключ - это последовательность весов проблемы нормального рюкзака с тем же решением. Меркл и Хеллман, используя модульную арифметику, разработали способ преобразования проблемы сверхвозрастающего рюкзака в проблему нормального рюкзака.

Создание открытого ключа из закрытого

Рассмотрим работу алгоритма, не углубляясь в теорию чисел: чтобы получить нормальную последовательность рюкзака, возьмем сверхвозрастающую последовательность рюкзака, например, {2,3,6,13,27,52}, и умножим по модулю *m* все значения на число *n.* Значение модуля должно быть больше суммы всех чисел последовательности, например, 105. Множитель должен быть взаимно простым числом с модулем, например, 31. Нормальной последовательностью рюкзака будет

2*31 mod 105 = 62
3*31 mod 105 = 93
6*31 mod 105 = 81
13*31 mod 105 = 88
27*31 mod 105 = 102
52*31 mod 105 = 37
Итого- {62,93,81,88,102,37}.

Сверхвозрастающая последовательность рюкзака является закрытым ключом, а нормальная последовательность рюкзака - открытым.

Шифрование

Для шифрования сообщение сначала разбивается на блоки, равные по длине числу элементов последовательности рюкзака. Затем, считая, что единица указывает на присутствие члена последовательности, а ноль - на его отсутствие, вычисляем полные веса рюкзаков - по одному для каждого блока сообщения .

Например, если сообщение в бинарном виде выглядит как 011000110101101110, шифрование, использующее предыдущую последовательность рюкзака, будет происходить следующим образом:

сообщение = 011000 110101 101110

011000 соответствует 93 + 81 = 174
110101 соответствует 62 + 93 + 88 + 37 = 280
101110 соответствует 62 + 81 + 88 + 102 = 333
Шифротекстом будет последовательность 174,280,333

Дешифрирование

Законный получатель данного сообщения знает закрытый ключ: оригинальную сверхвозрастающую последовательность, а также значения *n* и *m,* использованные для превращения ее в нормальную последовательность рюкзака. Для дешифрирования сообщения получатель должен сначала определить n^{-1}, такое что $n(n^{-1}) \equiv l \pmod{m}$. Каждое значение шифротекста умножается на $n^{-1} \bmod m$, а затем разделяется с помощью закрытого ключа, чтобы получить значения открытого текста.

В нашем примере сверхвозрастающая последовательность - {2,3,6,13,27,52}, *m* равно 105, а и - 31. Шифротекстом служит 174,280,333. В этом случае n^{-1} равно 61, поэтому значения шифротекста должны быть умножены на 61 mod 105.

174*61 mod 105 = 9 = 3 + 6, что соответствует 011000
280*61 mod 105 = 70 = 2 + 3 + 13 + 52, что соответствует 110101
333*61 mod 105 = 48 = 2 + 6 + 13 + 27, что соответствует 101110
Расшифрованным открытым текстом является 011000 110101 101110.

Практические реализации

Для последовательности из шести элементов нетрудно решить задачу рюкзака, даже если последовательность не является сверхвозрастающей. Реальные рюкзаки должны содержать не менее 250 элементов. Длина каждого члена сверхвозрастающей последовательности должна быть где-то между 200 и 400 битами, а длина модуля должна быть от 100 до 200 битов. Для получения этих значений практические реализации используют генераторы случайной последовательности.

Вскрывать подобные рюкзаки при помощи грубой силы бесполезно. Если компьютер может проверять миллион вариантов в секунду, проверка всех возможных вариантов рюкзака потребует свыше 10^{46} лет. Даже миллион машин, работающих параллельно, не успеет решить эту задачу до превращения солнца в сверхновую звезду.

Безопасность метода рюкзака

Взломали криптосистему, основанную на проблеме рюкзака, не миллион машин, а пара криптографов. Сначала был раскрыт единственный бит открытого текста. Затем Шамир показал, что в определенных обстоятельствах рюкзак может быть взломан. Были и другие достижения - но никто не мог взломать систему Мартина-Хеллмана в общем случае. Наконец Шамир и Циппел (Zippel), обнаружили слабые места в преобразовании, что позволило им восстановить сверхвозрастающую последовательность рюкзака

по нормальной. На конференции, где докладывались эти результаты, вскрытие было продемонстрировано по стадиям на компьютере Apple II.

Варианты рюкзака

После вскрытия оригинальной схемы Меркла-Хеллмана было предложено множество других систем на принципе рюкзака: несколько последовательных рюкзаков, рюкзаки Грэм-Шамира (Graham-Shamir), и другие. Все они были проанализированы и взломаны, как правило, с использованием одних и тех же криптографических методов.

Были предложены и другие алгоритмы, использующие похожие идеи, но все они тоже были взломаны. Криптосистема Lu-Lee была взломана, ее модификация также оказалась небезопасной. Вскрытия. Криптосистема Pieprzyk была взломана аналогичным образом.

Хотя вариант алгоритма рюкзака в настоящее время безопасен - алгоритм рюкзака Char-Rivest, несмотря на "специализированное вскрытие" - количество необходимых вычислений делает его намного менее полезным, чем другие рассмотренные здесь алгоритмы. Вариант, названный Powerline System (система электропитания) небезопасен. Более того, учитывая легкость с которой пали все остальные варианты, доверять устоявшим пока вариантом, по видимому, неосторожно.

2.7.2. Алгоритм RSA

Концепция криптографии с открытым ключом была предложена Уитфилдом Диффи (Whitfield Diffie) и Мартином Хеллманом (Martin Hellman), и, независимо, Ральфом Мерклом (Ralph Merkle) Основная идея: использовать ключи парами, состоящими из ключа зашифрования и ключа расшифрования, которые невозможно вычислить один из другого. В 1976 году вышла основополагающая работа [19]. С 1976 г. Было создано много алгоритмов, использующих концепцию открытых ключей. Алгоритм является общедоступным, нет необходимости в секретных каналах связи. Общая схема выглядит следующим образом:

1. Каждый пользователь генерирует пару ключей: один для шифрования и один для дешифрования.
2. Каждый пользователь публикует свой ключ шифрования, размещает его в открытом для всех доступе. Второй ключ, соответствующий открытому, сохраняется в секрете.
3. Если пользователь А собирается послать сообщение пользователю В, он шифрует сообщение открытым ключом пользователя В.
4. Когда пользователь В получает сообщение, он дешифрует его с помощью своего личного (секретного) ключа. Другой получатель не сможет дешифровать сообщение, поскольку личный ключ В знает только В.

Описание алгоритма RSA

В 1978 г. Появилась работа [20], в которой Рон Райвест(Ron Rivest), Ади Шамир(Adi Shamir) и Лен Адлеман(Len Adleman) предложили алгоритм с открытым ключом. Схема Райвеста–Шамира–Адлемана (RSA) получила широкое распространение.

Опишем процесс шифрования. Исходный текст должен быть переведен в числовую форму. Метод преобразования текста в числовую форму считается известным. В результате текст представляется в виде одного большого числа. Затем полученное число разбивается на части (блоки) так, чтобы каждая из них была числом в промежутке *[0,N-1]*, о выборе N — см. ниже. Процесс шифрования одинаков для каждого блока. Поэтому мы можем считать, что блок исходного текста представлен числом x, $0 \le x \le N-1$.

Каждый абонент вырабатывает свою пару ключей. Для этого он генерирует два больших простых числа p и q, вычисляет произведение $N = p \cdot q$. Затем он вырабатывает случайное число e, взаимно простое со значением функцией Эйлера от числа N, $\varphi(N) = (p-1)\cdot(q-1)$ и находит число d из условия $e \cdot d = 1(\mathrm{mod}\,\varphi(N))$. Так как $(e,\varphi(N)) = 1$, то такое число d существует и единственно. Пару (N,e) он объявляет открытым ключом и помещает в открытый доступ. Пара (N,d) является секретным ключом. Для расшифрования достаточно знать секретный ключ. Числа p, q, $\varphi(N)$ в дальнейшем не нужны, поэтому их можно уничтожить.

Пользователь A, отправляющий сообщение x абоненту B, выбирает из открытого каталога пару (N,e) абонента B и вычисляет шифрованное сообщение $y = x^e (\mathrm{mod}\, N)$. Чтобы получить исходный текст, абонент B вычисляет $y^d (\mathrm{mod}\, N)$. Так как $e \cdot d \equiv 1(\mathrm{mod}\,\varphi(N))$, т.е. $e \cdot d = \varphi(N) \cdot k + 1$, k — целое, то применяя теорему Эйлера получим: $y^d \equiv (x^e)^d \equiv x^{ed} \equiv x^{\varphi(N)\cdot k+1} \equiv (x^{\varphi(N)})^k \cdot x \equiv x(\mathrm{mod}\, N)$.

Пример 1. Пусть p = 7, q = 17. Тогда N = 7·17 = 119, $\varphi(N) = 96$. Выбираем e такое, что: $e < 96, (e,96) = 1$. Пусть в нашем случае e = 5. Находим d: $d = 1/e(\mathrm{mod}\, 96)$. Получаем d = 77, т.к. 77·5 = 4·96+1. Открытый ключ: (119,5). Личный ключ: (119,77). Пусть, например, X = 19. Для зашифрования число 19 возводим в степень 5 по модулю 119. Имеем: 19^5 = 2476099, и остаток от деления 2476099 на 119 равен 66. Итак, y = 19^5 mod 119 = 66. Расшифрование: x = 66^7 mod 119 = 19.

О вычислениях

Как шифрование, так и расшифрование в RSA предполагают использовании операции возведения целого числа в целую степень по модулю N. Если возведение в степень выполнять непосредственно с целыми числами и только потом проводить сравнение по модулю N, то промежуточные значения окажутся огромными. К счастью, здесь можно воспользоваться свойствами арифметики в классах вычетов: $(a \bmod N)\cdot(b \bmod N) \bmod N = (ab) \bmod N$. Таким образом, мы можем

рассматривать промежуточные результаты по модулю *N*. Это делает вычисления практически выполнимыми.

О стойкости RSA

Безопасность алгоритма RSA основана на трудоемкости разложения на множители больших чисел. Современное состояние технических средств разложения на множители таково, что число, содержащее 193 десятичных знака, факторизовано в 2005 году. Следовательно, выбираемое *N* должно быть больше. Большинство общепринятых алгоритмов вычисления простых чисел p и q носят вероятностный характер.

О выборе чисел p и q.

Для работы алгоритма RSA нужны простые числа. Наиболее приемлемым является генерация случайных чисел и последующая проверка их на простоту. Существуют вероятностные тесты, определяющие с заданной степенью достоверности факт простоты числа. Возникает вопрос: что произойдет, если числа окажутся составными? Можно свести вероятность такого события до приемлемого минимума, используя тесты на простоту. Кроме того, если такое событие произойдет, это будет быстро обнаружено — шифрование и расшифрование не будут работать.

Кроме разрядности p и q, к ним предъявляются следующие дополнительные требования:

1. Числа не должны содержаться в списках известных больших простых чисел.
2. Они не должны быть близкими, так как иначе можно воспользоваться для факторизации *N* методом Ферма и решить уравнение $(\frac{p+q}{2})^2 - N = (\frac{p-q}{2})^2$.
3. В алгоритме RSA всегда есть эквивалентные по расшифрованию показатели степеней, например d и $d' = d + [p-1, q-1]$. При этом эквивалентных решений тем больше, чем больше (p-1,q-1). В лучшем случае (p-1,q-1) = 2, $p = 2t+1$, $q = 2s+1$, где s,t — нечетные числа с условием (s,t) = 1.

Чтобы исключить возможность применения методов факторизации накладывают следующее ограничение. Числа p-1, p+1, q-1, q+1 не должны разлагаться в произведение маленьких простых множителей, должны содержать в качестве сомножителя хотя бы одно большое простое число. В 1978 г. Райвест сформулировал наиболее сильные требования. Числа $p_1 = \frac{p-1}{2}$, $p_2 = \frac{p+1}{2}$, $q_1 = \frac{q-1}{2}$, $q_2 = \frac{q+1}{2}$ должны быть простыми, причем p_1-1 и q_1-1не должны разлагаться в произведение маленьких простых.

О выборе параметров e и d

Рассмотрим теперь вопрос о выборе экспонент шифрования и расшифрования. Так как значения e и d определяют время зашифрования и

расшифрования, то можно назвать ряд ситуаций, в которых желательно иметь малое значение e и d. Например, при использовании системы RSA при защите электронных платежей с применением кредитных карточек естественным является требование использования небольших значений экспоненты d у владельца карточки и большого значения экспоненты e у центрального компьютера.

Однако, выбор малых параметров e или d представляется небезопасным по ряду соображений. Если малым является секретный параметр d, то можно применить метод перебора малых значений до получения искомого числа d.

Если малым является параметр e, то достаточно большое число открытых сообщений, удовлетворяющих неравенству $x < \sqrt[e]{N}$ будут зашифровываться простым возведением в степень $y = x^e \pmod N$ и поэтому их можно найти путём извлечения корня степени e.

Другая аналогичная ситуация может сложиться, когда у нескольких абонентов используется одинаковая экспонента e. Тогда становится возможна атака на основе китайской теореме об остатках (см. ниже).

Подготовка текста к шифрованию

Сначала мы должны каким-либо способом представить текст сообщения в виде упорядоченного набора чисел по модулю *N*. Это еще не процесс шифрования, а только подготовка к нему.

Пример 2. Для простоты предположим, что текст сообщения содержит только слова, записанные только заглавными буквами. Первый шаг состоит в замене каждой буквы сообщения числом. Пусть наша таблица замен имеет вид:

Табл. 14. Таблица замен

А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У
10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29

Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
31	32	33	34	35	36	37	38	39	40	41

Пробел между словами будем заменять числом 99.

Например, пусть открытый текст — это девиз «ПОЗНАЙ СЕБЯ». Тогда его цифровое представление имеет вид:

252417231019992715141

Пусть в нашем примере p = 149, q = 157, тогда N = 23393. Поэтому цифровое представление открытого текста нужно разбить на блоки, меньшие, чем 23393. Одно из таких разбиений:

2524 – 1723 – 10199 – 9271 – 511 – 41

Конечно, выбор блоков неоднозначен, но и не совсем произволен. Например, во избежание двусмысленностей на стадии расшифровки не следует выделять блоки, начинающиеся с нуля.

При расшифровке сообщения получают последовательность блоков, затем

их соединяют вместе и получают число. После этого числа заменяют буквами в соответствии с таблицей.

Обратите внимание на то, что в этом примере мы каждую букву кодируем двузначным числом. Это сделано для предотвращения неоднозначности. Если бы мы пронумеровали буквы не по порядку, начиная с 1, т.е. А соответствует 1, Б — 2 и.т.д., то мы не смогли бы сказать, блок 12 обозначает пару букв АБ или букву Л, двенадцатую букву алфавита. Конечно, для кодирования можно использовать любые однозначные соответствия между буквами и числами, например, ASCII–кодировку, что чаще всего и делается.

Продолжим наш пример. Мы выбираем p = 149, q = 157, вычисляем $\varphi(N) = 23088$. Теперь нужно выбрать число e, взаимно простое с $\varphi(N)$. Наименьшее простое, не делящее $\varphi(N)$, равно 5. Положим e = 5. Зашифруем первый блок сообщения. Вычисляем $2524^5 \bmod 23393 = 22752$. Далее, $1723^5 \bmod 23393 = 6198$

$10199^5 \bmod 23393 = 14204$,
$9271^5 \bmod 23393 = 23191$,
$511^5 \bmod 23393 = 10723$,
$41^5 \bmod 23393 = 14065$.

Теперь шифрованный текст имеет вид:

22752619814204231911072314065

В нашем примере N = 23393, e = 5. Применив алгоритм Эвклида к числам $\varphi(N) = 23088$ и e = 5, найдем $d = e^{-1} \bmod 23088 = 13853$. Значит, для расшифровки блоков шифртекста мы должны возвести этот блок в степень 13583 по модулю 23393. В примере первый блок шифртекста — число 22752. Вычисляем: $22752^{13853} \bmod 23393 = 2524$.

Разбиение числа на блоки можно произвести различными способами. При этом промежуточные результаты зависят от способа разбиения, однако конечный результат — не зависит.

Атаки на RSA

Для дешифрации необходимо по известным N, e и шифртексту y найти такое $x \in ((Z/N))^*$, что $y = x^e \bmod N$.

Можно попытаться решить сравнение при конкретных y, затем использовать гомоморфность отображения D(x).

Один из возможных способов следующий. Пусть имеется набор пар $\{(x_1, y_1)...(x_k, y_k)\}$ с условием $x_i^e = y_i \bmod N$. Пусть 1<y<N, (y,N) = 1. Если каким-либо образом удалось представить y в виде: $y = y_1^{s_1} \cdot ... \cdot y_k^{s_k} \bmod N$ с целыми s_k, то $x = x_1^{s_1} \cdot ... \cdot x_k^{s_k}$ будет решением сравнения $y = x^e \bmod N$.

Пример 3. У нас есть в наличии открытый ключ N = 31459, e = 5 и набор пар соответствующих друг другу исходных и зашифрованных сообщений: (23,18707), (755,26871), (631,6384). Требуется расшифровать шифртекст y = 11638. Для этого представим y в виде $y = 18707^{-1} \cdot 26871^3 \cdot 6384^{-2} = 11638$. Отсюда легко вычислить исходное сообщение: $x = 23^{-1} \cdot 755^3 \cdot 631^{-2} = 28260$.

Однако заметим, что этот подход не менее труден, чем поиск алгоритма решения сравнения $y = x^e \bmod N$.

Взлом RSA при неудачном выборе параметров криптосистемы

Само по себе использование RSA не обеспечивает безопасности. Дело ещё в деталях реализации. Приведём ряд примеров. Для простоты вычислений будем работать с небольшими числами. Наша цель — показать особенности, не зависящие от размера.

Пример 4. Пусть пользователь выбрал N = 2047, e = 179, d = 411. Так как 2047 = 23·89, а $\varphi(23) = 22, \varphi(89) = 88$ имеют наименьшее общее кратное 88, то любой обратный к 179 по модулю 88, например, 59, будет действовать как d.

Пример 5. Число N = 536813567 является произведением простого числа Мерсенна 8191 и простого числа Ферма 65537. Это очень плохой выбор.

Пример 6. Число 23360947609 является очень плохим выбором для N из-за того, что два его простых делителя слишком близки к друг другу. Действительно, пусть p>q, имеем: $N = (\frac{p+q}{2})^2 + (\frac{p-q}{2})^2$. Обозначим: $t = \frac{p+q}{2}, S = \frac{p-q}{2}$. Так как S мало, то t — целое число, лишь немного большее $\sqrt{N}$, причем $t^2 - N$ является полным квадратом. Проверяем подряд целые числа $t > \sqrt{N}$. В нашем примере: t_1 = 152843, t_2 = 152844, t_3 = 152845, и $t^3 - N = 804^2$, тогда $p = 152845 + 804, p = 152845 - 804$. Таким образом мы с третьей попытки нашли p и q. Количество попыток, необходимых для факторизации N, можно при известных p и q вычислить по следующей формуле: $k = \sqrt{p \cdot q + (\frac{p-q}{2})^2} - \left[\sqrt{p \cdot q}\right]$, где [x] — операция округления x до ближайшего целого числа.

Атака повторным шифрованием

Строим последовательность: $y_1 = y,\ y_i = y_{i-1}^e (\bmod\ N), i > 1$. Итак, $y_m = y^{e^m} (\bmod N)$, а так как $(e, \varphi(N)) = 1$, то существует такое натуральное число m, что $e^m \equiv 1 (\bmod \varphi(N))$. Но тогда $y^{e^m - 1} \equiv 1 (\bmod N)$, отсюда следует: $y^{e^m} \equiv y (\bmod N)$, значит, y_{m-1} — решение сравнения $y = x^e (\bmod N)$.

Пример 7. Пусть у нас имеется открытый ключ N =84517 , e = 397 и зашифрованное им сообщение y = 8646. Необходимо найти исходный текст x. Возведем *y* в степень *e* и получим y_2 = 37043. Будем повторять операцию до тех пор, пока не получим y_n = y. y_{n-1} — искомое сообщение: y_3 = 5569, y_4 = 61833, y_5 = 83891, y_6 = 16137, y_7 = 8646. y_6 является решением сравнения $y = x^e (\bmod N)$, а, следовательно, искомым сообщением x.

Замечание. Анализ метода повторного шифрования хорошо показывает необходимость соблюдения требований на выбор p и q для обеспечения стойкости. В данном примере d = 82225. Неудачный выбор криптосистемы привел к тому, что атака методом повторного шифрования дала результат

почти сразу, тогда как нахождение d потребовало бы на порядок больших вычислений.

Атака на основе Китайской теоремы об остатках.

Как отмечалось ранее, системы шифрования с открытыми ключами работают сравнительно медленно. Для повышения скорости шифрования RSA на практике используют малую экспоненту зашифрования.

Если выбрать число e небольшим или таким, чтобы в его двоичной записи было мало единиц, то процедуру шифрования можно значительно ускорить. Например, выбрав e - 3 (при этом ни p — 1, ни q — 1 не должны делиться на 3), мы сможем реализовать шифрование с помощью одного возведения в квадрат по модулю N и одного перемножения. Выбрав $e = 2^{16} - 1 =$ 65537 — число, двоичная запись которого содержит только две 1, мы сможем реализовать шифрование с помощью 16 возведений в квадрат по модулю N и одного перемножения. Если экспонента e выбирается случайно, то реализация шифрования по алгоритму RSA потребует s возведений в квадрат по модулю N и в среднем s/2 умножений по тому же модулю, где 5 длина двоичной записи числа N . Вместе с тем выбор небольшой экспоненты e может привести к негативным последствиям. Дело в том, что у нескольких корреспондентов могут оказаться одинаковые экспоненты e.

Пусть, например, три корреспондента имеют попарно взаимно простые модули N_1, N_2, N_3 и общую экспоненту e = 3. Если еще один пользователь посылает им некое циркулярное сообщение x, то криптоаналитик противника может получить в свое распоряжение три шифрованных текста $y_i = x^3 \pmod{N_i},\ i = 1, 2, 3$. Далее он может найти решение системы сравнений

$$\begin{cases} y \equiv y_1 \pmod{N_1}, \\ y \equiv y_2 \pmod{N_2}, \\ y \equiv y_3 \pmod{N_3}, \end{cases}$$

лежащее в интервале $0 < y < N_1 \cdot N_2 \cdot N_3$. По китайской теореме об остатках такое решение единственно, а так как $x^3 < N1, N2, N3$, то $y = x^3$. Само x можно найти, вычисляя кубический корень: $x = \sqrt[3]{y}$.

Отметим, что выбор малой экспоненты расшифрования d также нежелателен в связи с возможностью определения d простым перебором. Известно также что если $d < \sqrt[4]{N}$, то экспоненту d легко найти, используя непрерывные дроби.

Пример 8. Три пользователя имеют модули $N_1 = 26549$, $N_2 = 45901$, $N_3 = 25351$. Все пользователи используют экспоненту e = 3. Всем пользователям было послано некое сообщение x, причем, пользователи получили сообщения $y_1 = 5366$, $y_2 = 814$, $y_3 = 4454$. Найдем $M_0 = N_1 \cdot N_2 \cdot N_3 = 30893378827799$. Далее находим

$m_1 = N_2 \cdot N_3 = 1163636251$
$m_2 = N_1 \cdot N_3 = 673043699$
$m_3 = N_1 \cdot N_2 = 1218625649$

$n_1 = m_1^{-1} \bmod N_1 = 13533$
$n_2 = m_2^{-1} \bmod N_2 = 27930$
$n_3 = m_3^{-1} \bmod N_3 = 22354$

$S = y_1 \cdot n_1 \cdot m_1 + y_2 \cdot n_2 \cdot m_2 + y_3 \cdot n_3 \cdot m_3 = 84501028038745578 + 15301661957638980 + 121332116653000684 = 221134806649385242$

$S \bmod M_0 = 1000000000$

$x = (S \bmod M_0)^{1/3} = 1000$ — исходное сообщение, отправленное пользователям.

Бесключевое чтение

Пусть два пользователя выбрали одинаковый модуль N и разные экспоненты e1 и e_2. Если один пользователь посылает им некое циркулярное сообщение x, то криптоаналитик противника может получить в свое распоряжение два шифрованных текста $y_1 = x^{e_1} (\bmod N)$ и $y_2 = x^{e_2} (\bmod N)$. В таком случае криптоаналитик может получить исходное сообщение, используя расширенный алгоритм Евклида, находим r, s такие, что $re_1 + se_2 = 1$. Отсюда получаем: $y_1^r y_2^s = x^{re_1 + se_2} = x$

Пример 9. Два пользователя используют общий модуль N = 137759, но разные взаимно простые экспоненты $e_1 = 191$ и $e_2 = 233$. Пользователи получили шифртексты $y_1 = 60197$ и $y_2 = 63656$, которые содержат одно и то же сообщение. Найдем исходное сообщение методом бесключевого чтения. Т.к. e_1 и e_2 взаимно просты, найдем такие r и s, что $re_1 + se_2 = 1$. С помощью расширенного алгоритма Евклида находим r = 61, s = -50. Искомое сообщение $x = y_1^r \cdot y_2^s = 60197^{61} \cdot 63656^{-50} = 1234$

Выводы

Как видно из приведенных выше примеров (а также из примеров выполнения заданий лабораторных работ) выбор параметров криптосистемы является ответственной задачей. Параметры необходимо выбирать в строгом соответствии с требованиями. Существующими в настоящими время методами (и при использовании существующих в настоящее время вычислительных мощностей) атака на алгоритм и/или криптосистему возможна лишь при неудачном выборе параметров. В процессе выполнения заданий лабораторных работ вы убедитесь в обоснованности перечисленных требований к параметрам криптосистемы. В частности, необходимо обеспечить каждому пользователю уникальные значения p, q и уникальное значение e, удовлетворяющие требованиям.

Для тренировки в использовании алгоритма авторы рекомендуют пользоваться учебно-методическим изданием [21].

2.7.3. Алгоритм Pohlig-Hellman

Схема шифрования Pohlig-Hellman похожа на RSA. Это не симметричный алгоритм, так как для шифрования и дешифрирования используются различные ключи. Это не схема с открытым ключом, потому что ключи легко получаются один из другого, и ключ шифрования, и ключ дешифрирования должны храниться в секрете. Как и в RSA,

$C = P^e \bmod n$

$P = C^d \bmod n$

где

$ed \equiv 1$ (mod какое-нибудь составное число)

В отличие от RSA n не определяется с помощью двух простых чисел и остается частью закрытого ключа. Если у кого-нибудь есть e и n, он может вычислить d. Не зная e или d, противник будет вынужден вычислить

$e = \log_p C \bmod n$, а это является трудной проблемой.

2.7.4. Алгоритм Rabin

Безопасность схемы Рабина (Rabin) опирается на сложность поиска квадратных корней по модулю составного числа. Эта проблема аналогична разложению на множители. Вот одна из реализаций этой схемы.

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

Для шифрования сообщения M (M должно быть меньше n), просто вычисляется $C = M^2 \bmod n$

Дешифрирование сообщения также несложно, но немного скучнее. Так как получатель знает p и q, он может решить две конгруэнтности с помощью китайской теоремы об остатках . Вычисляется

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

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

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

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

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 и M_4, равно M. Если сообщение написано по английски, выбрать правильное M, нетрудно. С другой стороны, если сообщение является потоком случайных битов (скажем, для генерации ключей или цифровой подписи), способа определить, какое M_i - правильное, нет. Одним из способов решить эту проблему служит добавление к сообщению перед шифрованием известного заголовка.

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

$p = 3 \bmod 8$

$q = 7 \bmod 8$

и

$N = pq$

Кроме того, используется небольшое целое число, *S,* для которого *J(S,N)* = -1. (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 $J(M,N) = (-1)^{c1}$. Затем вычисляется $M = (S^{c1} * M) \bmod N$. 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 С, получатель вычисляет *M″* с помощью

$C^k = \pm M'' \pmod N$

Правильный знак *M″* определяет c_2. Наконец

$M = (S^{cl} * (-1)^{c1} * M'') \bmod N$

Впоследствии Вильямс улучшил эту схему. Вместо возведения в квадрат открытого текста сообщения, возведите его в третью степени. Большие простые числа должны быть конгруэнтны 1 по модулю 3, иначе открытый и закрытый ключи окажутся одинаковыми. Даже лучше, существует только одна уникальная расшифровка каждого шифрования.

Преимущество схем Рабина и Вильямса перед RSA в том, что доказано, что они также безопасны, как и разложение на множители. Однако перед вскрытием с выбранным шифротекстом они совершенно беззащитны. Если вы собираетесь использовать эти схемы для случаев, когда взломщик может выполнить такое вскрытие (например, алгоритм цифровой подписи, когда взломщик может выбирать подписываемые сообщения), не забывайте использовать перед подписанием однонаправленную хэш-функцию. Рабин предложил другой способ защититься от такого вскрытия: к каждому сообщению перед хэшированием и подписанием добавляется уникальная случайная строка. К несчастью, после добавления однонаправленной хэш-функцией тот факт, что система столь же безопасна, как и разложение на множители, больше не является доказанным. Хотя с практической точки зрения добавление хэширования не может ослабить систему.

2.7.5. Алгоритм EIGamal

Схему EIGamal можно использовать как для цифровых подписей, так и для шифрования, его безопасность основана на трудности вычисления дискретных логарифмов в конечном поле.

Для генерации пары ключей сначала выбирается простое число *p* и два случайных числа, *g* и *x,* оба эти числа должны быть меньше *p.* Затем вычисляется

$y = g^x \bmod p$

Открытым ключом являются *y, g* и *p. И g,* и *p* можно сделать общими для группы пользователей. Закрытым ключом является *x.*

Шифрование EIGamal

Модификация EIGamal позволяет шифровать сообщения. Для шифрования сообщения *M* сначала выбирается случайное число *k,* взаимно простое с *p* - 1. Затем вычисляются

$a = g^k \bmod p$ $b = y^k M \bmod p$

Пара *(a,b)* является шифротекстом. Обратите внимание, что шифротекст в два раза длиннее открытого текста. Для дешифрирования *(a,b)* вычисляется

$M = b/a^x \bmod p$

Так как $a^x = g^{kx} \pmod p$ и $b/a^x = y^k M/a^x = g^{xk} M/g^{kx} = M \pmod p$, то все работает . По сути это то же самое, что и обмен ключами Диффи-Хеллмана за исключением того, что *y* - это часть ключа, а при шифровании сообщение умножается на y^k.

Табл. 15 Шифрование EIGamal

Открытый ключ:	
p	простое число (может быть общим для группы пользователей)
g	$<p$ (может быть общим для группы пользователей)
y	$=g^x \bmod p$
Закрытый ключ:	
x	$<p$
Шифрование:	
k	выбирается случайным образом, взаимно простое с *p*-1
a	(шифротекст) $=g^k \bmod p$
b	(шифротекст)= $y^k M \bmod p$
Дешифрирование:	
M (открытый текст) = $b/a^x \bmod p$	

Табл. 16 Скорости EIGamal для различных длин модулей при 160-битовом показателе степени (на SPARC II)

	512 битов	768 битов	1024 битов
Шифрование	0.33с	0.80 с	1.09 с
Дешифрирование	0.24 с	0.58 с	0.77 с
Подпись	0.25 с	0.47 с	0.63 с
Проверка	1.37 с	5.12 с	9.30 с

2.7.6. Алгоритм McELIECE

В 1978 году Роберт МакЭлис (Robert McEliece) разработал криптосистему с открытыми ключами на основе теории алгебраического

кодирования. Этот алгоритм использует существование определенного класса исправляющих ошибки кодов, называемых **кодами Гоппа** (Goppa). Он предлагал создать код Гоппа и замаскировать его как обычный линейный код. Существует быстрый алгоритм декодирования кодов Гоппа, но общая проблема найти слово кода по данному весу в линейном двоичном коде является **NP-полной.** Ниже приведен только краткий обзор.

Пусть $d_H(x,y)$ обозначает расстояние Хэмминга между x и y. Числа n, k и t служат параметрами системы.

Закрытый ключ состоит из трех частей: G' - это матрица генерации года Гоппа, исправляющего t ошибок. P -это матрица перестановок размером $n*n$. S - это nonsingular матрица размером $k*k$.

Открытым ключом служит матрица G размером k*n: $G = SG'P$.

Открытый текст сообщений представляет собой строку k битов в виде k-элементного вектора над полем GF(2).

Для шифрования сообщения случайным образом выбирается n-элементный вектор z над полем GF(2), для которого расстояние Хэмминга меньше или равно t.

$c=mG+z$

Для дешифрирования сообщения сначала вычисляется $c'= cP^{-1}$. Затем с помощью декодирующего алгоритма для кодов Гоппа находится m', для которого $d_H(m'G,c)$ меньше или равно t. Наконец вычисляется $m = m'S^{-1}$.

В своей оригинальной работе МакЭлис предложил значения n = 1024, t = 50 и k = 524. Это минимальные значения, требуемые для безопасности.

Хотя этот алгоритм был одним из первых алгоритмов с открытыми ключами, и вне появлялось публикаций о его успешном криптоаналитическом вскрытии, он не получил широкого признания в криптографическом соообществе. Схема на два-три порядка быстрее, чем RSA, но у нее есть ряд недостатков. Открытый ключ огромен: 2^{19} битов. Сильно увеличивается объем данных - шифротекст в два раза длиннее открытого текста. Ни одна из них не достигла успеха для общего случая, хотя сходство между алгоритмом МакЭлиса и алгоритмом рюкзака немного волнует.

В 1991 два русских криптографа заявили, что взломали систему МакЭлиса с некоторыми параметрами. В их статье это утверждение не было обосновано, и большинство криптографов не приняли во внимание этот результат.

Другие алгоритмы, основанные на линейных кодах, исправляющих ошибки

Алгоритм Нидеррейтера (Niederreiter) очень близок к алгоритму МакЭлиса и считает, что открытый ключ - это случайная матрица проверки четности кода, исправляющего ошибки. Закрытым ключом служит эффективный алгоритм декодирования этой матрицы.

Другой алгоритм, используемый для идентификации и цифровых подписей, основан на декодировании синдрома, использующий коды,

исправляющие ошибки, небезопасен.

2.7.7. Алгоритм LUC

Некоторые криптографы разработали обобщенные модификации RSA, которые используют различные перестановочные многочлены вместо возведения в степень. Вариант, называющийся Kravitz-Reed и использующий неприводимые двоичные многочлены, небезопасен. Винфрид Мюллер (Winfried Muller) и Вилфрид Нобауер (Wilfried Nobauer) используют полиномы Диксона (Dickson). Рудольф Лидл (Rudolph Lidl) и Мюллер обобщили этот подход в (этот вариант назван схемой Reidi), и Нобауер проанализировал его безопасность. Несмотря на все предыдущие разработки группе исследователей из Новой Зеландии удалось запатентовать эту схему в 1993 году, назвав ее LUC.

n-ое число Лукаса, $V_n(P,1)$, определяется как $V_n(P,1) = PV_{n-1}(P,1)-V_{n-2}(P,1)$

В любом случае для генерации пары открытый ключ/закрытый ключ сначала выбираются два больших числа p и q. Вычисляется *n,* произведение p и q. Ключ шифрования *e* - это случайное число, взаимно простое с *p-1, q-1,p+1* и q+*1*. Существует четыре возможных ключа дешифрирования,

$d=e^{-1}$ mod (НОК(p+l), *(q+1)))*

$d=e^{-1}$ mod (НОК(p+l), *(q-1)))*

$d=e^{-1}$ mod (НОК(p-l), *(q+1)))*

$d=e^{-1}$ mod (НОК(p-l), *(q-1)))*

где НОК означает наименьшее общее кратное.

Открытым ключом являются *d* и и; закрытым ключом - *e* и *n. p* и *q* отбрасываются.

Для шифрования сообщения *P* (*P* должно быть меньше и) вычисляется

$C = V_e(P,l)\ (mod\ n)$

А для дешифрирования:

$P = V_d(P, 1)$ (mod n), с соответствующим *d*

В лучшем случае LUC не безопаснее RSA. А недавние опубликованные результаты показывают, как взломать LUC по крайней мере в нескольких реализациях.

2.7.8. Криптосистемы на базе конечных автоматов

Китайский криптограф Тао Ренжи разработал алгоритм с открытым ключом, основанный на использовании конечных автоматов. Такой же сложной задачей, как и разложение на множители произведения двух больших простых чисел, является задача разложения на составляющие произведения двух конечных автоматов. Это тем более верно, если один из автоматов нелинеен.

Большая часть работы в этой области была выполнена в Китае в 80-х годах и опубликована на китайском языке. Ренжи начал писать по-английски. Его главным результатом было то, что обратное значение некоторых нелинейных (квазилинейных) автоматов является слабым тогда и только тогда, когда эти автоматы обладают определенной ступенчатой матричной структурой. Это свойство исчезает, если они объединены с другим автоматом (хотя бы линейным). В алгоритме с открытым ключом секретный ключ является инвертируемым квазилинейным автоматом, а соответствующий открытый ключ может быть получен с помощью их почленного перемножения. Данные шифруются, проходя через линейный автомат, а дешифрируются, проходя через обратные значения компонентов алгоритма (в некоторых случаях автоматы должны быть установлены в подходящее начальное значение). Эта схема работает и для шифрования, и для цифровых подписей.

О производительности таких систем вкратце можно сказать следующее: они, как и система McEliece, намного быстрее RSA, но требуют использования более длинных ключей. Длина ключа, обеспечивающая, как думают, безопасность, аналогичную 512-битовому RSA, равна 2792 битам, а 1024-битовому RSA - 4152 битам. В первом случае система шифрует данные со скоростью 20869 байт/с и дешифрирует данные со скоростью 17117 байт/с, работая на 80486/33 МГц.

Ренжи опубликовал три алгоритма. Первым был FAPKC0. Эта слабая система использует линейные компоненты и, главным образом, является иллюстративной. Каждая из двух серьезных систем, FAPKC1 и FAPKC2, использует один линейный и один нелинейный компонент. Последняя сложнее, она была разработана для поддержки операции проверки подлинности.

2.8. КРИПТОСИСТЕМЫ НА ЭЛЛИПТИЧЕСКИХ КРИВЫХ

В последние два десятилетия все большее применение в криптографии находит одна из областей теории чисел и алгебраической геометрии — теория эллиптических кривых над конечными полями. Основная причина этого состоит в том, что эллиптические кривые над конечными полями доставляют неисчерпаемый источник конечных абелевых групп, которые (даже если они велики) удобны для вычислений и обладают богатой структурой. Во многих отношениях эллиптические кривые — естественный аналог мультипликативных групп полей групп, но более удобный, так как существует большая свобода в выборе эллиптической кривой, чем в выборе конечного поля.

Начнем с изложения основных определений и свойств эллиптических кривых. Мы ограничимся минимальным числом основных фактов, необходимых для понимания приложений к криптографии , уделяя больше внимания примерам и конкретным описаниям и меньше заботясь о доказательствах и общности. Более систематическое изложение этих вопрсов можно найти в литературе,см. список.

2.8.1. Эллиптические кривые

В этом параграфе мы предполагаем, что K - поле: либо поле R вещественных чисел, либо поле **Q** рациональных чисел, либо поле C комплексных чисел, либо поле E_q из $q = p^r$ элементов.

Определение: Пусть К — поле характеристики, отличной от 2, 3. и $x^3 + ax + b$ (где a, b$\in K$) — кубический многочлен без кратных корней. *Эллиптическая кривая над К* — это множество точек (x, y) x, y$\in K$, удовлетворявших уравнению

$$y^2 = x^3 + ax + b \quad (1)$$

вместе с единственным элементом, обозначаемым О и называемым «точка в бесконечности» (о ней подробнее будет сказано ниже).

Если K — поле характеристики 2, то *эллиптическая кривая над К* — это множество точек, удовлетворяющих уравнению либо типа

$$y^2 + cy = x^3 + ax + b, \quad (2а)$$

либо типа

$$y^2 + xy = x^3 + ax^2 + b \quad (2б)$$

(здесь кубические многочлены в правых частях могут иметь кратные корни), вместе с «точкой в бесконечности» *O*.

Если *K* — поле характеристики 3, то *эллиптическая кривая над K* — это множество точек, удовлетворяющих уравнению

$$y^2 = x^3 + ax^2 + bx + c \quad (3)$$

(где кубический многочлен справа не имеет кратных корней), вместе с «точкой в бесконечности» *O*.

Замечания.

1. Имеется общая форма уравнения эллиптической кривой, которая применима при любом поле: $y^2 + a_1xy + a_3y = x^2 + a_2x^2 + a_4x + a_6$: в случае, когда char $K \neq 2$, ее можно привести к виду $y^2 = x^3 + ax^2 + bx + c$ (или к виду $y^2 = x^3 + bx + c$. если $K > 3$). В случае, когда поле *K* имеет характеристику 2, это уравнение преобразуется либо к виду (2а), либо к виду (2*б*).

2. Если F(x, y) = O — неявное уравнение, выражающее *y* как функцию *x* в (1) (или в (2), (3)),

$$\text{т. е. } F(x, y) = y^2 - x^3 - ax - b$$

(или $F(x, y) = y^2 + cy + x^3 - ax + b$, $y^2 + xy + x^3 + ax + b$, $y^2 - x^3 - ax^2 - bx - c$), то точка (x, y) этой *кривой* называется *неособенной* (или *гладкой*) точкой, если, по крайней мере, одна из частных производных $\partial F / \partial x, \partial F / \partial y$ в этой точке не равна нулю. (Производные многочленов можно определить обычными формулами над любым полем.)

Нетрудно показать, что условие отсутствия кратных корней у кубических многочленов в правой части в (1) и (3) эквивалентно требованию, чтобы все точки кривой были неособенными.

Эллиптические кривые над вещественными числами. Перед обсуждением конкретных примеров эллиптических кривых дат. разными полями мы отметим чрезвычайно важное свойство множества точек эллиптической кривой: они образуют абелеву группу. Чтобы объяснить наглядно, как это получается, мы временно будем полагать, что *K* = *R*. т.е. что эллиптическая кривая — обычная плоская кривая (с добавлением еще одной точки *O* «в бесконечности»).

Определение.

Пусть *E* — эллиптическая кривая над вещественными числами, и пусть *P* и *Q* - две точки на *E*. Определим точки -*P* и *P*+*Q* по следующим правилам.

1.Точка *O* - тождественный элемент по сложению («нулевой элемент»)

группы точек. В следующих пунктах предполагается, что ни P, ни Q не являются точками в бесконечности.

2. Точки P= (x, y) и $-P$ имеют одинаковые x - координаты, а их y - координаты различаются только знаком, т.е. -(x. y) = (x.- y). Из (1) сразу следует, что (x, -y) — также точка на E.

3. Если P и Q имеют различные x - координаты, то прямая $l = \overline{PQ}$ имеет с E еще в точности одну точку пересечения R (за исключением двух случаев: когда она оказывается касательной в P, и мы тогда полагаем $R = P$, или касательной в Q. и мы тогда полагаем $R = Q$). Определяем теперь $P + Q$ как точку $-R$, т.е. как отражение от оси x третьей точки пересечения. Геометрическое построение, дающее P + Q, приводится ниже в примере 1.

4. Если $Q = -P$ (т. е. x - координата Q та же, что и y P, а y - координата отличается лишь знаком), то полагаем P + Q = O (точке в бесконечности; это является следствием правила 1).

5. Остается возможность P = Q. Тогда считаем, что l — касательная к кривой в точке P. Пусть R — единственная другая точка пересечения l с E. Полагаем $P + Q = -R$ (в качестве R берем P, если касательная прямая в P имеет «двойное касание», т. е. если P есть точка перегиба кривой).

Пример 1. На (рис.1) , изображены эллиптическая кривая $y^2 = x^3 - x$ в плоскости xy и типичный случай сложения точек P и Q. Чтобы найти $P + O$. проводим прямую $\overline{PQ}$ и в качестве $P+Q$ берем точку, симметричную относительно оси x третьей точке, определяемой пересечением прямой PQ и кривой. Если бы P совпадала с Q, т.е. если бы нам нужно было найти $2P$, мы использовали бы касательную к кривой в P: тогда точка $2P$ симметрична третьей точке, в которой эта касательная пересекает кривую.

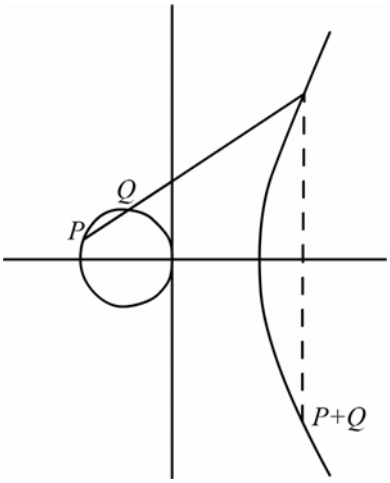


рис.1

Теперь мы покажем, почему существует в точности еще одна точка, где прямаяд *l*, проходящая через *P* и *Q*, пересекает кривую; заодно мы выведем формулу для координат этой третьей точки и тем самым — для координат *P*+*Q*.

Пусть (x_1, y_1), (x_2, y_2) и (x_3, y_3) обозначают координаты соответственно *P*, *Q* и *P*+*Q*. Мы хотим выразить x_3, y_3 через x_1, y_1, x_2, y_2.

Предположим, что мы находимся в ситуации п.3 определения *P*-*Q*, и пусть $y = ax + \beta$ есть уравнение прямой, проходящей через *P* и *Q* в этой ситуации она не вертикальна). Тогда $a = (y_2 - y_1)/(x_2 - x_1)$ и $\beta = y_1 - ax_1$. Точка на *l*, т. е. точка $(x, ax + \beta)$, лежит на эллиптической кривой тогда и только тогда, когда $(ax + \beta)^2 = x^3 + ax + b$. Таким образом, каждому корню кубического многочлена $x^3 - (ax + \beta)^2 + ax + b$ ответствует точка пересечения. Мы уже знаем, что имеется два корня x_1 и x_2. так как $(x_1, ax_1 + \beta)$,$(x_2, ax_2 + \beta)$ — точки *P*, *Q* на кривой. Так как сумма корней нормированного многочлена равна взятому с обратным знаком коэффициенту при второй по старшинству степени многочлена, то в нашем случае третий корень — это $x^3 = a^2 + x_1 + x_2$. Тем самым получаем выражение для x_3, и, следовательно, P+*Q* = $(x_3 - (ax_3 + \beta))$ или, в терминах x_1, y_1, x_2, y_2,

$$x_3 = (\frac{y_2 - y_1}{x_2 - x_1})^2 - x_1 - x_2, \qquad (4)$$

$$y_3 = -y_1 + \frac{y_2 - y_1}{x_2 - x_1}(x_1 - x_3).$$

Ситуация в п.5 аналогична, только теперь *a* — производная d*y*/ d*x* в *P*.

Дифференцирование неявной функции, заданной уравнением (1), приводит к формуле $a = (3x_1^2 + a)/(2y_1)$, и мы получаем следующие формулы для координат удвоенной точки :

$$x_3 = (\frac{3x_1^2 + a}{2y_1})^2 - 2x_1, \qquad (5)$$

$$y_3 = -y_1 + \frac{3x_1^2 + a}{2y_1}(x_1 - x_3).$$

Пример 2. На эллиптической кривой $y^2 = x^3 - 36x$ пусть P = (-3,9) и Q = (-2,8). Найти $P + O$ и $2P$.

Решение. Подстановка $x_1 = -3, y_1 = 9, x_1 = -2, y_1 = 8$ в первое из уравнений (4) дает $x_3 = 6$; тогда второе из уравнений (4) дает $y_3 = 0$. Далее, подставляя $x_1 = -3, y_1 = 9, a = -36$ в первое из уравнений (5). получаем для x - координаты точки $2P$ значение 25 / 4, а второе из уравнений (5) дает для y - координаты значение -35/ 8.

Существует несколько способов доказать, что принятое выше определение операции сложения $P + Q$ превращает множество точек на эллиптической кривой в абелеву группу. Можно использовать результаты из проективной геометрии, из комплексного анализа двоякопериодических функций или алгебраическое доказательство, использующее теорию дивизоров на кривых. Доказательства каждого из этих типов можно найти в источниках, указанных в списке литературы.

Если n — целое число, то, как и в любой абелевой группе, nP обозначает сумму n точек P при $n > 0$ и сумму $|n|$ точек $-P$, если $n \le 0$.

Еще несколько слов о «точке в бесконечности» O. По определению, это — тождественный элемент группового закона. На рисунке (см. выше) следует себе представлять ее расположенной на оси y в предельном направлении, определяемом все более «крутыми» касательными к кривой. Она является «третьей точкой пересечения» с кривой для любой вертикальной прямой: такая прямая пересекается с кривой в точках вида (x_1, y_1), $(x_1, -y_1)$ и O. Мы изложим сейчас более естественный способ введения точки O.

Под *проективной плоскостью* мы понимаем множество классов эквивалентности троек (X, Y, Z) (не все компоненты равны нулю), при этом две тройки называются эквивалентными, если одна из них — скалярное кратное другой, т.е. $(\lambda X, \lambda Y, \lambda Z) \sim (X, Y, Z)$. Такой класс эквивалентности

называется проективной точкой. Если проективная точка имеет ненулевую компоненту Z, то существует, причем только одна, тройка в ее классе эквивалентности, имеющая вид (x, y,1): просто полагаем $x = X/Z$, $y = Y/Z$. Тем самым проективную плоскость можно представить как объединение всех точек (x, y) обычной («аффинной») плоскости с точками, для которых Z =0. Эти последние точки составляют то, что называется бесконечно удаленной прямой; наглядно ее можно, себе представить на плоскости как «горизонт». Любому алгебраическому уравнению F(x,y) = 0 кривой в аффинной плоскости отвечает уравнение $\tilde{F}(X, Y, Z) = 0$, которому удовлетворяют соответствующие проективные точки: нужно заменить x на X/Z, y — на Y/Z и умножить на подходящую степень Z, чтобы освободиться от знаменателей. Например, если применить эту процедуру к аффинному уравнению (1) эллиптической кривой, то получится «проективное уравнение» $Y^2Z = X^3 + aXZ^2 + bZ^3$. Этому уравнению удовлетворяют все проективные точки (X, Y, Z) с $Z \neq 0$, для которых соответствующие аффинные точки (x, y), где x = X/Z, y = Y/Z, удовлетворяют (1). Помимо них, какие еще точки бесконечно удаленной прямой удовлетворяют последнему уравнению? Если положить в уравнении Z = 0, то уравнение примет ви д $X^3 = 0$, т.е. $X = 0$. Но единственный класс эквивалентности троек с X =0, Z =0 — это класс тройки (0, 1, 0). Это и есть точка, которую мы обозначили O; она лежит на пересечении оси y с бесконечно удаленной прямой.

. При рассмотрении эллиптических кривых с точки зрения теории алгебраических чисел обнаруживается глубокая аналогия между координатами «точек, делящих эллиптические кривые на n частей» (т.е. таких точек P, что nP = O) и точками, делящими на n частей единичную окружность (которые соответствуют корням степени n из единицы в комплексной плоскости). Более подробные сведения об этом можно найти в[ ] .

Точки конечного порядка.

Порядком n точки P на эллиптической кривой называется такое наименьшее натуральное число, что nP = 0; разумеется, такого конечного n может и не существовать,в этом случае мы будем говорить о точке бесконечного порядка. Часто требуется найти точки конечного порядка на эллиптической кривой, в особенности, на эллиптических кривых, определенных над полем Q.

Пример 3. Найти порядок точки P = (2, 3) на $y^2 = x^3 + 1$.

Решение. Применяя (5), находим, что 2P = (0, 1), и вновь с помощью (5). что 4P = 2(2P) = (0,-1). Поэтому 4P = -2P и, следовательно, 6P = O. Тем самым

порядок P может быть равен 2, 3 или 6. Но $2P = (0,1) \neq O$, а если бы P имела порядок 3, то было бы $4P = P$, что неверно. Итак, P имеет порядок 6.

Эллиптические кривые над рациональными числами.

Еели в уравнении (1) a и b — рациональные числа, то естественно рассматривать рациональные решения (x, y), т.е. эллиптическую кривую над полем Q рациональных чисел. Теория эллиптических кривых над рациональными числами очень обширна. Было доказано, что соответствующие абелевы группы являются конечно 'порожденными (теорема Морделла). Это означает, что каждая из таких групп есть сумма конечной «подгруппы кручения» (точек конечного порядка) и подгруппы, порожденной конечным числом точек бесконечного порядка. Число (минимальное) образующих бесконечной части называется *рангом r*; оно равно нулю тогда и только тогда, когда вся группа конечна. Изучение ранга r и других свойств группы точек эллиптической кривой над Q связано со многими интересными вопросами теории чисел и алгебраической геометрии. Например, известный с древних времен вопрос «Существует ли прямоугольный треугольник с рациональными сторонами, площадь которого равна данному целому n?» эквивалентен следующему: «Верно ли, что ранг эллиптической кривой $y^2 = x^3 - n^2x$ больше нуля?» Случай n = 6 и прямоугольного треугольника со сторонами 3, 4 и 5 соответствует точке P = (-3,9) из примера 2, которая является точкой бесконечного порядка на эллиптической кривой $y^2 = x^3 - 36x$. Более подробно см.[ ].

Эллиптические кривые над конечным полем.

Будем предполагать, что К — конечное поле F_q, с $q = p^r$ элементами. Пусть Е — эллиптическая кривая, определенная над F_q. Если p = 2 или 3, то E задается уравнением вида (2) или (3) соответственно.

Легко усмотреть, что эллиптическая кривая может иметь не более 2q+1 различных F_q-точек, т. е. точку в бесконечности и не более чем $2q$ пар (x, y), $x.y \in F_q$, удовлетворяющих (1) (или (2) или (3), если p = 2 или 3). А именно, для каждого из q возможных значений x имеется не более двух значений y, удовлетворяющих (1).

Но так как лишь у половины элементов F_q^* имеются квадратные корни, естественно ожидать (если бы $x^3 + ax + b$ были случайными элементами поля), что количество F_q-точек примерно вдвое меньше этого числа. Точнее, пусть χ

— квадратичный характер F_q. Это — отображение, переводящее $x \in F_q^*$ в 1 или -1 в зависимости от того, является или нет элемент x квадратом элемента из F_q (полагаем также χ (0) = 0). Например, если q — это простое число p, то χ {x) = $\left(\frac{x}{p}\right)$ есть символ Лежандра . В любом случае число решений $y \in F_q$, уравнения $y^2 = u$ равно 1 + $\chi(u)$ и, значит, число решений (1) (с учетом точки в бесконечности) равно

$$1 + \sum_{x \in F_q} (1 + \chi(x^3 + ax + b)) = q + 1 + \sum_{x \in F_q} \chi(x^3 + ax + b) \quad (6)$$

Следует ожидать, что $\chi(x^3 + ax + b)$ одинаково часто принимает значения + 1 и — 1. Вычисление суммы очень похоже на «случайное блуждание», когда мы подбрасываем монету q раз. продвигаясь на шаг вперед, если выдал герб, и назад — если решетка. Из теории вероятностей известно, что после q бросаний результат случайного блуждания оказывается на расстоянии порядка $\sqrt{q}$ от исходного положения. Сумма $\sum \chi(x^3 + ax + b)$ ведет себя в чем-то аналогично случайному блужданию. Тсчнее, удалось доказать, что она ограничена величиной $2\sqrt{q}$. Этот результат — теорема Хассе

Теорема Хассе.

Пусть N — число F_q*-точек на эллиптической кривой, определенной над* F_q. *Тогда*

$$|N \le (q + 1)| \le 2\sqrt{q}$$

В дополнение к числу N элементов на эллиптической кривой над Fg нам бывает нужно знать строение этой абелевой группы. Она — не обязательно циклическая, но можно показать, что она— всегда произведение двух циклических групп

2.8.2. Построение криптосистем на эллиптических кривых

Цель этого параграфа — построение систем с открытым ключом, основанных на конечной абелевой группе эллиптической кривой, определенной над F_q. Прежде чем описывать криптосистемы, нужно обсудить некоторые вспомогательные понятия.

Кратные точки. Для эллиптических кривых аналогом умножения двух

элементов группы F_q^* служит сложение двух точек эллиптической кривой E, определенной над F_q. Таким образом, аналог возведения в степень к элемента из F_q^* — это умножение точки $P \in E$ на целое число k. Возведение в k-ю степень в F_q^* методом повторного возведения в квадрат можно осуществить за $O(\log k \log^3 q)$ двоичных операций (см. предложение II. 1.9). Аналогично, мы покажем, что кратное $kP \in E$ можно найти за $O(\log k \log^3 q)$ двоичных операций методом повторного удвоения.

Пример 1. Чтобы найти $100P$, записываем $100P = 2(2(P + 2(2(2(P + 2P)))))$ и приходим к цели, производя 6 удвоений и 2 сложения точек на кривой.

Предложение 2.1. *Пусть эллиптическая кривая E определена уравнением Вейерштрасса (уравнением (1),(2) или (3) из предыдущего параграфа) над конечным полем F_q. Если задана точка P на E, то координаты kP можно вычислить за O ($\log k \log^3 q$) двоичных операций.*

Замечание 1. Оценки времени работы в предложении 2.1 не являются наилучшими, особенно для конечных полей характеристики $p = 2$. Нам, однако, достаточно этих оценок, которые следуют из наиболее очевидных алгоритмов арифметики в конечных полях.

Замечание 2. Если известно число N точек на нашей эллиптической кривой E и если $k > N$, то в силу равенства $NP = O$ мы можем заменить к его наименьшим неотрицательным вычетом по модулю N; в этом случае временная оценка заменяется на $O(\log^4 q)$

(напомним, что $N \le q + 1 + 2\sqrt{q} = O(q)$). Рене Шуф (R. Schoof) предложил алгоритм, вычисляющий N за $O(\log^8 q)$ двоичных операций.

Представление открытого текста. Мы намереваемся кодировать наши открытые тексты точками некоторой заданной эллиптической кривой E, определенной над конечным полем F_q Мы хотим ото осуществить простым и систематическим способом так чтобы открытый текст m (который можно рассматривать как целое число из некоторого интервала) можно было легко прочитать, зная координаты соответствующей точки P_m. Заметим, что это «кодирование» - не то же самое, что засекречивание. Позднее мы будем рассматривать способы шифрования точек P_mоткрытого текста. Однако законный пользователь системы должен быть в состоянии восстановить m после дешифрования точки шифртекста.

Следует сделать два замечания. Во-первых, не известно детерминистического полиномиального (по log q) алгоритма для выписывания

большого числа точек произвольной эллиптической кривой E над F_q. Однако, как мы увидим далее, существуют вероятностные алгоритмы с малой вероятностью неудачи. Во-вторых, порождать случайные точки на E недостаточно: чтобы закодировать большое число возможных сообщений m, необходим какой-то систематический способ порождения точек, которые были бы связаны с m определенным образом например, чтобы x-координата имела с m простую связь.

Приведем один возможный вероятностный метод представления открытых текстов как точек на эллиптической кривой E, определенной над F_q, где $q = p^r$ предполагается большим (и нечетным (см ниже упражнение 8 при $q = p^r$)). Пусть κ - достаточно большое целое число, настолько, что можно считать допустимым, если попытка представить в нужном нам виде элемент («слово») открытого текста m оказывается неудачной в одном случае из 2^κ; практически достаточно $\kappa = 30$ или, на худой конец, $\kappa = 50$. Пусть элементы нашего сообщения m - целые числа, $0 \le m \le M$. Предположим также, что выбранное нами конечное толе имеет q элементов, $q > M\kappa$. Записываем петые числа от 1 до $M\kappa$ в виде $m\kappa + j$, где $1 \le j \le \kappa$ устанавливаем взаимно однозначное соответствие между такими числами и некоторым множеством элементов из F_q. Например, можно записать такое число как r -значное числе в p-ичной системе счисления и, отождествляя цифры в этой записи с элементами $F_p \cong Z / pZ$, рассматривать их как коэффициенты многочлена степени не выше r - 1 над F_p. соответствующего элементу поля F_q. Таким образом, числу $(a_{r-1}a_{r-2}...a_1a_0)$ ставится в соответствие многочлен $\sum_{i=0}^{r-1} a_i X^i$ который, будучи рассмотрен по модулю некоторого фиксированного неприводимого многочлена степени r над F_p, дает элемент F_q.

Итак, при данном m при каждом j = 1,2,... , κ мы получаем элемент x из F_q, соответствующий $m\kappa + j$. Для такого x вычисляем правую часть уравнения

$$y^2 = f(x) = x^3 + ax + b$$

и пытаемся найти квадратный корень из $f(x)$, используя метод, изложенный в конце § II. 2. Хотя этот алгоритм был приведен для простого поля F_p, он дословно переносится на любое конечное поле F_q. Чтобы воспользоваться им, нужно иметь элемент g в этом поле, не являющийся квадратом; его можно легко найти с помощью вероятностного алгоритма.) Если мы находим такой y, что

$y^2 = f(x)$. то берем $P_m = (x, y)$. Если f(x) не оказывается квадратом, то увеличиваем j на 1 и повторяем попытку с соответствующим значением x. Если мы находим x, для которого j(x) — квадрат, прежде, чем j превысит κ, то мы можем восстановить m по известной тетке (x, y) с помощью формулы $m = [(\tilde{x} - j)/\kappa]$, где $\tilde{x}$ — целое число, соответствующее x при установленном взаимно однозначном соответствии между целыми числами и элементами F_q. Так как f(x) — квадрат приблизительно в 50% случаев, то вероятность того, что метод не приведет κ результату и мы не найдем точки P_m с x-координатой, соответствующей целому числу $\tilde{x}$ между $m\kappa + 1$ и $m\kappa + \kappa$, равна примерно 2^{-x}. (Точнее, вероятность того, что f(x) есть квадрат, фактически равна $N/(2q)$, однако $N/(2q)$ очень близко к 1/2.)

Дискретный логарифм на *E*.

Определение. Пусть E — эллиптическая кривая над F_q, и B — точка на E. *Задачей дискретного логарифмирования на E* (с основанием B) называется задача нахождения для данной точки $P \in E$ такого целого числа $x \in Z$ (если оно существует), что $xB = P$.

Вполне возможно, что задача дискретного логарифмирования на эллиптической кривой окажется более трудной для решения, чем задача дискретного логарифмирования в конечных полях. Наиболее сильные методы, разработанные для конечных полей, по-видимому, не работают в случае эллиптических кривых. Это обстоятельство поособенному отчетливо проявляется в случае полей характеристики 2. Как объясняется в обзорной статье Одлыжко, упомянутой в списке литературы, специальные методы решения задачи дискретного логарифмирования в F_{2^r} позволяют сравнительно легко вычислять дискретные логарифмы и, следовательно, вскрывать соответствующие криптосистемы, если r не выбрано очень большим. Аналогичные системы, использующие эллиптические кривые над F_q (см. ниже), судя но всему, являются надежными при значительно меньших значениях r. Так как имеются практические причины (связанные с устройством ЭВМ и программированием) предпочтительности арифметических операций над полям F_{2^r}, криптосистемы с открытым ключом, рассматриваемые ниже, могут оказаться более удобными для практического применения, чем системы, основанные на задаче дискретного логарифмирования в F_q^*

Основные преимущества криптосистем на эллиптических кривых заключаются в том, что не известны субэкспоненциальные алгоритмы E

вскрытия этих систем, если в них не используются суперсингулярные кривые, а также кривые, порядки которых делятся на большое простое число.

Теперь мы опишем аналоги систем с открытым ключом из § IV. 3, основанные на задаче дискретного логарифмирования на эллиптической кривой, определенной над конечным полем F_q.

Аналог ключевого обмена Диффи-Хеллмана. Предположим, что абоненты А и Б хотят договориться о ключе, которым будут впоследствии пользоваться в некоторой классической криптосистеме. Прежде всего они открыто выбирают какое-либо конечное поле F_q и какую-либо эллиптическую кривую E над ним. Их ключ строится по случайной точке P на этой эллиптической кривой. Если у них есть случайная точка P, то, например, ее x-координата дает случайный элемент F_q который можно затем преобразовать в x-разрядное целое число в p-ичной системе счисления (где $q = p^r$), и это число может служить ключом в их классической криптосистеме. (Здесь мы пользуемся словом «случайный» в неточном смысле; мы лишь хотим сказать, что выбор из некоторого большого множества допустимых ключей произволен и непредсказуем). Они должны выбрать точку P так, чтобы все их сообщения друг другу были открытыми и все же никто, кроме них двоих, ничего бы не знал о P.

Абоненты (пользователи)А и Б первым делом открыто выбирают точку $B \in E$ в качестве «основания»; B играет ту же роль, что образующий q в системе Диффи-Хеллмана для конечных полей. Мы, однако, не требуем, чтобы B была образующим элементом в группе точек кривой E. Эта группа может и не быть циклической. Даже если она циклическая, мы не хотим тратить время на проверку того, что B — образующий элемент (или даже на нахождение общего числа N точек, которое нам не понадобится в последующем). Нам хотелось бы, чтобы порожденная B подгруппа была большой, предпочтительно того же порядка величины, что и сама E. Позже мы обсудим этот вопрос. Пока что предположим, что B — взятая открыто точка на E весьма большого порядка (равного либо N, либо большому делителю N).

Чтобы образовать ключ, А вначале случайным образом выбирает целое число a, сравнимое по порядку величины с q (которое близко к N); это число он держит в секрете. Он вычисляет $aB \in E$ и передает эту точку открыто.Абонент Б делает то же самое: он выбирает случайно b и открыто передает $bB \in E$. Тогда используемый ими секретный ключ - это $P = abB \in E$. Оба пользователя могут вычислить этот ключ. Например, А знает bB (точка была передана открыто) и свое собственное секретное a. Однако любая третья сторона знает лишь aB и bB. Кроме решения задачи дискретного логарифмирования - нахождения a по B и aB (или нахождения b по B и bB по-видимому, нет способа найти abB. зная лишь

aB и *bB*.

Аналог системы Мэсси-Омуры. Как и в случае конечного толя, это — криптосистема с открытым ключом для передачи элементов сообщения *m*, которые мы теперь предположим представленными точками P_m фиксированной (и не скрываемой) эллиптической кривой *E* над F_q (*q* берется большим). Предполагается также, что обшее число *N* точек на *E* вычислено и не составляет секрета. Каждый пользователь системы секретно выбирает такое целое случайное число *e* между 1 и *N*, что НОД (*e, N*) = 1. Используя алгоритм Евклида. он находит затем обратное e^{-1} к числу *e* по модулю *N*, т.е. такое целое число *d*. что $de \equiv 1 \pmod N$. Если А хочет послать Б сообщение P_m, то он сначала посылает ему точку $e_A P_m$ индекс *A* указывает на пользователя А). Это ничего не говорит Б. который, не зная ни e_A, ни, d_A, не может восстановить P_m, Однако, не придавая этому значения, он умножает ее на свое e_B и посылает обратно А. На третьем шаге А должен частично раскрыть свое сообщение, умножив $e_B e_A P_m$ на d_A. Так как $NP_m = O$ и $d_A e_A = 1 \pmod N$, при этом получается точка $e_B P_m$, которую А возвращает Б. Тот может теперь прочитать сообщение, умножив точку $e_B P_m$ на d_B.

Заметим, что злоумышленник может знать $e_A P_m$, $e_B e_A P_m$ и $e_B P_m$ Если бы он умел решать задачу дискретного логарифмирования на *E*, то он мог бы определить e_B по первым двум точкам, вычислить $d_B = e_B \pmod N$ и $P_m = d_B e_B P_m$.

Аналог системы Эль-Гамаля. Это — другая криптосистема с открытым ключом для передачи сообщений P_m. Как и в описанной выше системе ключевого обмена, мы исходим из данных несекретных: а) конечного поля F_q, б) определенной над ним эллиптической кривой *E* к в) точки-«основания» В на ней. (Знать общее число *N* точек на *E* нам не нужно.) Каждый из пользователей выбирает случайное целое число *a*, которое держит в секрете, затем находит и делает общедоступной точку *aB*.

Чтобы послать Б сообщение P_m, А выбирает случайно целое число *k* и посылает пару точек $\{kB,\ P_m + k\{a_B B\})$ (где $a_B B$ — открытый ключ Б). Чтобы прочитать сообщение, Б умножает первую точку из полученной пары на свое секретное число a_B и вычитает результат умножения из второй точки:

$$P_m + k\{a_B B) - a_B (k B) = P_m .$$

Таким образом. А посылает замаскированное P_m вместе с «подсказкой» *k*

B, при помощи которой можно снять «маску» $k\,a_B B$, если знать секретное число a_B. Злоумышленник, который умеет решать загачу дискретного логарифмирования на E, может, конечно, найти a_B зная $a_B B$ и B.

Выбор кривой и точки. Существуют различные способы выбора эллиптической кривой и (в системах Диффи-Хеллмана и Эль-Гамаля) точки B на ней.

Случайный выбор (E,B). Взяв какое-либо большое конечное поле F_q, можно следующим образом осуществить одновременный выбор E и $B = (x, y) \in E$. (Будем предполагать, что характеристика p поля F_q больше 3, так что эллиптическая кривая задана уравнением (1) из § 1; при $q = 2^r$ или 3^r нетрудно сделать очевидные изменения в дальнейшем изложении.) Выбираем сначала случайным образом три элемента из F_q^* в качестве x, y, a. Далее полагаем $b = y^2 - (x^3 + ax)$. Убеждаемся в том, что кубический многочлен $x^3 + ax + b$ не имеет ' кратных корней, что равносильно проверке условия $4a^3 + 27b^2 \neq 0$. (Если это условие не выполняется, берем другую случайную тройку x, y, a.) Полагаем $B = (x, y)$. Тогда B — точка на эллиптической кривой $y^2 = x^3 + ax + b$.

Число N точек кривой можно найти несколькими способами. Первый полиномиальный алгоритм для вычисления $\#E$, построенный Рене Шуфом (Rene Schoof), является даже детерминистическим. Он основывается на нахождении значения $\#E$ по модулю l для всех простых чисел l, меньших некоторой границы. Для этого анализируется действие автоморфизма Фробениуса (отображения в p-ю степень) на точках порядка l.

В статье Шуфа оценка времени работы была фактически $O(\log^8 q)$, т. е. хотя и полиномиальной, однако быстро растущей. Вначале казалось, что алгоритм не имеет практического значения. Однако с тех пор многие пытались повысить скорость алгоритма Шуфа (Миллер (V. Miller), Элкис (N. Elkis). Бухман (J. Buchman), Мюллер (V. Miiller), Менезес (A. Menezes), Чарлап (L. Charlap), Коули (R. Coley) и Роббинс (D.Robbins)). Кроме того. Эткин (A. O. L. Atkin) разработал другой метод, который, хотя и не гарантирует полиномиального времени работы, на практике дает весьма удовлетворительные результаты. В результате всех этих усилий стало возможным вычислять порядок произвольной эллиптической кривой над F_q, если q — степень простого числа, записываемая 50 или даже 100 знаками.

Следует также отметить, что хотя для реализации систем Диффи-Хеллмана или Эль-Гамаля знать N не нужно, на практике желательно быть уверенным в их надежности, которая зависит от того, имеет ли N большой простои делитель. Если N есть произведение малых простых чисел, то для

решения задачи дискретного логарифмирования можно использвать метод Полига-Силвера-Хеллмана (см. § IV. 3). Заметим, что метод Полига-Силвера-Хеллмана решает задачу дискретного логарифмирования в любой конечной абедевой группе (в отличие от также рассмотренного в § IV.3 алгоритма зычисления индекса, который зависят от особенностей F_q^*). Таким обрезом, нужно знать, что N не есть произведение малых простых чисел и не похоже, что это можно узнать, если не найти фактически значение N.

Редукция глобальной пары {E,B) по модулю p. Упомянем теперь второй способ нахождения пары, состоящей из эллиптической кривой и точки на ней. Выберем сначала раз и навсегда «глобальную» эллиптическую кривую и точку бесконечного порядка на ней. Итак, пусть E — эллиптическая кривая, определенная над полем рациональных чисел (или, для большей общности, можно было бы использовать эллиптическую кривую, определенную над некоторым числовым полем), и B — точка беексконечного порядка на E.

Пример 2. Точка B = (0, 0) является точкой бесконечного порядка на эллиптической кривой E: $y^2 + y = x^3 - x$ и фактически порождает всю группу рациональных точек на E.

Пример 3. Точка B = (0, 0) является точкой бекспечного порядка на, E: $y^2 + y = x^3 - x^2$ и порождает всю группу рациональных точек.

Далее, мы выбираем большое простое число p (или, если наша эллиптическая кривая определена над расширением K поля Q, выбираем некоторый простой идеал в K) и рассматриваем редукцию E и B по модулю p. Точнее, для всех p, за исключением нескольких малых простых чисел, коэффициенты в уравнении для E имеют взаимно простые с p знаменатели и, следовательно, могут рассматриваться пак коэффициенты в уравнении по модулю p. Если сделать замену переменных, приведя полученное уравнение над F_p к виду $y^2 = x^3 + ax + b$ то кубический многочлен в главой части не будет иметь кратных корней (за исключением нескольких малых простых p) и дает поэтому эллиптическую кривую над F_p (которую мы будем обозначать $E(\mathrm{mod}\ p)$). Координаты точки B, будучи также приведенными по модулю p, дают точку на эллиптической кривой $E\ (\mathrm{mod}\ p)$, которую мы будем обозначать $B\ (\mathrm{mod}\ p)$.

При использовании этого второго способа мы раз и навсегда фиксируем E и B и за счет этого получаем много различных возможностей посредством изменения простого p.

Порядок точки B. С какой вероятностью «случайная» точка B на «случайной» эллиптической кривой оказывается порождающим элементом? Или, в случае нашего второго метода выбора (E, B), какова вероятность того, что (для случайного p) точка B при редукции по модулю дает образующий

элемент кривой E (mod p)? Этот вопрос близок к следующему вопросу о мультипликативных группах конечны: полей: пусть целое b фиксированно, а простое p случайно; какова вероятность того, что b — образующий в F_p^*? Вопрос изучался как для конечных полей, так и для эллиптических кривых. Более подробное изложение можно найти в работе Гулты (Gupta) и Мурти (Murty), приведенной в списке литературы.

Описанные криптосистемы могут быть надежными, даже если точка B не является порождающим элементом Фактически нужно, чтобы в циклической группе, порождаемой B: задача дискретного логарифмирования не была эффективно разрешима. Это будет так (т.е. все известные методы решения задачи дискретного логарифмирования в произвольной абелевой группе оказываются слишком медленными), если порядок B делится на очень большое простое число, скажем, имеющее порядок величины, близкий к N.

Один из способов гарантировать что наш выбор B является надлежащим (а фактически, что B порождает эллиптическую кривую) — это взять такую эллиптическую кривую и такое конечное поле, чтобы число точек N было простым чистом. Тогда всякая точка $B \neq O$ будет порождающим элементом. Если использовать первый из описанных выше методов, то при фиксированном F_p можно продолжать выбор пар (E, B), пока не найдется такая, для которой число точек на E есть простое число (что можно определить одним из тестов на простоту, обсуждавшихся в § V.1). Если применять второй метод, то для фиксированной глобальной эллиптической кривой E над Q можно продолжать выбирать простые p, иона не найдем кривую E(mod p), число точек на которой — простое. Как долго нам придется ждать? Этот вопрос аналогичен следующему вопросу о группах F_p^*: является ли (p-1)/2 простым числом, т.е. верно ли, что любой элемент, отличный от ±1, — либо порождающий, либо квадрат порождающего элемента (см. упражнение 13 к §II.1)? Ни для эллиптических кривых, ни для конечных полей вопрос дока не получил явного ответа, однако в обоих случаях предполагается, что вероятность выбора p с требующимся свойством есть $O(1/\log p)$

Зашечание. Для того чтобы E(mod p) имела простой порядок N при большом p, надо выбирать E так, чтобы она имела тривиальное кручение, т.е. чтобы на ней не было точек конечного порядка. кроме O. В противном случае N будет делиться на порядок периодической подгруппы.

2.8.3. Примеры эллиптических кривых и их применение

Пример. пусть p = 23. Рассмотрим эллиптическую кривую $y^2 = x^2 + x + 1$. В

этом случае $a=b=1$ и мы имеем $4\times 1^2 + 27\times 1^2 (mod\ 23) = 8 \neq 0$, что удовлетворяет условиям эллиптической группы по модулю 23.

Для эллиптической группы рассматриваются только целые значения от (0, 0) до (*p, p*) в квадранте неотрицательных чисел, удовлетворяющих уравнению по модулю *p*. В таблице 1. представлены точки (отличные от *O*), являющиеся элементами $E_{23}(1,1)$. В общем случае такой список создается по следующим правилам.

1. Для каждого такого значения *x*, что $0 \le x < p$, вычисляется $x^2 + ax + b(mod\ p)$.

2. Для каждого' из полученных на предыдущем шаге значений выясняется, имеет ли это значение квадратный корень по модулю *p*. Если нет, то в $E_p(a,b)$ нет точек с этим значением *x*. Если же корень существует, имеется два значения *y*, соответствующих операции извлечения, квадратного корня (исключением является случай, когда единственным таким значением оказывается $y = 0$). Эти значения (*x, y*) и будут точками $E_p(a,b)$

Таблица 1 Точка $E_{23}(1,1)$ на эллиптической кривой

(O,1)	(6,4)	(12,19
(0,22)	(6, 19)	(13,7)
(1,7)	(7,11)	(13,16
(1,16)	(7,12)	(17,3)
(3,10)	(9.7)	(17,20
(3,13)	(9,16)	(18,3)
(4,0)	(11,3)	'
(5,4)	(11,20)	(19,5
(5,19)	(12,4)	(19,1

Пусть $P = (3, 10)$ и $Q = (9, 7)$.

Тогда

$$\lambda = \frac{7-10}{9-3} = \frac{-3}{6} = \frac{-1}{2} \equiv 11 mod\ 23$$

$$x_3 = 11^2 - 3 - 9 = 109 \equiv 17\ mod\ 23$$

$$y_3 = 11(3-(-6)) = 89 \equiv 20\ mod\ 23$$

Поэтому $P + Q = (17, 20)$. Чтобы найти $2P$, найдем

$$\lambda = \frac{3(3^2)+1}{2\times 10} = \frac{5}{20} = \frac{1}{4} \equiv 6 \bmod 23$$

$$x_3 = 6^2 - 3 - 9 = 30 \equiv 7 \bmod 23$$

$$y_3 = 6(3-7) - 10 = -34 \equiv 12 \bmod 23$$

и поэтому 2P = (7, 12).

Пример обмена ключами по схеме Диффи-Хеллмана

В качестве примера возьмем p = 211, $E_p(0,-4)$ (что соответствует кривой $y^2 = x^3 - 4$ и G = (2, 2). Можно подсчитать, что 241G = O. Личным ключом пользователя A является n_A= 121, поэтому открытым ключом A будет P_A = 121(2, 2) = (115, 48). Личным ключом пользователя B является n_B = 203 , поэтому открытым ключом участника B будет 203(2, 2) = (130, 203). Общим секретным ключом является 121(130, 203) = 203(115, 48) = (161,169).

Обратите внимание на то, что общий секретный ключ представляет собой пару чисел. Если этот ключ предполагается использовать в качестве сеансового ключа для традиционного шифрования, то из этой пары чисел необходимо генерировать одно подходящее значение. Можно, например, использовать просто координату x или некоторую простую функцию от x.

Шифрование/расшифрование с использованием эллиптических кривых

В специальной литературе можно найти анализ нескольких подходов к шифрованию/дешифрованию, предполагающих использование эллиптических кривых. В этом разделе мы рассмотрим, пожалуй, наиболее простой из этих подходов. Первой задачей в рассматриваемой системе является шифрование открытого текста сообщения m, которое должно будет пересылается в виде значения x – y для точки P_m. Здесь точка P_m будет представлять шифрованный текст и впоследствии будет дешифроваться. Обратите внимание, что мы не можем закодировать сообщение просто координатой x или y точки, так как не все такие координаты имеются в $E_p(a,b)$ (см., например, табл. 6.4). Опять же, существует несколько подходов к такому кодированию, но мы их рассматривать не будем — для наших целей достаточно просто отметить, что имеются относительно прямые методы, которые могут быть здесь применены.

Как и в случае системы обмена ключами, в системе шифрования, дешифрования в качестве параметров тоже рассматривается точка G и эллиптическая группа $E_p(a,b)$. Пользователь А выбирает личный ключ n_A и генерирует открытый ключ $P_A = n_A + G$. Чтобы зашифровать и послать сообщение P_m пользователю В, пользователь А выбирает случайное положительное целое число к и вычисляет шифрованный текст C_m, состоящий из пары точек

$$C_m = \{kG,\ P_m + k P_B\}.$$

Заметим, что сторона А использует открытый ключ стороны В: P_B. Чтобы дешифровать этот шифрованный текст, В умножает первую точку в паре на секретный ключ В и вычитает результат из второй точки:

$$P_m + k P_B \text{ - } n_B(kG) = P_m + k(n_B G) - n_B(kG) = P_m$$

Пользователь А замаскировал сообщение P_m с помощью добавления к нему $k P_m$. Никто, кроме этого пользователя, не знает значения k, поэтому, хотя P_B и является открытым ключом, никто не сможет убрать маску $k P_B$. Однако пользователь А включает в сообщение и "подсказку", которой будет достаточно, чтобы утрать маску тому, кто имеет личный ключ n_B. Противнику для восстановления сообщения придется вычислить k по данным G и kG, что представляется трудной задачей.

В качестве примера подобного шифрования рассмотрим случай p = 751, E_p(-1,188) (что соответствует кривой $y^2 = x^3 - x + 188$ и G = (0, 376). Предположим, что пользователь А собирается отправить пользователю В сообщение, которое кодируется эллиптической точкой P_m =(562, 201), и что пользователь А выбирает случайное число k = 386. Открытым ключом В является P_B=(201,5). Мы имеем 386(0,376)= (676,558) и (562,201) + 386(201, 5) = (385, 328). Таким образом, пользователь А должен послать шифрованный текст {(676, 558), (385, 328)}.

2.8.4. Безопасность криптографии с использованием эллиптических кривых

Безопасность, обезпечизаемая криптографическим подходом на основе

эллиптических кривых, зависит от того, насколько трудной для решения оказывается задач; определения k по данным kP и P. Эту задачу обычно называют проблемой логарифмирования на эллиптической кривой. Наиболее быстрым из известных на сегодня методов логарифмирования на эллиптический кривой является так. называемый ρ-метод Полларда (Pollard). В таблице 2. сравнивается эффективность этого метода и метода разложения на простые множители с помощью решета в поле чисел общего вида. Как видите, по сравнению с RSA в случае применения методов криптографии на основе эллиптических кривых примерно тот же уровень защиты достигается со значительно меньшими значениями длины ключей.

К тому же при равных длинах ключей вычислительные усилия, требуемые при использовании RSA и криптографии на основе эллиптических, кривых, не сильно различаются [JLJRI97]. Таким образом, в сравнении с RSA при равных уровнях защиты явное вычислительное преимущество принадлежит криптографии на основе эллиптических кривых с более короткой длиной ключа.

Таблица 2 Вычислительные усилия, необходимые для криптоанализа при использовании эллиптических кривых и RSA

Размер ключа	MIPS-годы
150	$3{,}8 \times 10^{10}$
205	$7{,}1 \times 10^{18}$
234	$1{,}6 \times 10^{28}$

а) Логарифмирование на эллиптической кривой с помощью *p*-метода Полларда

Размер ключа	MIPS-годы
512	3×10^{4}
768	3×10^{2}
1024	3×10^{11}
1280	1×10^{14}
1536	3×10^{16}
2048	3×10^{20}

б) Разложение на множители в целых числах с помощью метод! решета в поле чисел общего вида

2.9. ШИФРЫ СОВЕРШЕННЫЕ И БЛИЗКИЕ К СОВЕРШЕННЫМ

Вопрос о теоретической стойкости шифров систематически исследовал К. Шеннон в фундаментальной работе [23]. Заметим, что независимо изучение теоретической стойкости шифров проводил коллектив под руководством В.А. Котельникова [24].

2.9.1. Шифры совершенные по К. Шеннону

Шеннон ввел понятие совершенного шифра, точнее, совершенного по открытому тексту. Обозначим: X, Y, K, F – множества соответственно открытых текстов, шифртекстов, ключей, отображений. Отображение $f_k : X \to Y,\ k \in K$ - это ограничение отображения $f : X \times K \to Y$ на множество $X \times \{k\},\ f \in F$. Здесь $\{k\}$ – множество из одного элемента. Для дальнейшего изложения удобным будет следующее определение.

Определение 1. Тройка множеств X, Y, K с функцией

$f: X \times K \to Y$ $A = (X, K, Y, f)$ называется шифром, если выполнены условия [25, с.89]:

1. Функция f сюръективна.
2. Для любого $k \in K$ функция f_k инъективна.

Определение 2. Шифр (X, K, Y, f) с вероятностными распределениями $P(x) = (P(x), x \in X)$, $P(k) = (P(k), k \in K)$ называется совершенным (при нападении на xЄX и перехвате yЄY), если для любых xЄX и yЄY [15, с.153]: p(x/y)=p(x).

Иными словами, совершенным по тексту называется шифр, который имеет свойство: никакая перехваченная противником криптограмма не добавляет противнику никакой информации об исходном тексте, т.е. вероятность определения исходного текста при знании соответствующей ему криптограммы равна вероятности использования этого текста в языке.

Если противник попытается вскрыть такой шифр методом полного перебора, то он получит набор из всех возможных осмысленных текстов соответствующей криптограмме длины.

Рассмотрим пример под названием «Одноразовый блокнот» («One Time Pad»).

Пусть имеется русский алфавит в виде таблицы:

А	Б	В	Г	Д	Е	Ё
Ж	З	И	Й	К	Л	М
Н	О	П	Р	С	Т	У
Ф	Х	Ц	Ч	Ш	Щ	Ъ
Ы	Ь	Э	Ю	Я		

Для зашифрования и расшифрования в «одноразовом блокноте» используется следующая таблица шифрования, полученная путем случайного перемешивания:

Ж	Ш	Г	А	Ь	Ы	Д
П	О	Р	Н	Е	Ъ	С
З	В	И	Э	Я	Б	Й
М	Л	К	Ф	Ч	Ё	У
Х	Ц	Т	Ю	Щ		

Шифрование: первой букве А исходного текста соответствует буква Ж шифрованного текста, букве Я – буква Щ и так далее. Шифрование происходит следующим образом: перейдя к очередной букве исходного текста, шифровальщик заменяет ее в тексте на соответствующую букву из таблицы шифрования. Расшифрование происходит в обратном порядке.

Таблица шифрования вырабатывается в двух экземплярах и более (в зависимости от числа участников информационного обмена), которые распределяются между участниками. Таблицу шифрования следует хранить в секрете.

Очевидно, что без знания этой таблицы противник не сможет добыть открытый текст.

В битовом варианте данный шифр работает следующим образом.

Исходный текст преобразуется в битовую последовательность.

Генерируется случайная битовая последовательность (например, с помощью генератора случайных чисел), длина которой равна длине сообщения. Этак последовательность является ключом.

Этот ключ по защищенному каналу передается всем участникам защищенного информационного обмена.

Ключ надлежит хранить в секрете.

Зашифрование происходит путем побитового сложения текста с ключом по модулю 2 (операция XOR).

Расшифрование происходит аналогичным образом.

В работе «Заметки о совершенных шифрах» А.С. Щепинов [22] доказывает, что группа подстановок элементов конечного множества содержит подмножества, некоторые из которых являются совершенными шифрами, по определению Шеннона.

Приведем ряд утверждений о совершенных шифрах без доказательства (доказательства подробно изложены в работе Зубова А.Ю.[15]).

1. Шифр с ограниченным ключом не является совершенным
2. У совершенного шифра может быть ограниченный ключ, но только если длина ключа равна длине сообщения
3. Для совершенного шифра справедливы неравенства $|x| \le |y| \le |k|$.

Определение 3. Шифр с условием p(k/y)=p(k) для любых kЄK и yЄY называется совершенным по ключу.

Криптостойкость совершенных шифров не вызывает сомнения. Однако использование их при защищенной передаче данных сопряжено с рядом трудностей. Например, длина ключа должна быть не меньше длины сообщения. То есть защищенный канал, по которому будет передан ключ, должен пропускать объем информации не меньше объема открытого текста.

В силу этой и других причин широкое распространение получили несовершенные шифры (например, в шифровании деловой переписки

малого и среднего бизнеса).

Поэтому вместе с определением совершенного шифра возникла необходимость определить шифр, близкий к совершенному.

2.9.2. Шифры, близкие к совершенным

В книге «Криптография» А. Бабаш и Г. Шанкин говорят о шифрах, близких к совершенным [25 с.304-305].

Пусть $A = A_1 \cdot A_2 = (X_1, K_1 \times K_2, Y_2, f)$ - произведение шифров такое, что $A_1=(X_1, K_1, Y_1, f_1)$, $A_2=(X_2, K_2, Y_2, f_2)$, $Y_1=X_2$, то есть множество шифробозначений первого шифра является множеством шифрвеличин второго шифра.

Произведение шифров, не являющихся совершенными, может дать шифр «близкий» к совершенному шифру.

Пусть (X, K, Y, f) – шифр модульного гаммирования (X = Y) и распределение вероятностей P(K)=(p(k), kЄK), такое, что p(k) > 0, для любого kЄK. Тогда квадратная матрица условных вероятностей M = ||p(y/x)|| является дважды стохастической и, следовательно, как известно из курса алгебры, существует предел

$$\lim_{k\to\infty} M^k = \frac{1}{|X|}\begin{pmatrix} 1 & .1 & . & 1 \\ 1 & . & & . 1 \\ . & . & & . . \\ 1 & . & . & 1 \end{pmatrix}$$

В частности, к-ая степень $(X, K, Y, f)^k$ шифра предсавляет собой шифр, переходные вероятности которого стремятся к величине 1/|X|при $к\to\infty$, то есть шифр $(X, K, Y, f)^k$ близок к совершенному шифру при достаточно большом k.

Аналогично определениям 2 и 3, введем понятия шифров, близких к совершенным.

Определение 4. Шифр с условием $P(x/y) - p(x) < \varepsilon$ для любых хЄX, уЄY назовем ε – совершенным по тексту, обозначение: A_ε^T.

Заметим, что конструкция Бабаша-Шанкина является примером такого шифра: к-я степень $(X, K, Y, f)^k$ является ε(k) – совершенным шифром (т.е. ε зависит от k).

Ясно, что совершенный по тексту шифр является A_ε^T для любого ε > 0, обратное, вообще говоря, неверно. Однако можно построить

последовательность $\left\{A_{\varepsilon}^{T}\right\}$ шифров со свойством ε→0.

Интересной является задача построения «шкалы» шифров, т.е. ε-отклонений от совершенного по тексту.

Точно так можно ввести понятие БСШ по ключу, обозначим A_{δ}^{K}. Теперь шифр $A_{\varepsilon\delta}^{KT}$, являющийся ε-совершенным по тексту и δ-совершенным по ключу, является обобщением шифра, совершенного по тексту и по ключу.

Известно, что свойства шифра быть совершенным по тексту и по ключу являются независимыми. При переходе от совершенного к БСШ эта независимость сохраняется.

И «меру отклонения» шифра от совершенного можно изучать по тому, насколько распределения P(x), P(k) отличаются от равномерных (ср. с теоремой 3 [15 с/156]).

Как известно, совершенные шифры используются крайне редко. Причина тому – требуемая большая длина ключа. Шифры $A_{\varepsilon\delta}^{KT}$, представляются разумным компромиссом в соотношении «цена - качество». Именно, можно задавать необходимые ε max и δ max, одновременно вычислив возможные для шифров данного вида ε min и δ min. Тогда множество допустимых в данной ситуации шифров изображается точками прямоугольника $\{(\varepsilon,\delta):\varepsilon\min\le\varepsilon\le\varepsilon\max,\ \delta\min\le\delta\le\delta\max\}$, и можно строить целевую функцию: стоимость реализации $A_{\varepsilon\delta}^{KT}$. После этого задача выбора оптимального при заданных условиях шифра может быть сформулирована так: найти экстремум функции двух переменных в области.

2.10. ЭКСТРЕМАЛЬНЫЕ ШИФРЫ

К. Шеннон в ряде своих основополагающих работ по теории шифрования [23] сформулировал условия, которым должен удовлетворять стойкий блочный шифр. Этими условиями он назвал **рассеивание** и **перемешивание**:

Рассеивание – это свойство шифра, при котором один символ (бит) исходного текста влияет на несколько символов (битов) шифртекста, оптимально - на все символы в пределах одного блока. Если данное условие выполняется, то при шифровании двух блоков данных с минимальными отличиями между ними должны получаться совершенно непохожие друг на друга блоки шифртекста. Точно такая же картина должна иметь место и для зависимости шифртекста от ключа - один символ (бит) ключа должен влиять на несколько символов (битов) шифртекста.

Перемешивание – это свойство шифра скрывать зависимости между символами исходного текста и шифртекста. Если шифр достаточно хорошо "перемешивает" биты исходного текста, то соответствующий шифртекст не содержит никаких статистических, и, тем более, функциональных закономерностей - опять же, для стороннего наблюдателя, обладающего лишь ограниченными вычислительными ресурсами.

А.Винокуров в серии выпусков по криптографии для электронного журнала "iNFUSED BYTES Online" раскрывает смысл этих свойств следующим образом: «Если шифр обладает обоими указанными свойствами в достаточной степени, то любые изменения в блоке открытых данных приводят к тому, что с точки зрения наблюдателя все символы (биты) в зашифрованном блоке получат новые значения, равновероятные в области их определения и независимые друг от друга. Так, если шифр оперирует информацией, представленной в двоичной форме, то инвертирование даже одного бита в блоке исходных данных приведет к тому, что все биты в соответствующем блоке шифрованных данных с вероятностью 1/2 независимо друг от друга так же поменяют свое значение. Такой шифр невозможно вскрыть способом, менее затратным с точки зрения количества необходимых операций, чем полный перебор по множеству возможных значений ключа. Данное условие является обязательным для шифра рассматриваемого типа, претендующего на то, чтобы считаться хорошим». [26]

В своей работе Столлингс говорит о следующем свойстве шифра, которое называет лавинным эффектом.

Лавинный эффект – это способность шифра максимально изменять выходные данные при минимальных изменениях во входных данных [16]. Например, изменение значения всего одного бита открытого текста или ключа должно отражаться в изменении значений многих битов

шифрованного текста.

По мнению В. Столлингса если малые изменения в открытом тексте приведут к малым изменениям в шифрованном тексте, то это позволит злоумышленнику сузить пространство ключей или область поиска открытого текста.

Далее предлагается свойство, которое объединяет в себе все вышеперечисленные свойства.

Понятие экстремальности

Здесь рассматривается двоичное представление текстов и ключей. В теоретическом аспекте это оправдано, так как мы всегда можем перевести любой текст в двоичную форму.

Далее нам потребуется одна из теорем Эрнесто Чезаро.

Теорема 2.1 (Э. Чезаро).

Вероятность того, что наибольшим общим делителем двух случайно выбранных целых чисел является 1, равна $6/\pi^2$.

Приведенная теорема используется для исследования последовательности целых чисел (например, номеров бит) на случайность. Если при обработке последовательности получена оценка для значения π, близкая к 3,14, то можно считать данную последовательность случайной.

Далее нам потребуется понятие кодового расстояния. Введем это понятие.

Кодовым расстоянием двух битовых последовательностей назовем количество бит, на которые отличаются последовательности.

Пусть имеется шифр A с длиной блока N бит, открытые тексты X_1 и X_2 длины N и ключ K. Открытый текст X_2 отличается на один бит от текста X_1, позиция этого бита случайна, обозначим ее $i', i' \in [1; N]$. При зашифровании текстов X_1 и X_2 получены зашифрованные тексты Y_1 и Y_2 соответственно. Сравнив тексты Y_1 и Y_2, получим $d(i')-$ кодовое расстояние этих текстов при i'-м отличном бите исходных текстов и $S = \{i_k\}, i \in [1; N]$ - последовательность номеров бит, на которые Y_1 и Y_2 отличаются. Мы излагаем здесь свой подход к характеристикам шифра.

Проверим S на случайность с помощью метода, основанного на теореме 2.1. Удалив из последовательности S элементы, которые делают ее неслучайной, получим последовательность $S' = \{i_j\}, i \in [1; N]$ независимых номеров бит. Количество элементов последовательности S' обозначим $d'(i')$ и назовем его **случайным кодовым расстоянием** выходных текстов при i'-м отличном бите исходных текстов.

Определение 2.1.

Пусть имеется шифр A с длиной блока N, и случайным кодовым расстоянием $d'(i')$, тогда экстремальностью шифра A по тексту при i'-м

отличном бите исходных текстов назовем величину $\varepsilon(i')$, которая вычисляется по формуле:

$$\varepsilon(i') = \frac{\left|\frac{N}{2} - d'(i')\right|}{\frac{N}{2}},$$

(2.1)

Очевидно, что нельзя судить об экстремальности шифра, проанализировав отклик в выходном тексте, при изменении всего одного бита исходного текста. Нужно получить значения $\varepsilon(i')$ для каждого бита исходного текста в пределах блока, каким-то образом обработать полученные данные (например, взять среднеарифметическое значение) и вывести финальное значение экстремальности шифра ε.

Из формулы (2.1) видно, что экстремальность ε может принимать значения от 0 до 1 включительно.

Чем ближе значение ε к нулю, тем более «хорошим» является алгоритм.

Действительно, если ε мало, то это значит, что при изменении всего одного бита в исходных данных в выходных данных меняется количество бит, близкое к половине. При этом номера этих бит образуют случайную последовательность. А это значит, что инвертирование даже одного бита в блоке исходных данных приведет к тому, что все биты в соответствующем блоке шифрованных данных с вероятностью близкой к 1/2 независимо друг от друга так же поменяют свое значение.

И наоборот, чем ближе значение ε к 1, тем более «плохим» является алгоритм.

Действительно, если значение ε близко к 1, то это значит, что малые изменения исходных данных приводят к малым изменениям в выходных данных. Что, согласно Столлингса (см. выше), позволит злоумышленнику сузить пространство ключей или область поиска открытого текста.

Введем понятие экстремальности по ключу.

Пусть имеется шифр A с длиной блока N бит, открытый текст X длины N и ключи K_1 и K_2. Ключ K_1 отличается на один бит от ключа K_2, позиция этого бита случайна, обозначим ее j'. При зашифровании текста X на ключах K_1 и K_2 получены зашифрованные тексты Y_1 и Y_2 соответственно. Сравнив тексты Y_1 и Y_2, получим $d(j')$ – кодовое расстояние этих текстов при j'-м отличном бите исходных текстов и $S = \{i_k\}, i \in [1; N]$ - последовательность номеров бит, на которые Y_1 и Y_2 отличаются.

Проверим S на случайность с помощью метода, основанного на теореме 2.1. Удалив из последовательности S элементы, которые делают ее неслучайной, получим последовательность $S' = \{i_j\}, i \in [1; N]$ независимых номеров бит. Количество элементов последовательности S' обозначим $d'(j')$ и назовем его **случайным кодовым расстоянием** выходных текстов при j'-

м отличном бите ключей.

Определение 2.2.

Пусть имеется шифр A с длиной блока N, и случайным кодовым расстоянием $d'(j')$, тогда экстремальностью шифра A по ключу при j'-м отличном бите ключей назовем величину $\varepsilon(j')$, которая вычисляется по формуле:

$$\varepsilon(j') = \frac{\left|\frac{N}{2} - d'(j')\right|}{\frac{N}{2}},$$

(2.2)

Нахождение и интерпретация значения экстремальности ε по ключу аналогично нахождению и интерпретации значения экстремальности ε по тексту.

Определение 2.3.

Пусть имеется шифр A с известными значениями ε_1 экстремальности по тексту и ε_2 экстремальности по ключу, тогда вектором экстремальности или общей экстремальностью назовем вектор:

$$\bar{\varepsilon}\left(\varepsilon_1, \varepsilon_2\right)$$

(2.3)

Вектор экстремальности $\bar{\varepsilon}\left(\varepsilon_1, \varepsilon_2\right)$ является наиболее тонкой характеристикой алгоритмов шифрования. Он объединяет в себе все вышеперечисленные характеристики и свойства.

Экстремальный шифр

Введем понятие экстремального шифра.

Определение 2.4.

Шифр A назовем экстремальным, если его вектор экстремальности является нулевым:

$$\bar{\varepsilon}\left(0,0\right)$$

(2.4)

Иными словами экстремальным назовем шифр, который при изменении одного любого бита исходных данных (в исходном тексте или ключе) инвертирует в выходном тексте N/2 случайных бит. Это означает, что все биты в выходном тексте поменяли свое значение с вероятность 1/2. Согласно А. Винокурову такой шифр невозможно вскрыть способом, менее затратным с точки зрения количества необходимых операций, чем полный перебор по множеству возможных значений ключа.

Очевидно, что кроме экстремальных шифров существуют и не

экстремальные шифры, которые не намного отличаются от экстремальных. В связи с этим будет полезно ввести понятие шифров, близких к экстремальным.

Рассмотрим

$$A = A_1 \cdot A_2 \cdot ... A_n \quad (2.5)$$

- произведение шифров такое, что $A_1 = (X_1, K_1, Y_1, f_1), \quad A_2 = (X_2, K_2, Y_2, f_2), \quad Y_1 = X_2, ...,$ то есть множество шифробозначений первого шифра является множеством шифрвеличин второго шифра и так далее до n-го шифра.

Под исходным текстом будем понимать текст X_1, под выходным текстом - Y_n. Под длиной блока N будем понимать длину блока шифра A_1.

Если в качестве сомножителей в (2.5) выступает один и тот же алгоритм, то такую композицию будем называть итерацией. Например, пусть $A = \underbrace{A_0 \cdot A_0 \cdot ... A_0}_{n \ раз} = A_0^n$, тогда шифр А – это n итераций шифра A_0.

Очевидно, что при изменении n в (2.5) может меняться и вектор экстремальности шифра А. Для каждого конкретного значения n обозначим вектор экстремальности $\bar{\varepsilon}_n(\varepsilon_{1n}, \varepsilon_{2n})$.

Определение 2.5.

Пусть имеется шифр А с вектором экстремальности $\bar{\varepsilon}_n(\varepsilon_{1n}, \varepsilon_{2n})$, шифр А назовем близким к экстремальному, если для него существует следующий предел:

$$\lim_{n \to \infty}(\sqrt{\varepsilon_{1n}^2 + \varepsilon_{2n}^2}) = 0 \quad (2.6)$$

Иначе говоря, шифр А близок к экстремальному, если, при стремлении числа сомножителей в (2.5) к бесконечности, вектор экстремальности шифра А стремится к нулю.

Введенная характеристика позволяет судить об эффективности шифра не анализируя его природу, ограничиваясь лишь исследованиями с помощью ЭВМ.

Рассмотрим пример.

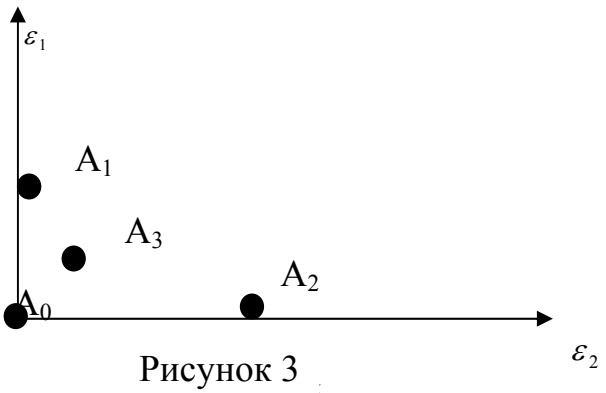


Рисунок 3

Пусть имеется шифр A_0, A_1, A_2, и A_3, вектора экстремальности которых отмечены на рисунке 3.

Согласно рисунку 3. по величине вектора экстремальности алгоритмы относительно друг друга располагаются следующим образом:

$A_0 < A_3 < A_1 < A_2$.

Что означает следующее распределение эффективности шифров относительно друг друга:

$A_2 < A_1 < A_3 < A_0$.

То есть для шифра A_1 вероятность того, что наиболее эффективным способом вскрытия шифра будет являться полный перебор по множеству возможных значений ключа, больше, нежели для шифра A_2 и меньше, чем для шифров A_3 и A_0.

Очевидно, что если стоимость использования шифров распределена следующим образом: $C_1 < C_3 < C_0 < C_2$, то использование шифра A_2 не представляется целесообразным. Решение о применении остальных шифров следует принимать, исходя из финансовых возможностей предприятия и требуемого уровня эффективности.

Кроме стоимости применения алгоритмов шифрования при выборе оптимального шифра для конкретной задачи можно рассматривать также и сложность применения этих алгоритмов, необходимы уровень образования сотрудников, работающих с алгоритмом и другие аспекты применения алгоритмов шифрования, которые зависят от корректного случая.

2.10.3 Процедура исследования

Для анализа экстремальности шифров DES, ECC и их композиций сотрудниками кафедры безопасности информационных технологий Сибирского государственного аэрокосмического университета была разработана процедура, показанная на рисунке 4 (процедура является универсальной и может быть применена для исследования экстремальности других алгоритмов шифрования).

Исследование происходит следующим образом. Исходный текст $И_0$ длины N шифруется исследуемым алгоритмом A, получается выходной текст B_0. Изменив в исходном тексте $И_0$ один бит (сначала первый, потом второй и так далее до N), получаем текст $И_{i'}$. Зашифровав $И_{i'}$ исследуемым алгоритмом A, получаем выходной текст $B_{i'}$. К текстам B_0 и $B_{i'}$ применяется функция сравнения $F_{срав}$, которая возвращает $d(i')$ – кодовое расстояние этих текстов при i'-м отличном бите исходных текстов и $S = \{i_k\}, i \in [1; N]$ - последовательность номеров бит, на которые B_0 и $B_{i'}$ отличаются. К полученным данным применяется функция $F_{случ,}$ которая возвращает последовательность $S' = \{i_j\}, i \in [1; N]$ независимых номеров бит и $d'(i')$ случайное кодовое расстояние выходных текстов при i'-м отличном бите исходных текстов. Далее по формуле (2.1) вычисляем экстремальность по

тексту $\varepsilon(i')$. Процедура вызывается N раз, каждый раз добавляя значение $\varepsilon(i')$ в совокупность значений $E = \{\varepsilon(i')\}, i' \in [1; N]$. После чего к совокупности $E = \{\varepsilon(i')\}, i' \in [1; N]$ применяется функция обработки $F_{обр}$, которая возвращает ε. Это и есть экстремальность по тексту алгоритма А.

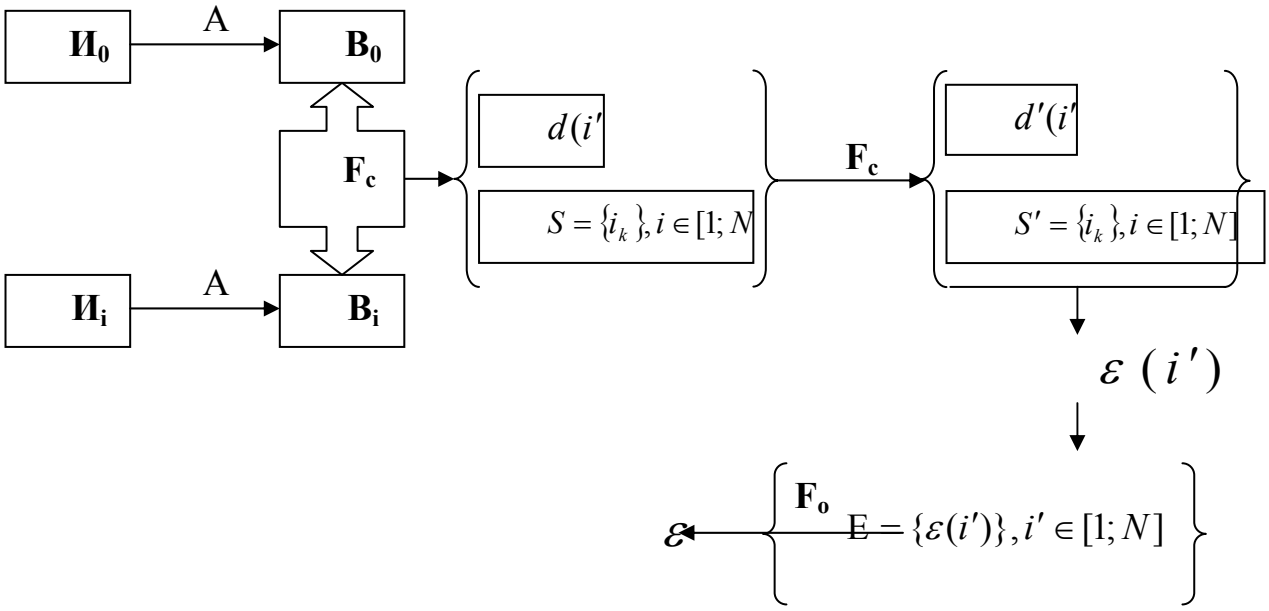


Рисунок 34 – Процедура исследования

Описанная процедура (с небольшими изменениями) может быть применена и к исследованию других алгоритмов.

Контрольные вопросы и задания

1. Перечислите основные недостатки алгоритма DES и предложите пути их устранения.
2. Выпишите в явном виде подстановку IP^{-1}. Найдите её разложение в произведение независимых циклов и вычислите её порядок.
3. Докажите, что все преобразования стандарта DES являются четными подстановками на пространстве V_{64}.
4. Докажите, что все S-блоки DES являются нелинейными преобразованиями из V_6 в V_8.
5. Докажите свойство дополнительности DES: $E_{\bar{k}}(\bar{x}) = \overline{E_k}(x)$, где черта означает взятие дополнительного вектора, т.е. $x + \bar{x} = 0$.

6. Какой из режимов: обратной связи по выходу (OFB) или шифрованной обратной связи (CFB) следует выбрать, если:
   a. нужно, чтобы возможные5 искажения битов при передаче данных не распространялись на последующие порции данных?

b. нужна большая надежность в отношении нарушений типа модификации потока данных?

7. Почему при передаче достаточно длинных сообщений режим ECB может не обеспечивать необходимый уровень защиты?
8. Обоснуйте необходимость защиты инициализующего вектора (IV) в режиме сцепления шифрованных блоков (CBC).
9. Алгоритм DES – это блочный шифр с размером блока 64 бит. Однако бывает необходимо каждый символ шифровать и сразу передавать адресату, не дожидаясь окончания шифрования остальной части сообщения. Предложите способ использовать DES для этого.
10. Перечислите отличия ГОСТ 28147-89 от DES.
11. Докажите, что всякое преобразование стандарта ГОСТ 28147-89 обратимо и является четной подстановкой на множестве всех 64-битовых слов.
12. Предложите алгоритмы генерации псевдослучайных последовательностей при помощи алгоритма ГОСТ 28147-89. Оцените величину их периода.
13. В чем заключается метод «опробования» (полного перебора, “brute force attack”)?
14. Применить метод линейного криптоанализа к трехтактовому DES.
15. Пусть h – блочно-итерационная функция и шаговая функция хэширования ǽ является обратимой (допускает вычисление y по известным ǽ(x, y) и x). Ставится задача: для заданного x найти другое слово x′, такое, что h(x)=h(x′). Для решения этой задачи применяется атака «встреча посередине» - модификация атаки «дней рождения». Дайте описание возможного варианта такой атаки.

16. Предложите возможные криптоатаки на:
    a. двухтактовый
    b. трехтактовый ГОСТ 28147-89

17. Почему алгоритм Blowfish не подходит для применения в смарт-картах?
18. Изобразите блок-схему алгоритма IDEA.
19. Почему операция умножения в IDEA выполняется по модулю $2^{16}+1$, а не по модулю 2^{16}?
20. Каким образом в алгоритме Rijndael удалось избежать появления полуслабых ключей?
21. Какому многочлену соответствует число 21 в шестнадцатеричной записи?
22. За счет чего обеспечивается высокая скорость работы алгоритма Rijndael?
23. Шифрованный текст C=10, открытый ключ e=5, N=35. Найдите открытый текст.

24. Открытый ключ e=31, N=3599. Найдите личный ключ этого пользователя.
25. Предположим, что пользователь RSA в качестве модуля N выбрал большое простое число. Показать, что в этом случае дешифровать текст легко.
26. Рассмотрим систему RSA с модулем N. Целое число x, где $1 \le x \le N-1$, назовем неподвижной точкой, если оно и в зашифрованном виде тоже x. Показать, что если x – неподвижная точка, то и N-x тоже неподвижная точка.

27. Пусть E — эллиптическая кривая, определенная над C, уравнение (1) которой имеет коэффициенты $a, b \in R$; тогда точки E с вещественными координатами образуют подгруппу. Описать все возможные виды структуры такой подгруппы комплексной кривой E (которая как группа изоморфна произведению окружности на себя). Приведите пример для каждой из них.

28. Привести пример эллиптической кривой над R, имеющей в точности 2 точки порядка 2, и пример кривой, имеющей в точности 4 точки порядка 2.

29. Пусть P — точка на эллиптической кривой над R. Предположим, что P не есть точка в бесконечности. Найти геометрическое условие, эквивалентное тому, что P — точка порядка а) 2; б) 3; в) 4.

30. Каждая из следующих точек имеет конечный прядок на данной эллиптической кривой над Q. Найти в каждом случае порядок P.

а) P=(0,16) на $y^2 = x^3 + 256$

б) P=$(\frac{1}{2}, \frac{1}{2})$ на $y^2 = x^3 + \frac{1}{4}x$

в) P=(3,8) на $y^2 = x^3 - 43x + 166$

г) P=(0,0) на $y^2 + y = x^3 - x^2$ (уравнение можно привести к виду (1) заменой переменных $y \to y - \frac{1}{2}$, $x \to x + \frac{1}{3}$)

31. Доказать, что число F_q-точек на каждой из следующих эллиптических кривых равно q+1:

а) $y^2 = x^3 - x$, когда $q \equiv 3 \pmod 4$;

в) $y^2 = x^3 - 1$, когда $q \equiv 2 \pmod 3$ (q нечетно);

г) $y^2 + y = x^3$, когда $q \equiv 2 \pmod 3$ (здесь q может быть четным);

3. СРЕДСТВА КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ

Применение криптографических методов защиты информации основано на изложенной выше теории, но, как и любая практическая задача, такое применение должно базироваться на собственных принципах, рекомендациях и ограничениях. В настоящий момент практическая криптография распространяется все шире вслед за развитием электронного документооборота, широкодоступных сетевых служб, обмена информацией через сети передачи данных общего пользования. Кроме того, существует ряд направлений, в которых применению средств криптографической защиты информации практически нет альтернативы. Согласно [7], такими направлениями в настоящий момент являются:

- защищенная электронная почта;
- обеспечение безопасности электронных платежей (электронный документооборот);
- виртуальные частные сети.

Кроме того, можно выделить и другие задачи, решаемые средствами криптографической защиты информации:

- создание и использование носителей ключевой информации [1];
- шифрование данных, хранимых в базе данных или в электронном виде на различных носителях информации;
- электронная цифровая подпись и связанные с ней виды шифрования, в частности, проверка авторства;
- криптографические интерфейсы;
- задачи идентификации и аутентификации и т.п.

Таким образом, средствами криптографической защиты информации решается множество задач, непосредственно связанных с обеспечением информационной безопасности любой системы, базы данных или сети передачи информации.

3.1 Виды средств криптографической защиты информации

Определение 1. Средства криптографической защиты информации (СКЗИ) – совокупность аппаратных и(или) программных компонентов, обеспечивающих возможность шифрования информации с использованием одного или нескольких криптографических методов защиты.

Как видно из определения, существует ряд возможных реализаций методов криптографической защиты информации:

- программные, представляющие собой реализацию одного или нескольких криптоалгоритмов на языке программирования высокого или низкого уровня, в виде модулей, отдельных библиотек или выделенных

программ с функцией криптографической защиты;

- аппаратные, реализующие криптоалгоритмы или их отдельные участки в микросхемах, процессорах и специализированных блоках (системы встроенной защиты) и аппаратных модулях (системы наложенной защиты), совмещенные со средствами вычислительной техники или встраиваемые в автоматизированные системы;

- программно-аппаратные, представляющие собой комплексы, состоящие из взаимосвязанных программной и аппаратной части с функциями криптографической защиты.

При организации систем засекреченной связи и защищенных автоматизированных систем требуется учитывать некоторые особенности размещения средств криптографической защиты информации, причем нет принципиальных различий между программными, программно-аппаратными и аппаратными средствами. Кроме некоторых преимуществ использования различных реализаций криптографической защиты, описанных далее по тексту, дающих возможность варьирования места использования в информационно-телекоммуникационной системе, можно выделить несколько узловых точек (рис. 37), в которых возможно их внедрение.

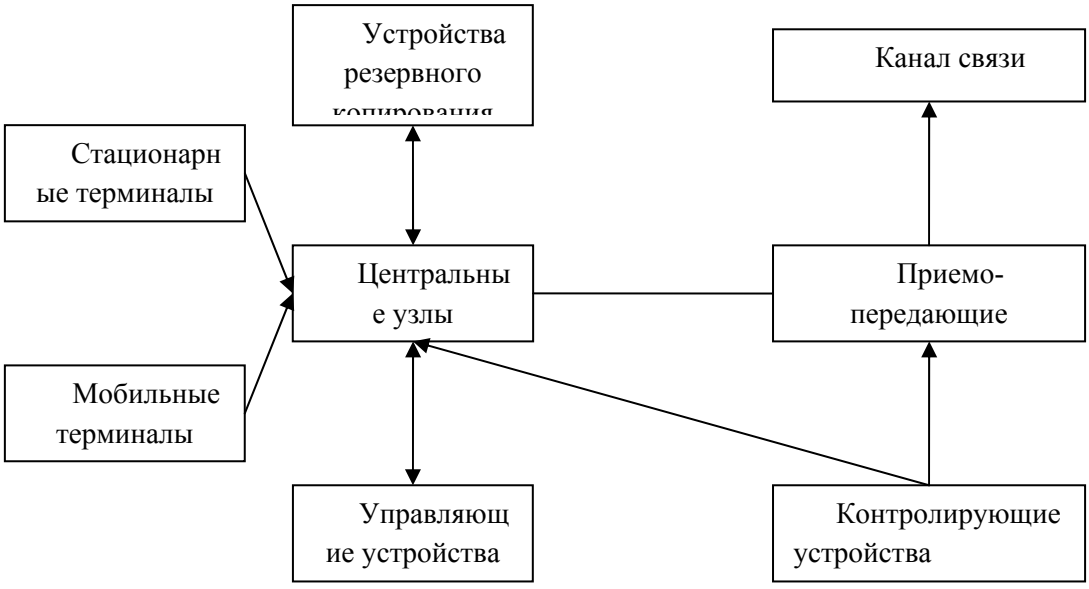


Рис. 37. Места внедрения СКЗИ в информационно-телекоммуникационной системе

Итак, рассмотрим перечисленные возможности подробнее:

Системы встроенной защиты, сопрягаемые с программными и аппаратными устройствами автоматизированных систем, могут использоваться в следующих случаях:

- защита терминалов, стационарных или мобильных, а также устройств ввода-вывода информации и периферийных устройств;

- защита центральных узлов, а также соединений с внешними терминалами;

- защита управляющих и контролирующих устройств, причем с

обязательной гарантией высокого уровня надежности;

- обеспечение безопасности хранения информации, в том числе и резервных копий.

Системы наложенной защиты, реализуемые в виде отдельных блоков, могут использоваться кроме того и в некоторых других случаях, как то:

- для организации независимого защищенного канала передачи информации;
- обеспечения «прозрачного» шифрования передаваемой по каналу связи информации;
- применения как устройств защиты приемо-передающих компонентов информационно-телекоммуникационных систем.

Особенностью использования СКЗИ в информационно-телекоммуникационных системах является наличие нескольких (обычно трех-четырех) выделенных режимов работы. Такими режимами обычно являются:

- абонентское шифрование;
- сквозное шифрование;
- канальное шифрование;
- цифровая подпись;
- идентификация/аутентификация.

Подробно все режимы применения средств криптографической защиты информации описаны ниже, но можно остановиться на некоторых особенностях. В частности, стоимость средства напрямую зависит от количества используемых режимов, причем комбинированных систем на рынке представлено достаточно, но в основном в программно-аппаратных или аппаратных реализациях. Программные реализации часто специализируются на нескольких режимах.

Абонентские терминалы могут защищаться как комбинированными системами, так и специализированными. Для экономии средств часто используется сочетание «программный клиент - программно-аппаратный сервер», что упрощает организацию криптографической защиты, но снижает показатели надежности системы обеспечения информационной безопасности.

Сервер целесообразно защищать комбинированными системами с максимально возможным количеством режимов работы, что позволяет обеспечивать безопасную работу все сетевых служб. Кроме того, для центрального узла телекоммуникационной сети предпочтительной может быть аппаратная реализация СКЗИ, реализующая предобработку и шифрование/расшифровку информации.

Аппаратные средства

Обратимся к аппаратной технологии реализации криптографической защиты информации. Начнем с ряда важных определений.

Определение 2. Аппаратное средство криптографической защиты информации – специализированный блок, компонент средства вычислительной техники или отдельное устройство, выполняющее шифрование информации.

Собственно шифрование, согласно определению [1], представляет собой криптографическое преобразование данных для получения шифртекста (закрытого текста). Дополняя это определение, можно заметить на основании первого раздела данной книги, что шифрование в аппаратных средствах криптографической защиты также требует взаимодействия различных специализированных компонентов автоматизированной системы или средства вычислительной техники для реализации криптоалгоритмов.

Определение 3. Устройство криптографической защиты данных (УКЗД) – аппаратное СКЗИ, выполняющее также дополнительные функции по защите информации, например, защищающий от НСД [2].

Кроме того, аппаратное средство криптографической защиты содержит ряд дополнительных блоков, которые не требуются в программной реализации:

- блок управления криптографическими ключами;
- генератор случайных чисел;
- память, постоянная и оперативная;
- блок синхронизации времени;
- устройство хранения и проверки контрольных сумм и хэш-значений.

Также возможно использование специализированных шифропроцессоров, используемых для вычислений и отдельных блоков идентификации, аутентификации и авторизации (проверки и генерации электронной цифровой подписи). Таким образом, полнофункциональное (с реализацией всех перечисленных функций) аппаратное СКЗИ может являться даже специализированным компьютером, реализованным в виде аппаратного блока, внешнего или внутреннего, и связанного с центральным процессором собственными каналами передачи данных (например, системной шиной).

В статье [2] выделены несколько функций, которые также могут быть реализованы аппаратно:

- контроль доступа;
- выдача идентификационных ключей;
- проверка целостности данных.

Именно такое СКЗИ с дополнительными функциями в дальнейшем будем называть устройством криптографической защиты данных.

Структура аппаратного СКЗИ

Поскольку, в отличие от программных технологий реализации криптоалгоритмов, которые будут рассмотрены далее, аппаратные реализации требуют физического исполнения на плате, при проектировании таких средств соблюдаются определенные требования и ограничения, в целом соответствующие следующей базовой схеме (рис. 38).

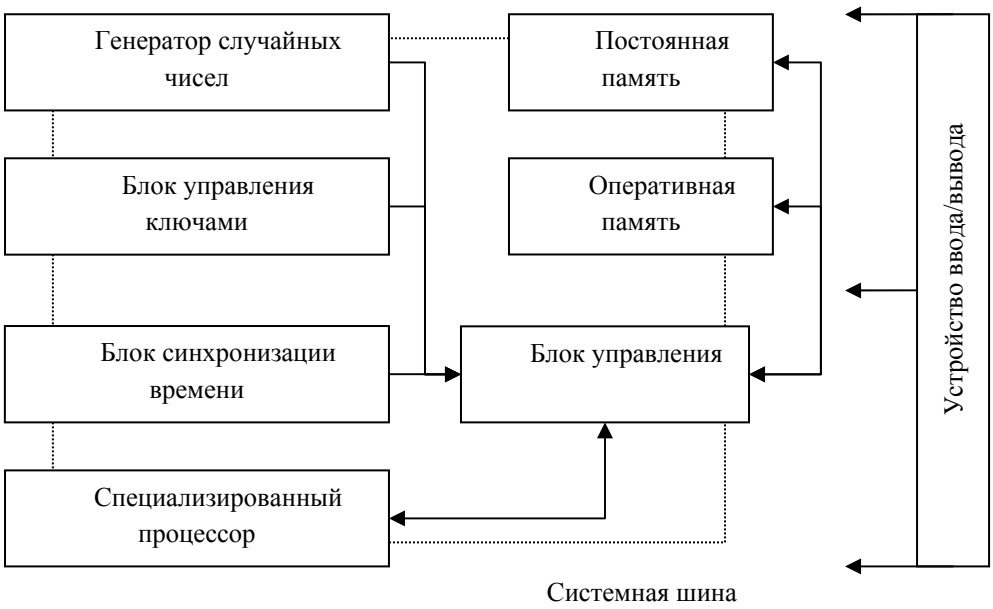


Рис. 38. Базовая схема аппаратного СКЗИ

Блок управления выполняет координацию функций различных компонентов СКЗИ, коммутацию устройств и внешних потоков, управление данными, передаваемыми по системной шине.

Память аппаратного средства подразделяется на два блока:

- оперативная, хранящая программное обеспечение СКЗИ, а также все программируемые компоненты, изменяемые значения и прочее;
- постоянная, содержащая набор команд, реализующих защитные функции и загрузку программного обеспечения СКЗИ.

Кроме того, в статье [2] и ряде других источников [1,3,4] выделяются два дополнительных блока памяти:

- память журнала;
- память хранения ключей.

Специализированные процессоры применяются в аппаратных СКЗИ для увеличение скорости вычислений и исполнения большинства криптографических операций. Они представляют собой программируемые блоки или блоки с жестко установленным набором команд.

Генератор случайных чисел используется для получения псевдослучайных последовательностей, отвечающих требованиям криптоалгоритма, используемого СКЗИ. Используемые в аппаратной реализации криптографической защиты генераторы случайных чисел основаны на различных физических процессах.

Блок управления ключами выполняет прием, обработку и выдачу ключевой информации, а также, если это необходимо, функции идентификации и аутентификации.

Блок синхронизации времени требуется в операциях контроля функционирования самого СКЗИ, при синхронизации потока данных, для расчета и согласования операций.

Устройство ввода-вывода обеспечивает обработку сигналов,

поступающих от внешних устройств. Для устройств ввода-вывода информации в аппаратных СКЗИ является принципиально важным обеспечение возможности взаимодействия со следующими устройствами:

- системная шина компьютерной системы, либо приемно-передающее устройство линии передачи информации, что актуально для организации сквозного, или «прозрачного», шифрования;
- устройства хранения ключевой информации, применяемые в задачах идентификации/аутентификации, а также контроля функционирования аппаратного СКЗИ;
- контроллеры жестких дисков и устройства резервного копирования информации, что позволяет организовать безопасное хранение данных и их резервных копий.

Следовательно, аппаратное средство криптографической защиты информации представляет собой сложное многофункциональное устройство, принципиально схожее по организации с устройствами программно-аппаратной защиты данных, рассмотренных подробно в пособии [5], но имеющее ряд специфических функций и обеспечивающее реализацию криптографического преобразования исходной информации.

Наконец, рассмотрим кратко достоинства аппаратных СКЗИ, основываясь на информации, приведенной в статье [13]. Итак, в число основных достоинств аппаратных шифраторов входят следующие:

- гарантия неизменности алгоритма шифрования;
- наличие аппаратного датчика случайных чисел, используемого при создании криптографических ключей;
- возможность прямой (минуя системную шину компьютера) загрузки ключей шифрования в специализированный процессор аппаратного СКЗИ с персональных идентификаторов - носителей типа смарт-карт и "таблеток" Touch Memory (TM);
- хранение ключей шифрования не в ОЗУ компьютера (как в случае с программной реализацией), а в памяти шифропроцессора;
- идентификация и аутентификация пользователя до загрузки операционной системы;
- аппаратная реализация позволяет добиться высокой скорости шифрования данных.

Кроме того, можно отметить ряд достоинств аппаратной реализации, не перечисленных в данной статье, а именно:

- надежность, позволяющая использовать средство криптографической защиты в критичных по надежности узлах автоматизированных систем;
- возможность реализации отдельным блоком, что зачастую позволяет более гибко строить топологию защищенной автоматизированной системы;
- исключение возможности программного повреждения ключей шифрования, что дает гарантию стабильности системы в целом.

Программные средства

Программные средства шифрования представляют собой реализацию криптографического алгоритма на высокоуровневом или низкоуровневом языке программирования. Обычно функционирование таких средств криптографической защиты требует выполнения ряда вычислительных операций стандартными аппаратными средствами компьютерной системы.

Технология реализации криптоалгоритмов программными средствами имеет ряд особенностей и отличий. Кроме очевидного факта, что при использовании программного СКЗИ применение аппаратного расширения не является обязательным условием, можно отметить и ряд других особенностей:

- необходимость дополнительного контроля за качеством функционирования, поскольку в общем случае работу программного СКЗИ нарушить легче, чем его аппаратного аналога;
- возможность контроля ошибок в закрытом тексте при шифровании путем внедрения избыточности;
- необходимость обеспечения надежного хранения ключей, что решается за счет создания мастер-ключа (ключа для шифрования файла, содержащего базу данных ключей) и других технологий, не требующихся обычно при использовании аппаратного СКЗИ;
- возможность масштабирования и дополнения СКЗИ новыми программными блоками и модификациями используемых;
- принципиальная возможность использования программного СКЗИ с открытым кодом, что допускается при шифровании информации в частных целях и облегчает общую схему защиты.

Таким образом, программное СКЗИ отличает способность использования в распределенных и глобальных информационно-телекоммуникационных системах, более гибкая реализация, способность масштабирования и высокая мобильность.

Можно выделить следующие функции программных средств криптографической защиты:

- идентификация/аутентификация пользователей;
- обеспечение встроенной производителем криптографической защиты программных и операционных систем;
- генерация псевдослучайных последовательностей;
- шифрование данных на диске;
- «прозрачное» шифрование данных;
- абонентское шифрование данных;
- формирование и проверка ключей, цифровых подписей, защита от копирования программного кода;
- обеспечение безопасной передачи секретного ключа при инициализации СКЗИ, в том числе и аппаратных.

Итак, можно отметить, что базовые функции аппаратного СКЗИ дополняются возможностями встраивания программных средств в

программное обеспечение, используемое для хранения, обработки и передачи данных в информационно-телекоммуникационных системах.

Структура программного СКЗИ

Программные технологии реализации криптоалгоритмов имеют в целом схожие базовые элементы, в целом соответствующие следующей схеме (рис. 39).

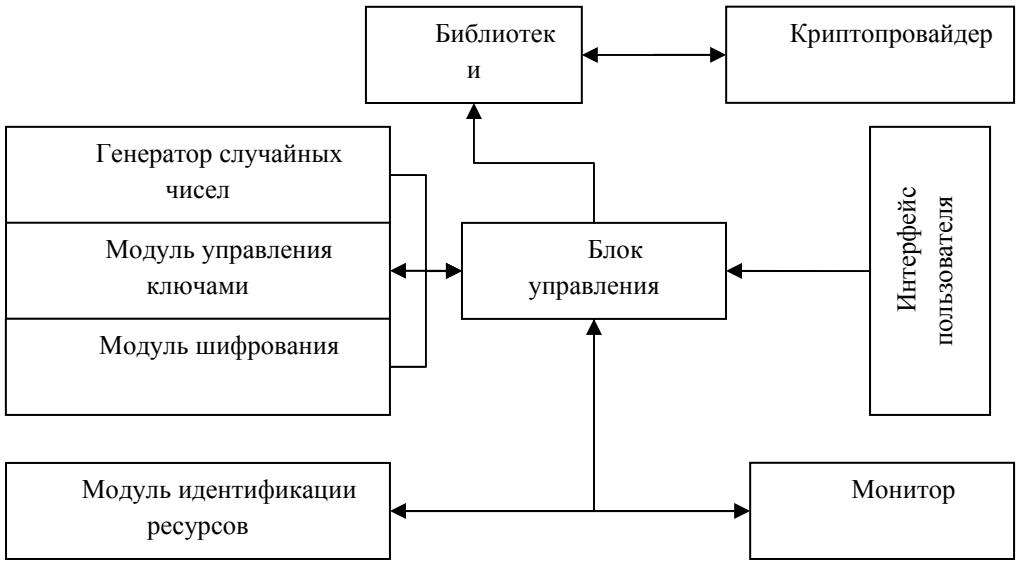


Рис. 39. Схема программного СКЗИ

Блок управления выполняет координацию функций различных компонентов СКЗИ, коммутацию модулей и внешних потоков информации, а также внешних команд.

Модуль управления ключами выполняет задачи обработки ключевой информации, индентификации/аутентификации пользователей. Дополнительно используется идентификация устройств, которая может использоваться как элемент защитной подсистемы.

Генератор случайных чисел, используемый в программных СКЗИ, также реализуется программно. В них применяются различные алгоритмы генерации псевдослучайных битовых последовательностей.

Библиотеки программных средств криптографической защиты применяются при решении прикладных задач и настройке СКЗИ в зависимости от условий эксплуатации.

Компонентом взаимодействия с пользователем является интерфейс, который предоставляет возможности управления программной реализацией алгоритма, контроля функционирования.

Кроме того, программное средство криптографической защиты информации вне зависимости от реализованного криптоалгоритма имеет ряд особенностей шифрования [1]:

- файлы, зашифрованные программной СКЗИ, могут храниться на других носителях автоматизированной системы;

- размер блока в блочном алгоритме может превышать размер сегмента

файла, в результате размер файла увеличивается;

- скорость шифрования программными средствами может быть ниже, чем в аппаратной реализации, за счет загрузки центрального процессора криптографическими вычислениями;

- работа с ключами усложняется отсутствием аппаратной идентификации пользователей.

Основным выводом может стать то, что программные средства предназначены для решения прикладных задач криптографической защиты информации, в которых требуется гибкость и масштабируемость системы. Реализация криптографических алгоритмов такого типа не позволяет гарантированно обеспечивать защиту от атак на программный код, но достаточно и других аспектов, в частности, экономический и эксплуатационный, которые могут обосновать их использование в системах защиты информации любой сложности.

Криптографический комплекс «Игла-П»

Данное программное средство построено по модульному принципу и представляет собой сетевой драйвер для ОС Windows NT, который заменяет типовой программный драйвер NDIS, с комплектом установочных и конфигурационных программ. Скорость шифрования – до 10 Мбит/с. На практике такая скорость работы достигается редко, поскольку характерным недостатком любого программного СКЗИ, как уже было отмечено выше, является зависимость от ресурсов центрального процессора.

Рекомендуется использовать данное программное СКЗИ для защиты вновь подключаемых к локальной вычислительной сети и предназначенных для передачи защищаемой информации рабочих станций. Можно отметить, что использование СКЗИ «Игла-П» представляет собой достаточно практичный способ защиты трафика в локальной сети, а также возможность взаимодействия с многофункциональным программно-аппаратным комплексом «Шип».

Пример: программный эмулятор «Криптон»

Программный эмулятор «Криптон» является частью широкой линейки продуктов фирмы «Анкад», позволяющей подбирать звенья системы криптографической защиты по необходимым функциям. Целью разработки именно этого звена, по утверждению производителя, стало обеспечение гибкости использования криптографических средств. Таким образом, программный эмулятор решает задачу обеспечения криптографической защиты абонентских терминалов, а также подключаемых мобильных устройств.

Схема функционирования программного эмулятора позволяет использовать ключевые носители различных типов, что частично устраняет недостаток систем идентификации. Уязвимость возникает в процессе передачи ключа по системной шине, но перехват именно в этой зоне

маловероятен.

Отметим, что использование программного «Криптона» может дополняться его аппаратным аналогом. Все специальное программное обеспечение, используемое программным СКЗИ, в данном случае поддерживается и другими продуктами, в состав которых могут входить и аппаратные, и программные модули.

Производитель рекомендует использовать данное программное обеспечение в прикладных задачах криптографии, в которых требуется совместимость разных средств криптографической защиты. Среди них можно выделить наиболее важные:

- межсетевое экранирование;
- шифрование трафика в гетерогенных сетях;
- шифрование информации на диске.

Вывод: программные средства криптографической защиты используются обычно в качестве дополнения к аппаратным, защищающим наиболее критичные узлы защищенной информационной системы, но хочется заметить, что речь идет о сертифицированных средствах, не включая открыто распространяемые продукты, которые не входят в область рассмотрения данного пособия.

Программно-аппаратные комплексы криптографической защиты информации являют собой наиболее сложную и эффективную разновидность средств обеспечения информационной безопасности с использованием криптоалгоритмов.

Под программно-аппаратными средствами криптографической защиты информации будем понимать специальным образом организованные комплексы, содержащие взаимосвязанные программные и аппаратные блоки и реализующие следующий набор функций:

- идентификацию и аутентификацию пользователей;
- криптографическое преобразование данных;
- обеспечение целостности информации.

Основной целью применения программно-аппаратных средств обеспечения информационной безопасности является усиление или замещение существующих функций защиты компьютерных систем для обеспечения требуемого уровня защищенности [5].

Кроме приведенного определения, можно добавить важное замечание, состоящее в возможности реализации комплексного метода защиты информации. Это метод криптографической защиты как целостности, так и конфиденциальности информации. При использовании данного метода основную роль играют шифрование, электронная цифровая подпись и криптографические ключи.

Следовательно, программно-аппаратное СКЗИ является одним из важнейших компонентов комплексного обеспечения информационной безопасности в компьютерных системах.

Структура программно-аппаратного СКЗИ

Как уже было отмечено выше, в состав программно-аппаратного СКЗИ (как системы, состоящей из программных и аппаратных компонентов) должны входить следующие базовые модули:

- блок шифрования (для программного шифрования используются ассиметричные криптосистемы (см., например, гл.7), для аппаратного – симметричные), программный, аппаратный или комбинированный;
- блок электронной цифровой подписи;
- блок управления ключами;
- модуль идентификации/аутентификации;
- модуль управления с внешним интерфейсом;
- модуль контроля функционирования.

Структура же соответствует приведенной на рис. 40 базовой схеме, хотя нужно учитывать, что при проектировании программно-аппаратных средств постоянно происходит поиск новых решений, повышающих эффективность и надежность защиты.

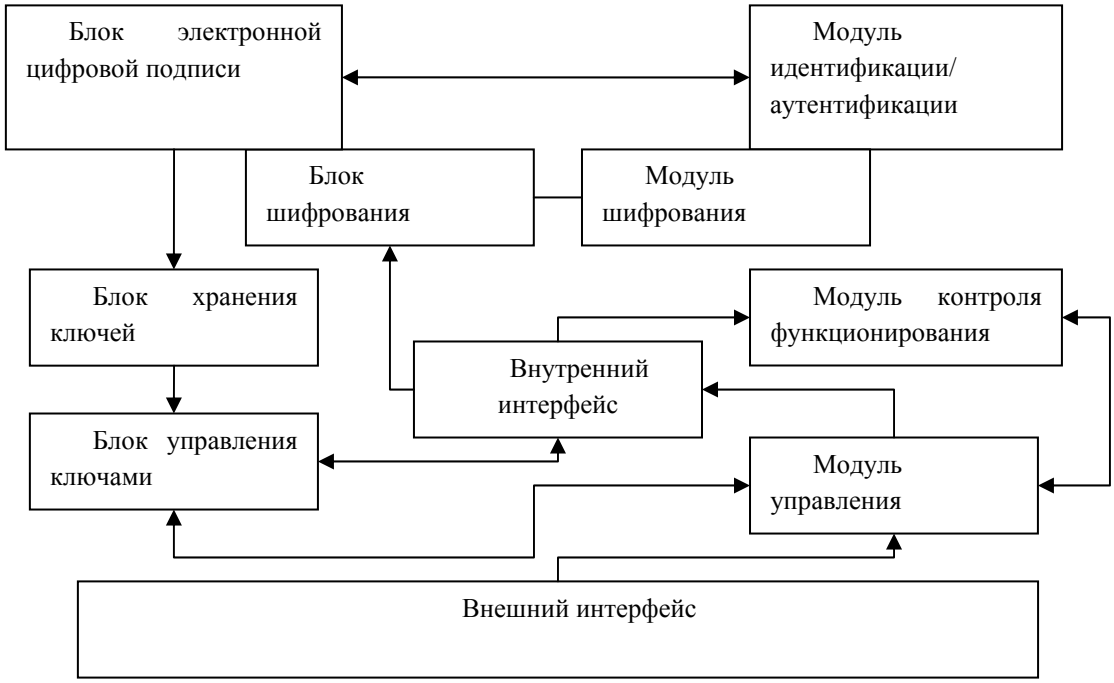


Рис. 40. Базовая схема программно-аппаратного СКЗИ

Отметим, что модули программной защиты в данном случае связывает с программной частью специализированный внутренний интерфейс, то есть взаимодействие происходит по собственным каналам СКЗИ (без использования системной шины), и, кроме того, с соблюдением внутренней (встроенной по умолчанию) политики безопасности.

СКЗИ «Шип».СКЗИ «Шип» представляет собой программно-аппаратный комплекс криптографической защиты информации, основной функцией которого является комплексная защита компьютерной сети.

Наиболее важными функциями СКЗИ являются:

- возможность создания защищенной виртуальной частной сети;
- шифрование IP-пакетов
- организация фильтрации и маршрутизации;
- шифрование и проверка целостности данных с использованием имитовставки;
- одностороннюю аутентификацию узлов защищенной сети на основе имитовставки;
- управление ключевой системой защищенной сети.

Основными компонентами СКЗИ «Шип» являются программно-аппаратный комплекс «Шип» и центр управления ключевой системой.

Программно-аппаратный комплекс «Шип» используется при создании защищенной виртуальной частной сети. Его основными достоинствами является реализация режимов прозрачного и сквозного шифрования, скрытие трафика в защищенной сети, а также проверка целостности данных при их получении.

Центр управления ключевой системой является реализацией блока управления ключами (рис. 40) в виде отдельного компонента комплекса и выполняет следующие функции:

- генерация и управление списком абонентов, имеющих право доступа (справочников соответствия по терминологии разработчика);

журналирование внештатных ситуаций;

периодическая (плановая) смена ключей шифрования, используемых в системе;

оповещение о компрометации ключей.

Программно-аппаратный комплекс «Шип» поставляется в двух вариантах по производительности и по оснащению портами:

вариант №1 – «Шип-1»: производительность до 15 Мбит/с (3000 пакетов/с), порты: 2 асинхронных порта V.24, 2 порта Ethernet (Fast-Ethernet/FDDI) – для сетей пакетной коммутации с возможностью модемного подключения;

вариант №2 – «ШИП-2»: Производительность около 8 Мбит/с (3000 пакетов/с) при обмене информацией по схеме «Ethernet (вход) – Ethernet (выход)» и около 2 Мбит/с (750 пакетов/с) при обмене по интерфейсу V.35 и протоколу Frame Relay. Порты: 2 асинхронных порта V.24, 2 порта Ethernet (Fast-Ethernet/FDDI), 1 синхронный порт V.24/V35 (X.25/Frame Relay/PPP/Cisco HDLC) – для сетей канальной коммутации либо подключения по модему, с использованием дополнительных аппаратных устройств защиты информации.

Использование программно-аппаратных СКЗИ целесообразно при организации комплексной защиты информации. Являясь мощным инструментом обеспечения информационной безопасности, особенно в аспекте защиты компьютерной информации, СКЗИ такого типа могут резко повышать надежность и эффективность системы защиты информации.

3.2. ПРИМЕНЕНИЕ СРЕДСТВ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ

Наиболее важным компонентом реализации любой криптографической системы является криптографический ключ. Под криптографическим ключом будем понимать далее конкретизированное значение набора параметров криптоалгоритма, обеспечивающее выбор одного преобразования из совокупности возможных для данного алгоритма преобразований. Таким образом, криптографический ключ – критичное звено любой криптосистемы. Исключением являются открытые ключи ассиметричных криптосистем, по параметрам которых теоретически невозможно определить необходимое значение парного ключа, предназначенного для расшифровки.

При использовании криптографических ключей и шифровании информации необходимо соблюдать ряд принципов. Основным является недопущение компроментации (разглашения или возможности разглашения) секретных частей криптографической системы. Кроме того, используются принципы комплексности (в данном случае касается использования различных криптосистем для повышения надежности шифрования) и прозрачности (непричастности пользователей к формированию и управлению ключами).

3.2.1. Принципы использования ключей шифрования в СКЗИ

Главной задачей при эксплуатации средств криптографической защиты информации, как программных, так и аппаратных, а также программно-аппаратных комплексов, неизбежно становится организация передачи криптографических ключей. Решением может стать (при соблюдении описанных выше принципов) использование ассиметричных криптосистем для шифрования ключа, дальнейшая пересылка его по открытому каналу связи (или каналу связи с возможной компроментацией ключа). Схема такой реализации, отвечающая принципу комплексности, показана на рис. 41.

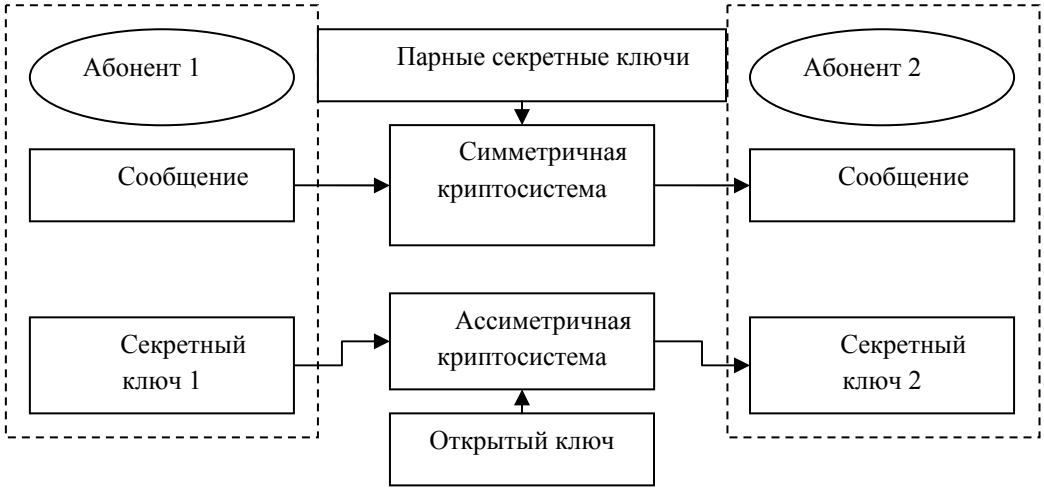


Рис. 41. Реализация передачи криптографического ключа

Организация симметричного шифрования, как сеансового, так и постоянного, на основе переданного таким образом ключа имеет высокий уровень защиты от компроментации, что ценно для систем передачи данных, не полностью контролируемых защитными средствами.

3.2.2. Виды шифрования с использованием средств криптографической защиты информации

Абонентское шифрование

Определение 4. Абонентское шифрование - шифрование информации для последующей передачи по сети определенным пользователям (абонентам) [6].

Из определения следует, что при шифровании данных таким способом требуется определение как минимум двух участников обмена информацией – отправителя и получателя (абонента). Кроме того, предполагается наличие пары ключей, предназначенных для шифрования/расшифровки информации.

Ключи в данной схеме подразделяют на файловые и долговременные [6]:

- файловые ключи предназначены для шифрования собственно информации. Обычно такие ключи генерируются случайным образом для каждого шифруемого объекта.
- долговременные ключи используются для шифрования и передачи файловых. Часто долговременными являются открытые ключи ассиметричных криптосистем.

Структура систем абонентского шифрования предполагает несколько компонентов, необходимых для качественного выполнения ее основной функции. Такими компонентами являются:

- абонентский модуль;
- системный журнал;
- устройство ввода/вывода ключевой информации;
- модуль управления (регистрация, контроль доступа, запрет доступа

абонентов).

Рассмотрим на примере СКЗИ «Криптон» процесс передачи документов в системе абонентского шифрования.

При подготовке документов к передаче автоматически осуществляется:

- запрос из базы данных открытых ключей зарегистрированных абонентов, которым направляются документы;
- электронная подпись передаваемых документов;
- сжатие документов в один файл;
- генерация сеансового ключа;
- шифрование файла с документами на сеансовом ключе;
- вычисление парно-связных ключей (на основе секретного ключа отправителя и открытых ключей получателей) и шифрование на них сеансового ключа.

После приема автоматически выполняются обратные действия:

вычисление парно-связного ключа (на основе секретного ключа получателя и открытого ключа отправителя с автоматической проверкой сертификата) и расшифрование сеансового ключа;

- расшифрование файла с помощью полученного сеансового ключа;
- разархивирование документов;
- проверка электронной подписи полученных документов.

Абонентское шифрование не обеспечивает полной и гарантированной защиты передаваемой информации. Такая проблема возникает вследствие взаимодействия СКЗИ и программного обеспечения абонентской рабочей станции, в общем случае незащищенного от внешнего и внутреннего деструктивного воздействия. В целом абонентское шифрование целесообразно применять при организации передачи информации по фиксированному каналу (или для дополнительной защиты данных в виртуальной частной сети).

“Прозрачное” шифрование информации

Определение 5. Прозрачное шифрование – это скрытое от пользователя и происходящее автоматически криптографическое преобразование данных. Основное применение скрытого шифрования – архивирование защищаемой информации и шифрование логических дисков.

Существует два основных вида средств криптографической защиты, обеспечивающих выполнение функций программного шифрования. Это программные средства, реализованные в виде компонентов операционной системы (драйверов), и программно-аппаратные и аппаратные средства, выполняющие прозрачное шифрование в рамках одного или нескольких режимов работы.

Недостатком программной реализации в данном случае, очевидно, является зависимость от операционной системы, как правило ненадежной. Тогда как аппаратные и программно-аппаратные комплексы более сложны в эксплуатации и накладывают ограничения на состав и качество исполнения

информационной системы.

Основной реализацией, которая может удовлетворять указанным выше принципам использования средств криптографической защиты информации, является криптомаршрутизатор.

Криптомаршрутизатор представляет собой сетевую плату со встроенными функциями криптографического преобразования данных. Выгодным такое решение может быть в любых системах передачи данных, требующих прозрачного шифрования трафика. В случае же прозрачного шифрования данных на диске более выгодным становится использование функций шифрования, встроенных в аппаратные СКЗИ.

Локальное шифрование

В системах хранения информации, базах данных или при обработке больших объемов электронных данных часто возникает необходимость создания зашифрованного архива. В данном случае применяется локальное шифрование, то есть шифрование информации для хранения в защищенном виде.

Принципиальным отличием локального шифрования от любых других способов применения средств криптографической защиты информации является отсутствие проблемы распределения ключей, что значительно упрощает общую схему организации такой защиты (рис. 42).

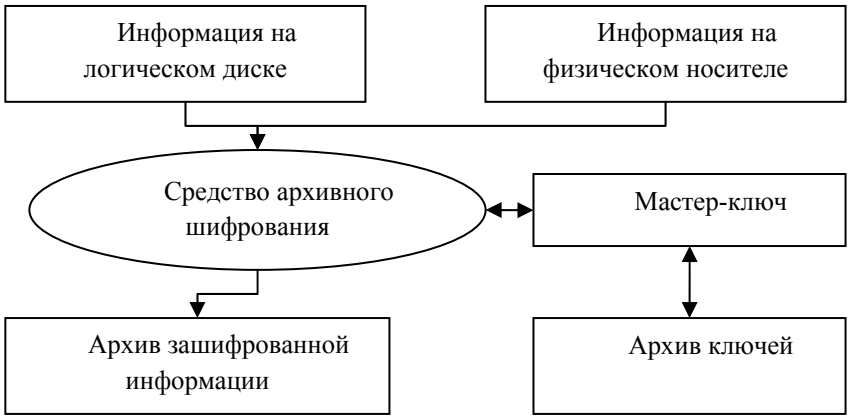


Рис. 42. Схема локального шифрования

Действительно, как видно из приведенной схемы, при криптографическом преобразовании архивных данных главной задачей становится не распределение, а хранение ключей.

Локальное шифрование, в отличие от других способов применения СКЗИ, не подразумевает обособленного использования. Обычно такой способ шифрования применяют в комплексе с одним или несколькими альтернативными.

3.2.3. Цифровые подписи

Цифровая подпись для сообщения является числом, зависящим от

самого сообщения и от некоторого секретного, известного только подписывающему субъекту, ключа. При этом предполагается, что она должна быть легко проверяемой и что осуществить проверку подписи должен иметь возможность каждый без получения доступа к секретному ключу. При возникновении спорной ситуации, связанной с отказом подписывающего от факта подписи им некоторого сообщения либо с попыткой подделки подписи, третья сторона должна иметь возможность разрешить спор.

Цифровая подпись позволяет решить следующие три задачи:

— осуществить аутентификацию источника сообщения,

— установить целостность сообщения,

— обеспечить невозможность отказа от факта подписи конкретного сообщения.

Использование термина "подпись" в данном контексте оправдано тем, что цифровая подпись имеет много общего с обычной собственноручной подписью на бумажном документе. Собственноручная подпись также решает три перечисленные задачи, однако между обычной и цифровой подписями имеются существенные различия. Сведем основные различия между обычной и цифровой подписями в таблицу 25 [15].

Таблица 25. Отличия собственноручной и цифровой подписи

Собственноручная подпись	*Цифровая подпись*
не зависит от подписываемого текста, всегда одинакова	зависит от подписываемого текста, практически всегда разная
неразрывно связана с подписывающим лицом, однозначно определяется его психофизическими свойствами, не может быть утеряна	определяется секретным ключом, принадлежащим подписывающему лицу, может быть утеряна владельцем
неотделима от носителя (бумаги), поэтому отдельно подписывается каждый экземпляр документа	легко отделима от документа, поэтому верна для всех его копий
не требует для реализации дополнительных механизмов	требует дополнительных механизмов, реализующих алгоритмы ее вычисления и проверки
не требует создания поддерживающей инфраструктуры	требует создания доверенной инфраструктуры сертификатов открытых ключей

Для реализации схемы цифровой подписи необходимы два алгоритма:

— алгоритм вычисления цифровой подписи;

— алгоритм ее проверки.

Главные требования к этим алгоритмам заключаются в исключении возможности получения подписи без использования секретного ключа и

гарантировании возможности проверки подписи без знания какой-либо секретной информации.

Надежность схемы цифровой подписи определяется сложностью следующих трех задач:

— *подделки подписи,* то есть нахождения значения подписи под заданным документом лицом, не являющимся владельцем секретного ключа;

— *создания подписанного сообщения,* то есть нахождения хотя бы одного сообщения с правильным значением подписи;

— *подмены сообщения,* то есть подбора двух различных сообщений с одинаковыми значениями подписи.

Имеется множество различных схем цифровой подписи, обеспечивающих тот или иной уровень стойкости. Основные подходы к их построению будут рассмотрены ниже.

Принципиальной сложностью, возникающей при использовании цифровой подписи на практике, является проблема создания *инфраструктуры открытых ключей.* Дело в том, что для алгоритма проверки подписи необходима дополнительная открытая информация, связанная с обеспечением возможности открытой проверки подписи и зависящая от секретного ключа автора подписи. Эту информацию можно назвать открытым ключом цифровой подписи. Для исключения возможности подделки этой информации (открытого ключа) лицами, которые хотят выступить от лица законного владельца подписи (секретного ключа), создается инфраструктура, состоящая из центров сертификации открытых ключей и обеспечивающая возможность своевременного подтверждения достоверности принадлежности данной открытой информации заявленному владельцу и обнаружения подлога.

Создание сертификационных центров с технической точки зрения не представляет большой сложности. Они строятся во многом аналогично центрам сертификации, которые используются в криптографических системах с открытыми ключами. Однако с юридической точки зрения здесь имеется множество проблем. Дело в том, что в случае возникновения споров, связанных с отказом от авторства или подделки подписи, такие центры должны нести юридическую ответственность за достоверность выдаваемых сертификатов. В частности, они должны возмещать понесенные убытки в случае конфликтных ситуаций, когда алгоритм проверки подписи подтверждает ее правильность. В связи с этим сложилась практика заключения договоров между участниками информационного взаимодействия с применением цифровых подписей. В таком договоре должно быть четко указано:

— кто должен нести ответственность в случае, если подписанные сделки не состоятся;

— кто должен нести ответственность в случае, если система окажется ненадежной и будет взломана, то есть будет выявлен факт подделки секретного ключа;

— какова ответственность уполномоченного по сертификатам в случае, если открытый ключ будет сфальсифицирован;

— какова ответственность владельца секретного ключа в случае его утраты;

— кто несет ответственность за плохую реализацию системы в случае повреждения или разглашения секретного ключа;

— каков порядок разрешения споров и т. п.

Поскольку данные проблемы носят юридический, а не технический характер, то для их разрешения нужен юридически правильно заключенный договор, оформленный стандартным образом на бумаге.

В настоящее время предложено несколько принципиально различных подходов к созданию схем цифровой подписи. Их можно разделить на три группы:

1) схемы на основе систем шифрования с открытыми ключами;

2) схемы со специально разработанными алгоритмами вычисления и проверки подписи;

3) схемы на основе симметричных систем шифрования. Рассмотрим их более подробно.

Цифровые подписи на основе шифрсистем с открытыми ключами

Идея использования систем шифрования с открытыми ключами для построения систем цифровой подписи как бы заложена в постановке задачи. Действительно, пусть имеется пара преобразований *(E, D),* первое из которых зависит от открытого ключа, а второе — от секретного. Для того чтобы вычислить цифровую подпись *S* для сообщения, владелец секретного ключа может применить к сообщению *M* второе преобразование *D:* $S = D(M)$. В таком случае вычислить подпись может только владелец секретного ключа, в то время как проверить равенство $E(S) = M$ может каждый. Основными требованиями к преобразованиям *E* и *D* являются:

— выполнение равенства $M = E(D(M))$ для всех сообщений *M;*

— невозможность вычисления значения *D(M)* для заданного сообщения *M* без знания секретного ключа.

Отличительной особенностью предложенного способа построения цифровой подписи является возможность отказаться от передачи самого подписываемого сообщения *M,* так как его можно восстановить по значению подписи. В связи с этим подобные системы называют *схемами цифровой подписи с восстановлением текста.*

Заметим, что если при передаче сообщение дополнительно шифруется с помощью асимметричного шифра, то пара преобразований *(E, D),* используемая в схеме цифровой подписи, должна отличаться от той, которая используется для шифрования сообщений. В противном случае появляется возможность передачи в качестве шифрованных ранее подписанных сообщений. При этом более целесообразно шифровать подписанные данные, чем делать наоборот, то есть подписывать шифрованные данные, поскольку в

первом случае противник получит только шифртекст, а во втором — и открытый, и шифрованный тексты.

Очевидно, что рассмотренная схема цифровой подписи на основе пары преобразований *(E, D)* удовлетворяет требованию невозможности подделки, в то время как требование невозможности создания подписанного сообщения не выполнено: для любого значения S каждый может вычислить значение $M = E(S)$ и тем самым получить подписанное сообщение.

Требование невозможности подмены сообщения заведомо выполняется, так как преобразование *E* взаимно однозначно.

Для защиты от создания злоумышленником подписанного сообщения можно применить некоторое взаимно-однозначное отображение $R : M \mapsto \tilde{M}$, вносящее избыточность в представление исходного сообщения, например, путем увеличения

его длины, а затем уже вычислять подпись $S = D(\tilde{M})$. В этом случае злоумышленник, подбирая *S* и вычисляя значения $\tilde{M} = E(S)$, будет сталкиваться с проблемой отыскания таких значений $\tilde{M}$ для которых существует прообраз *M*. Если отображение *R* выбрано таким, что число возможных образов $\tilde{M}$ значительно меньше числа всех возможных последовательностей той же длины, то задача создания подписанного сообщения будет сложной.

Другой подход к построению схем цифровых подписей на основе систем шифрования с открытым ключом состоит в использовании бесключевых хеш-функций. Для заданного сообщения *M* сначала вычисляется значение хеш-функции *h(M),* а затем уже значение подписи $S = D(h(M))$. Ясно, что в таком случае по значению подписи уже нельзя восстановить сообщение. Поэтому подписи необходимо передавать вместе с сообщениями. Такие подписи получили название *цифровых подписей с дополнением.* Заметим, что системы подписи, построенные с использованием бесключевых хеш-функций, заведомо удовлетворяют всем требованиям, предъявляемым к цифровым подписям. Например, невозможно создание сообщения с известным значением подписи, поскольку бесключевая хеш-функция должна быть однонаправленной.

В качестве системы шифрования с открытыми ключами можно использовать, например, систему RSA.

Цифровая подпись Фиата — Шамира

Рассмотрим подход к построению схемы цифровой подписи, основанной на сложности задач факторизации больших целых чисел и извлечения квадратного корня в кольце вычетов. Идея построения схемы принадлежит А. Фиату и А. Шамиру. Приведем одну из модификаций схемы, предложенную ими совместно с У. Фейджем. В ней реализуется цифровая подпись с дополнением.

Пусть *h* — некоторая хеш-функция, преобразующая исходное сообщение в битовую строку длины *m.* Выберем различные простые числа *p*

и q и положим $n = pq$. В качестве секретного ключа каждый абонент должен сгенерировать m различных случайных чисел $a_1,a_2,...,a_m \in Z_n$. Открытым ключом объявляется набор чисел $b_1,b_2,...,b_m \in Z_n$, где

$b_i = (a_i^{-1})^2 \bmod n$, $i = 1,...,m$.

Алгоритм вычисления цифровой подписи для сообщения M состоит в выполнении следующих действий:

1. Выбрать случайное число r, $1 \le r \le n-1$;
2. Вычислить $u = r^2 \bmod n$; 3. Вычислить $h(M,u) = s = (s_1,s_2,...,s_m)$;
4. Вычислить $t = r\prod_{i=1}^{m} a_i^{s_i} \bmod n$;
5. Подписью для сообщения M положить пару (s, t).

Алгоритм проверки подписи состоит в выполнении следующих действий:

1. По открытому ключу $b_1, b_2,...,b_m \bmod n$ и значению t вычислить

$$w = t^2 \prod_{i=1}^{m} b_i^{s_i} \bmod n$$

2. Вычислить $h(M,w) = s'$;
3. Проверить равенство $s = s'$. Достоинствами описанной схемы являются возможность выработки цифровых подписей для нескольких различных сообщений с использованием одного секретного ключа, а также сравнительная простота алгоритмов вычисления и проверки подписи. Например, для схемы цифровой подписи, основанной на алгоритме RSA, соответствующие алгоритмы требуют выполнения значительно большего числа умножений. Попытка компрометации этой схемы сталкивается с необходимостью решения сложной задачи нахождения квадратных корней по модулю n.

Недостатком схемы является большая длина ключа, которая определяется числом m. Если двоичная запись числа n содержит l знаков, то длина секретного ключа составляет ml бит, а открытого ключа — $(m+1)l$ бит. При этом необходимо учитывать, что для обеспечения достаточной стойкости данной схемы цифровой подписи числа l и m должны иметь в своей двоичной записи несколько сотен бит.

Подписи EIGamal

Чтобы подписать сообщение M, сначала выбирается случайное число k, взаимно простое с p-1. Затем вычисляется

$a = g^k \bmod p$

и с помощью расширенного алгоритма Эвклида находится b в следующем уравнении:

$M = (xa + kb) \bmod (p - 1)$

Подписью является пара чисел: a и b. Случайное значение k должно храниться в секрете. Для проверки подписи нужно убедиться, что

$y^a a^b \bmod p = g^M \bmod p$

Каждая подпись или шифрование EIGamal требует нового значения k, и это значение должно быть выбрано случайным образом. Если когда-нибудь злоумышленник раскроет k, используемое отправителем, он сможет раскрыть закрытый ключ отправителя *x*. Если злоумышленник когда-нибудь сможет получить два сообщения, подписанные или зашифрованные с помощью одного и того же *k,* то он сможет раскрыть *x,* даже не зная значение *k.*

	Открытый ключ:
p	простое число (может быть общим для группы пользователей)
g	$<p$ (может быть общим для группы пользователей)
y	$=g^x \bmod p$
	Закрытый ключ:
x	$<p$
	Подпись:
k	выбирается случайным образом, взаимно простое с p-1
a	(подпись) $=g^k \bmod p$
b	(подпись), такое что $M = (xa + kb) \bmod (p - 1)$
	Проверка:
Подпись считается правильной, если $y^a a^b \bmod p = g^M \bmod p$	

Рис.43 Подписи EIGamal [17]

Например, выберем p = 11 и g = 2, а закрытый ключ x = 8. Вычислим

$y = g^x \ mod p = 2^8 \bmod 11=3$

Открытым ключом являются $y = 3,$ g = 2 и p = 11. Чтобы подписать M= 5, сначала выберем случайное число $k=9$. Убеждаемся, что НОД(9, 10)= 1. Вычисляем

$a = g^k \ mod \ p = 2^9 \bmod 11=6$

и с помощью расширенного алгоритма Эвклида находим b:

$M= (xa + kb) \bmod (p - 1)$

$5 = (8*6 + 9*b) \bmod 10$

Решение: b = 3, а подпись представляет собой пару: a = 6 и b = 3.

Для проверки подписи убедимся, что

$y^a a^b \ mod \ p = g^M \ mod \ p$

$3^6 6^3 \bmod 11=2^5 \bmod 11$

Томас Бет (Thomas Beth) изобрел вариант схемы EIGamal, подходящий для доказательства идентичности. Существуют варианты для проверки подлинности пароля и для обмена ключами. И еще тысячи и тысячи других.

DSA

DSA представляет собой вариант алгоритма подписи ElGamal. Алгоритм использует следующие параметры:

p = простое число длиной L битов, где L принимает значение, кратное

64, в диапазоне от 512 до 1024. (В первоначальном стандарте размер p был фиксирован и равен 512 битам. Это вызвало множество критических замечаний.)

q = 160-битовой простое число - множитель p-\.

$g = h^{(p-1)/q} \bmod p$, где h - любое число, меньшее $p-1$, для которого $h^{(p-1)/q} \bmod p$ больше 1.

x = число, меньшее q.

$y = g^x \bmod p$.

В алгоритме также используется однонаправленная хэш-функция: $H(m)$. Стандарт определяет использование SHA.

Первые три параметра, p, q и g, открыты и могут быть общими для пользователей сети. Закрытым ключом является x, а открытым - y. Чтобы подписать сообщение, m:

(1)Отправитель генерирует случайное число k, меньшее q

(2)Отправитель генерирует

$r = (g^k \bmod p) \bmod q$

$s = (k^{-1} (H(m) + xr)) \bmod q$

Подписью служат параметры r и s, она посылает их получателю.

(3) Получатель проверяет подпись, вычисляя

$w = s^{-1} \bmod q$

$u_1 = (H(m) * w) \bmod q$

$u_2 = (rw) \bmod q$

$v = ((g^{u1} * y^{u2}) \bmod p) \bmod q$

Если $v = r$, то подпись правильна.

Алгоритм цифровой подписи ГОСТ Р 34.10-94

Алгоритм основан на DSA, и использует следующие параметры

p = простое число, длина которого либо между 509 и 512 битами, либо между 1020 и 1024 битами. q = простое число - множитель $p-1$, длиной от 254 до 256 битов. a = любое число, меньшее $p-1$, для которого $a^q \bmod p = 1$.

x = число, меньшее q. $y = a^x \bmod p$.

Этот алгоритм также использует однонаправленную хэш-функцию: $H(x)$. Стандарт определяет использование хэш-функции ГОСТ Р 34.11-94, основанной на симметричном алгоритме ГОСТ 28147-89.

Первые три параметра, p, q и a, открыты и могут использоваться совместно пользователями сети. Закрытым ключом служит x, а открытым - y. Чтобы подписать сообщение m

(1) Отправитель генерирует случайное число k, меньшее q

(2) Отправитель генерирует

$I = (a^* \bmod p) \bmod q$

$s = (ct + k(H(m))) \bmod q$

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

$s = (xr + k(H(m))) \bmod q$

Если $H(m)$ mod q =0, то значение хэш-функции устанавливается равным 1. Если r =0, то выберите другое значение k и начните снова. Подписью служат два числа: r mod 2^{256} и s mod 2^{256}, отправитель посылает их получателю.

(3) Получатель проверяет подпись, вычисляя v = $H(m)^{q-2}$ mod q

$z_1 = (sv)$ mod q

$z_2 = ((q-r)*v)$ mod q

$u = ((a^{Z1} * y^{Z2})$ mod p) mod q

Если $u = r$, то подпись правильна.

Различие между этой схемой и DSA в том, что в DSA $s = (k^{-1} (H(m) + xr))$ mod q, что дает другое уравнение проверки.

Деревья цифровых подписей

Ральф Меркл предложил систему цифровых подписей, основанную на криптографии с секретным ключом, создающей бесконечное количество одноразовых подписей, используя древовидную структуру. Основной идеей этой схемы является поместить корень дерева в некий открытый файл, удостоверяя его таким образом. Корень подписывает одно сообщение и удостоверяет подузлы дерева. Каждый из этих узлов подписывает одно сообщение и удостоверяет свои подузлы, и так далее.

Подпись документа с помощью криптографии с открытыми ключами

Существуют алгоритмы с открытыми ключами, которые можно использовать для цифровых подписей. В некоторых алгоритмах - примером является RSA - для шифрования может быть использован или открытый, или закрытый ключ. Зашифруйте документ своим закрытым ключом, и вы получите надежную цифровую подпись. В других случаях - примером является DSA - для цифровых подписей используется отдельный алгоритм, который невозможно использовать для шифрования. Эта идея впервые была изобретена Диффи и Хеллманом и в дальнейшем была расширена и углублена в других работах. Основной протокол прост:

(1) Отправитель шифрует документ своим закрытым ключом, таким образом, подписывая его.

(2) Отправитель посылает подписанный документ получателю.

(3) Получатель расшифровывает документ, используя открытый ключ отправителя, таким образом проверяя подпись.

Такая подпись соответствует всем требованиям:

1. Эта подпись достоверна. Когда получатель расшифровывает сообщение с помощью открытого ключа отправителя, происходит проверка авторства сообщения.

2. Эта подпись неподдельна. Только отправитель знает свой закрытый ключ.

3. Эту подпись нельзя использовать повторно. Подпись является функцией документа и не может быть перенесена на другой документ.

4. Подписанный документ нельзя изменить. После любого изменения документа подпись не сможет больше подтверждаться открытым ключом отправителя.

5. От подписи невозможно отказаться.

Метки времени

На самом деле, при определенных условиях получатель может повторно использовать документ и подпись совместно.

Поэтому в цифровые подписи часто включают метки времени. Дата и время подписания документа добавляются к документу и подписываются вместе со всем содержанием сообщения. Таким образом, использовать документа повторно становится невозможно.

Подпись документа с помощью криптографии с открытыми ключами и однонаправленных хэш-функций

На практике алгоритмы с открытыми ключами часто недостаточно эффективны для подписи больших документов. Для экономии времени протоколы цифровой подписи нередко используют вместе с однонаправленными хэш-функциями. Отправитель подписывает не документ, а значение хэш-функции для данного документа. В этом протоколе однонаправленная хэш-функция и алгоритм цифровой подписи согласовываются заранее .

(1) Отправитель получает значение однонаправленной хэш-функции для документа.

(2) Отправитель шифрует это значение своим закрытым ключом, таким образом подписывая документ.

(3) Отправитель посылает получателю документ и подписанное значение хэш-функции.

(4) Получатель получает значение однонаправленной хэш-функции для документа, присланного отправителем. Затем, используя алгоритм цифровой подписи, он расшифровывает подписанное значение хэш-функции с помощью открытого ключа отправителя. Если подписанное значение хэш-функции совпадает с рассчитанным, подпись правильна.

Скорость заметно возрастает и, так как вероятность получить для двух различных документов одинаковое 160-битное значение хэш-функции составляет только один шанс из 2^{160}, можно безопасно приравнять подпись значения хэш-функции и подпись документа. Должна использоваться только однонаправленная хэш-функция, иначе создать разные документы с одним и тем же значением хэш-функции нетрудно, и подпись одного документа приведет к ошибочной подписи сразу многих документов.

У протокола есть и другие выгоды. Во-первых, подпись может быть отделена от документа. Во-вторых, значительно уменьшаются требования к объему памяти получателя, в котором хранятся документы и подписи. Архивная система может использовать этот протокол для подтверждения

существования документов, не храня их содержания. В центральной базе данных могут храниться лишь значения хэш-функции для файлов. Вовсе не нужно просматривать файлы, пользователи помещают свои значения хэш-функции в базу данных, а база данных хранит эти значения, помечая их временем получения документа. Если в будущем возникнет какое-нибудь разногласие по поводу автора и времени создания документа, база данных сможет разрешить его при помощи хранящегося в ней значения хэш-функции.

3.3. ИНФРАСТРУКТУРА ОТКРЫТЫХ КЛЮЧЕЙ

От общего описания применения средств защиты перейдем к конкретным технологиям их использования.

Инфраструктура открытых ключей это совокупность аппаратного и программного обеспечения, персонала и организационных мер, необходимых для создания, управления, хранения, распределения и отзыва сертификатов открытых ключей (PKI «Internet X.509 Public Key Infrastructure PKIX Roadmap»).

Таким образом, инфраструктура открытых ключей включает следующие компоненты [7]:

- центры сертификации, которые отвечают за издание и отзыв сертификатов открытых ключей;
- центры регистрации, которые отвечают за привязку открытых ключей и их владельцев перед изданием сертификата;
- владельцы сертификатов открытых ключей, это конечные пользователи, субъекты системы, для которых издаются сертификаты, и которые их используют;
- клиенты отвечают за проверку ЭЦП и цепочек сертификатов, начиная с открытого ключа доверенного корневого центра сертификации;
- хранилища – это специальные базы данных, представляющие собой каталоги, сохраняющие и публикующие для общего доступа сертификаты, списки отозванных сертификатов компонентов инфраструктуры и списки отозванных сертификатов пользователей.

3.3.1. Сертификаты

Создание цифровой подписи позволило решить проблему *сертификации открытых ключей.* Она заключается в том, что перед тем как использовать открытый ключ некоторого абонента для отправки ему конфиденциального сообщения, отправитель должен быть уверен, что открытый ключ действительно принадлежит этому абоненту. Открытые ключи необходимо очень тщательно обезопасить, в том смысле, что если сервер, на котором они хранятся, не обеспечивает их целостность и аутентичность, то злоумышленник имеет возможность, подменив открытый ключ одного из абонентов, выступать от его имени. Поэтому для защиты открытых ключей создаются специальные *центры сертификации,* которые играют роль доверенной третьей стороны и заверяют открытые ключи каждого из абонентов своими цифровыми подписями.

Сертификат представляет собой набор данных, заверенный цифровой подписью центра, и включающий открытый ключ и список дополнительных атрибутов, принадлежащих абоненту. К таким атрибутам относятся: имена пользователя и центра сертификации, номер сертификата, время действия сертификата, предназначение открытого ключа (цифровая подпись,

шифрование) и т. д.

Международный стандарт ISO X.509 определяет общую структуру сертификатов открытых ключей и протоколы их использования для аутентификации в распределенных системах.

3.3.2. Центры сертификации

Центр сертификации предназначен для регистрации абонентов, изготовления сертификатов открытых ключей, хранения изготовленных сертификатов, поддержания в актуальном состоянии справочника действующих сертификатов и выпуска списка досрочно отозванных сертификатов.

Для сетей с большим числом абонентов создается несколько центров сертификации. Центры сертификации объединяются в древовидную структуру, в корне которой находится главный центр сертификации, который выдает сертификаты подчиненным ему отраслевым центрам, тем самым обеспечивая доверие к открытым ключам этих центров. Каждый центр вышестоящего уровня аналогичным образом делегирует право выпуска сертификатов подчиненным ему центрам. В результате доверие сертификату открытого ключа каждого центра основано на заверении его сертификата ключом вышестоящего центра. Сертификаты главного центра подписывает сам главный центр.

Зная иерархию и подчиненность друг другу центров сертификации, можно всегда точно установить, является ли абонент владельцем данного открытого ключа.

Основная трудность при создании центров сертификации заключается в их юридическом статусе и потенциальных финансовых возможностях по выплате компенсаций за ущерб, понесенный в результате невыполнения подписанных цифровыми подписями с использованием сертификатов, выданных этим центром, договоров и контрактов, сорванных по причине отказов от цифровых подписей или их подделки.

Общая политика безопасности информационной системы определяет и устанавливает общие положения корпоративной политики в области информационной безопасности, например, такие как правила и порядок использования продуктов, содержащих криптографические функции.

Взаимодействие между всеми компонентами приведено на рис. 44 [7]:

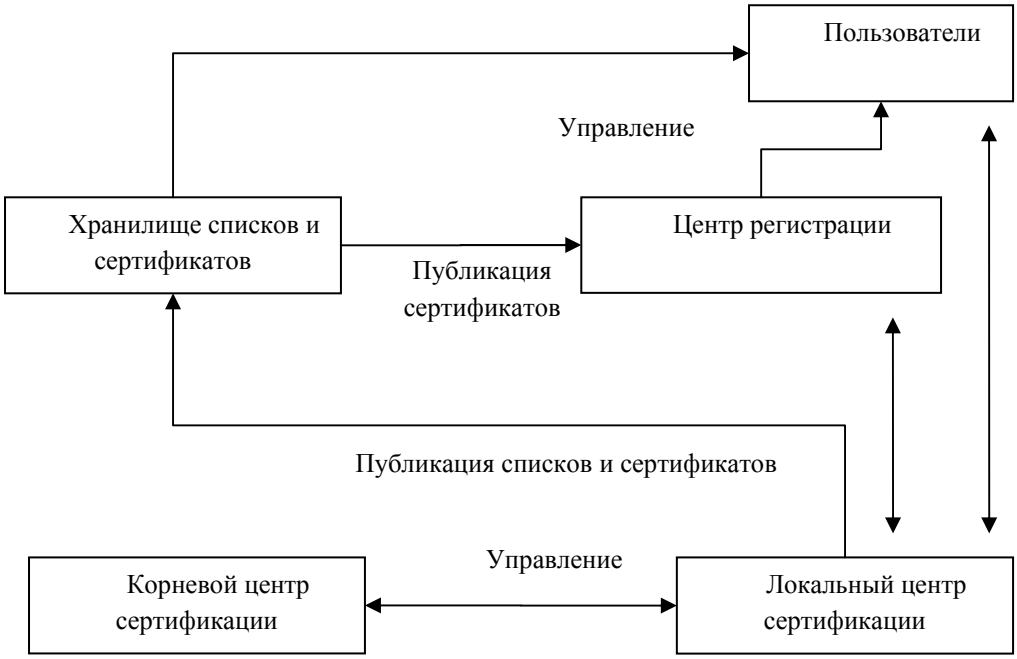


Рис.44. Инфраструктура открытых ключей

Отметим, что использование инфраструктуры открытых ключей требует использования как ассиметричных, так и симметричных криптосистем, а также организации всего комплекса компонентов. Очевидно также, что без использования подтверждения открытого ключа вся система неэффективна.

Пример: удостоверяющий центр "КриптоПРО"

Обеспечение возможности реализации и управления инфраструктурой открытых ключей предоставляет ряд программно-аппаратных комплексов криптографической защиты, к которым относится и удостоверяющий центр «КриптоПРО». Основой данного решения является СКЗИ «КриптоПро», представляющее собой многофункциональный комплекс с гибкими и масштабируемыми возможностями применения.

Удостоверяющий центр позволяет полностью реализовать все типы криптографических средств защиты, нуждающихся в инфраструктуре открытых ключей. К числу основных функций комплекса относятся:

- регистрация пользователей;
- изготовление сертификатов открытых ключей;
- генерация ключей и управление личными сертификатами;
- ведение реестра сертификатов открытых ключей;
- управление сертификатами открытых ключей.

При этом удостоверяющий центр «КриптоПро» обеспечивает [7]:

- выполнение процедуры генерации личных закрытых и открытых ключей ЭЦП и шифрования на рабочем месте пользователя;
- формирование запроса на сертификат нового открытого ключа на рабочем месте пользователя;
- выполнение процедуры регистрации электронных запросов от

пользователей на сертификаты открытых ключей в Центре регистрации УЦ;

- формирование электронных сертификатов открытых ключей пользователей в соответствии с рекомендациями X.509 версии 3 и RFC 2459, позволяющими с помощью криптографических методов (ЭЦП) централизованно заверять соответствие открытого ключа и атрибутов определенному пользователю;

- формирование и доставку зарегистрированным пользователям списка отозванных сертификатов открытых ключей пользователей.

Основными структурными элементами удостоверяющего центра, в соответствии с базовой структурой, показанной на рис. 44, являются:

- центр сертификации;
- центр регистрации;
- АРМ администратора (компонент управления);
- пользовательский интерфейс и средства взаимодействия;
- программный интерфейс взаимодействия с удостоверяющим центром.

Рассмотрим в соответствии с источником [7] функции каждого из компонентов системы.

Центр сертификации составляет основу инфраструктуры как компонент, обеспечивающий генерацию сертификатов открытых ключей. Кроме того, центр сертификации выполняет формирование списков отозванных сертификатов, а также совмещает функции традиционного центра сертификации и хранилища списков и сертификатов, то есть представляет собой компонент, хранящий эталоны сертификатов и списков. Взаимодействие с центром сертификации удостоверяющего центра «КриптоПРО», таким образом, возможно только через центр регистрации, причем с использованием защищенного логического канала.

Центр регистрации выполняет обработку и хранение регистрационных данных, в соответствии с базовой схемой инфраструктуры открытых ключей, а также организует и координирует взаимодействие пользователей и центра сертификации. Являясь центральным узлом системы, центр регистрации концентрирует на себе пользовательские запросы и выдачу криптографических ключей и обеспечивает функционирование пользовательских интерфейсов и средств взаимодействия.

Как управляющий компонент удостоверяющего центра «КриптоПРО» используется специальным образом организованный программный блок, АРМ администратора центра регистрации. Основной функцией АРМ администратора является выполнение организационно-технических мероприятий, связанных с регистрацией пользователей, формированием служебных ключей и сертификатов пользователей и управление центром регистрации.

Пользовательский интерфейс выполняет функции управления ключами пользователей, сертификатами и служебной информацией. С помощью средств взаимодействия, размещенных на сервере центра регистрации, пользовательский интерфейс полностью обеспечивает работу пользователя

инфраструктуры.

Кроме того, центр регистрации имеет возможность программируемого доступа, реализуемого через программный интерфейс внешних приложений. Осуществление взаимодействия безопасно и требует обязательного использования сертификата, причем допускается и разграничение доступа к функциям программного интерфейса и средствам управления.

Разработчики рекомендуют использование удостоверяющего центра при реализации системы электронного документооборота или любой другой системы, требующей применения открытых ключей ассиметричных криптосистем. Ключевой задачей описанного выше решения является создание основы инфраструктуры открытых ключей, подтверждение и генерация сертификатов, а также поддержка СКЗИ, реализующих электронную цифровую подпись.

ВИРТУАЛЬНЫЕ ЧАСТНЫЕ СЕТИ

Виртуальная частная сеть (VPN) – это логический канал передачи данных, сконфигурированный на основе существующих физических каналов и обеспечивающий реализацию технологии туннелирования [5]. Существующие физические каналы внешней информационной среды используются для передачи данных между компьютерами защищенной виртуальной сети, но дополнительно для защиты передаваемой информации используются криптографические преобразования, фильтрация пакетов и идентификация передаваемой информации служебными данными, что позволяет организовать устойчивый канал передачи, не зависящий от внешней информационной среды.

3.4.1. Классификация виртуальных частных сетей

Существует несколько классификаций виртуальных частных сетей, основанных на особенностях технологий их реализации. Признаками классификаций являются вид и собственник каналов, используемых для построения сети, а также вид применяемых средств шифрования.

По первому признаку виртуальные частные сети подразделяют следующим образом:

- истинные частные сети;
- сети на арендованных каналах;
- сети на каналах открытого доступа.

По второму признаку можно выделить виртуальные частные сети, функционирование которых основано на следующих элементах:

- сетевые операционные системы со встроенными функциями организации виртуальной частной сети;
- маршрутизаторы или коммутаторы;
- межсетевые экраны;
- средства криптографической защиты информации, предназначенные только для организации виртуальной частной сети.

Рассмотрим предложенную классификацию подробнее. Истинные частные сети организуются только в тех случаях, когда все каналы передачи защищаемой информации принадлежат корпоративной сети. Итак, истинная частная сеть - это такая сеть, в которой все оборудование (включая территориальные кабельные системы, коммутирующие устройства, средства управления и т.п) являются собственностью организации [8].

Такая виртуальная частная сеть гарантирует, что риск доступа к информации извне, то есть без привлечения сотрудников организации, практически сведен к нулю. Кроме того, в пределах истинной частной сети можно варьировать в широких пределах качество обслуживания и методы шифрования (кроме случаев, когда такая сеть используется для передачи информации, составляющей государственную тайну).

Виртуальные частные сети первого типа необходимо строить, соблюдая некоторые ограничения. Во-первых, сеть не должна располагаться на территории, принадлежащей другой организации; во-вторых, весь обслуживающий персонал, в том числе и службы ремонта, технической поддержки и информационной безопасности также не должны быть внешними.

Для истинных виртуальных частных сетей используют два типа телекоммуникаций:

- технологические линии связи – инфраструктура, предназначенная для передачи служебной информации или для обслуживания производственных процессов;
- специальные линии связи - инфраструктура, предназначенная непосредственно для виртуальных частных сетей.

Дополнительно следует отметить, что истинные частные сети являются наиболее защищенными, обладают наименьшим уровнем информационных рисков при передаче информации.

Сети на арендованных каналах построены по принципу защищенных виртуальных тоннелей в частично защищенной общей сети. В этом случае техническое и программное обеспечение передачи информации между локальными подразделениями организации берет на себя доверенный провайдер транспортных услуг.

В источнике [8] приведены несколько особенностей таких виртуальных частных сетей:

- арендуемые территориальные каналы прокладываются провайдером транспортных территориальных услуг в его первичной сети или сети с интегральными услугами ISDN;
- каналы, связывающие центральную сеть предприятия с сетями филиалов, проходят через мультиплексор, объединяющий каналы всех абонентов в магистральный канал;
- коммутация каналов в первичных сетях выполняется только оператором сети;
- пропускная способность выделенного каждому конкретному арендатору канала постоянна и заранее оговорена.

Кроме того, особенностью сети на арендованных каналах является существенное затруднение пассивного анализа трафика. Это происходит из-за того, что провайдер в данном случае не задействован в поддержке криптографических преобразований, и, следовательно, используемое сквозное или канальное шифрование обеспечивает достаточную конфиденциальность.

Последним вариантом является использование в качестве канала передачи глобальных сетей пакетной коммутации (Интернет). Такая сеть, называемая сетью на каналах открытого доступа, имеет ряд особенностей:

- использование неспециализированного оборудования (то есть отсутствие криптомаршрутизаторов);

- привлечение к процессу передачи информации большого числа организаций, функционально, территориально и структурно не ограниченных;
- использование пакетной коммутации.

В такой виртуальной частной сети требуется максимально обеспечить автономность средств криптографического преобразования и управления системой в целом, а также использовать ряд стандартизированных решений, в частности, инфраструктуру открытых ключей.

В том же источнике [8] производитель отмечает несколько преимуществ такого решения:

- простота и доступность;
- доступ к общим базам данных;
- территориальная независимость;
- доступ к корпоративной электронной почте;
- передача больших объемов данных по FTP протоколу;
- экономичность;
- гибкость;
- устойчивость.

Несмотря на перечисленные достоинства, необходимо отметить, что защита информации в такой сети может быть недостаточной. В первую очередь это касается аспекта доступности, который может быть утерян в результате отказов средств коммутации глобальной сети, а также подверженности подобного решения разного рода сетевым атакам.

3.4.2. Технология построения виртуальной частной сети

Технология построения виртуальной частной сети – это методы обеспечения конфиденциальности и целостности данных, передаваемых между пользователями, а также контроль эффективности этих методов [9].

Виртуальные частные сети строятся в первую очередь на технологии туннелирования, которая обеспечивает создание условно постоянного соединения между абонентами. Кроме того, необходимо использовать специальные устройства или программное обеспечение, реализующие криптографическое преобразование. Такими устройствами могут быть любые из перечисленных в классификации выше видов устройств шифрования.

Итак, для построения виртуальной частной сети необходимо использовать следующие функции оборудования:

- туннелирование;
- управление доступом;
- аутентификация;
- шифрование.

Управление доступом, аутентификация и шифрование - важнейшие элементы защищенного соединения. При реализации этих функций обеспечивается достижение главных целей построения виртуальной частной

сети – целостности и конфиденциальности передаваемой информации. Средством передачи при этом становится криптографический протокол. Рассмотрим основные протоколы по данным источника [9]:

- протокол PPP (Point-to-Point Protocol) используется в качестве универсального канального уровня;
- протокол PPTP (Point-to-Point Tunneling Protocol) реализует технологии туннелирования на канальном уровне;
- протокол L2TP (Layer 2 Tunneling Protocol) объединяет протоколы PPTP и L2F (Layer-2 Forwarding), управляя коммутацией каналов на туннельном уровне;
- протоколы IPSec и SKIP используется для организации туннеля на сетевом уровне;
- протоколы SSL, TLS, SOCKS обеспечивают существование туннеля на уровне представления.

Необходимо отметить, что в настоящее время для решения задач обеспечения информационной безопасности сведений, составляющих государственную тайну, могут быть использованы перечисленные выше протоколы, использующие в качестве основы криптопреобразования не стандарт DES, а отечественный стандарт ГОСТ 28147-89.

Далее рассмотрим пример организации виртуальной частной сети на базе сертифицированного средства – программно-аппаратного СКЗИ «Континент-К».

Пример: СКЗИ «Континент-К»

Данный аппаратно-программный комплекс (АПК) предназначен для построения виртуальных частных сетей (VPN) на основе глобальных сетей общего пользования, использующих протоколы семейства TCP/IP. Его основные функции:

- шифрование данных по ГОСТ 28147-89 в режиме гаммирования с обратной связью;
- контроль целостности по ГОСТ 28147-89 в режиме имитовставки;
- маршрутизация сетевого трафика;
- фильтрация сетевого трафика;
- возможность интеграции с системами обнаружения атак;
- организация защищенного обмена электронной почтой.

Общая структурная схема построения виртуальной частной сети на базе СКЗИ «Континент-К» представлена на рисунке 45:

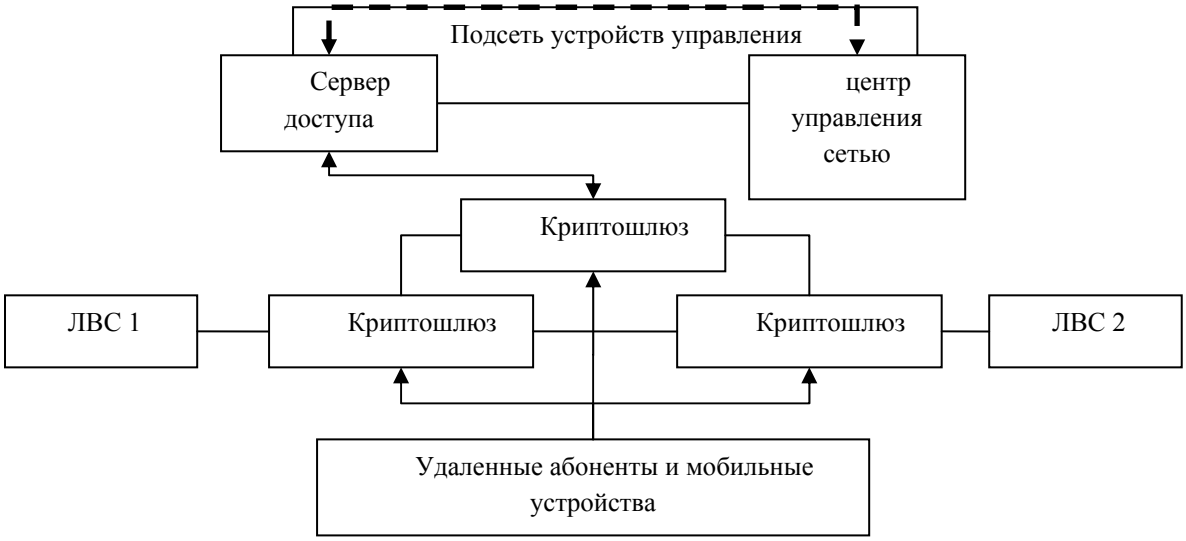


Рисунок 45. Схема виртуальной частной сети

Применение аппаратно-программного комплекса «Континент-К» возможно в сетях передачи данных при условии передачи информации, не составляющей государственную тайну. Достоинства описанного решения:

- обеспечение высокой пропускной способности (до 80 Мбит/с и выше);
- малая избыточность трафика: за счет реализации собственного протокола защищенной передачи данных сокращена до величины 26-36 байт на пакет (в зависимости от выбранного режима сжатия), что составляет существенно меньшую величину, чем при использовании IPSec (реализация ГОСТ 28147-89) - 54 байта на пакет или SKIP (реализация ГОСТ 28147-89) - 112 Байт на пакет;
- низкая стоимость эксплуатации корпоративной сети за счет использования сетей общего пользования вместо собственных или арендуемых линий связи;
- возможность сжатия передаваемой информации. Сжатие содержимого IP - пакета производится по алгоритму deflate (RFC 1951);
- скрытие внутренней структуры защищаемой сети;
- регистрация до 500 пользователей на каждом сервере доступа;
- возможность объединения в сеть до 5000 VPN устройств;
- создание информационных подсистем с разделением доступа на физическом уровне. В АПК обеспечивается возможность подключения 1 внешнего и до 5-7 внутренних интерфейсов на каждом криптошлюзе.

К основным особенностям реализации виртуальной частной сети таким способом можно отнести также поддержку возможности удаленного управления коммутационным оборудованием по защищенным каналам. Реализовано в системе «горячее» резервирование с помощью резервного аппаратного криптошлюза с аналогичными основному сетевыми адресами и создание резервной базы данных настроек и политик безопасности.

Необходимо отметить, что под «криптошлюзом» в данном случае понимается специальное устройство, совмещающее в себе шифратор

трафика, статический маршрутизатор и межсетевой экран. КШ функционирует под управлением защищенной версии ОС FreeBSD и программного обеспечения разработки НИП "ИНФОРМЗАЩИТА". Криптошлюз обеспечивает преобразование проходящего трафика в соответствии с ГОСТ 28147-89 (с длиной ключа шифрования 256 бит). Для защиты от несанкционированного доступа к системным данным криптошлюза используется электронный замок "Соболь".

3.5. НОВЫЕ НАПРАВЛЕНИЯ В КРИПТОГРАФИИ

В настоящее время получают распространение новые направления в криптографии, в частности, мультибазисная криптография и квантовая криптография.

Мультибазисная криптография основана на оригинальной алгоритмической идее (одновременно шифруется несколько сообщений).

Квантовая криптография базируется на использовании физических свойств элементарных частиц, пригодных для формиролвания информативного сигнала.

3.5.1. МУЛЬТИБАЗИСНАЯ КРИПТОГРАФИЯ

При рассмотрении данного направления обратимся к статье [11]. Перебор ключей при известном криптоалгоритме давно применяется при попытках получения доступа к важной информации. Такая атака на криптоалгоритм имеет смысл, когда нет других более быстрых (или менее затратных) методов получения интересующей информации. Любой подбор ключа подразумевает истинность одного важного условия – существует только один правильный вариант расшифрованного шифртекста. Если же это не так, и, вдобавок, неизвестно точное количество правильных вариантов, которое можно получить из шифртекста, то подбор ключа сводится к полному перебору всех допустимых комбинаций. В противном случае атакующий не может быть уверен, что расшифровал этот шифртекст целиком.

При подборе ключа крайне важно автоматизировать процесс определения валидности расшифровываемых данных, т.е. нужно уметь отличать смысловые данные, получаемые при расшифровке верным ключом от бессмысленных шумоподобных данных, получаемых при расшифровке с ошибочным ключом. Важность этого легко понять, если принять во внимание то, что ручной анализ расшифрованных данных будет производиться с частотой около одного раза в секунду, а мощность ключевого множества даже устаревшего DES очень велика.

Для получения неоднозначности при расшифровке, лицо, шифрующее информацию, готовит несколько сообщений, каждое из которых может быть опознано как смысловые данные, причем одно из этих сообщений несет действительную смысловую нагрузку, а остальные играют роль отвлекающего фактора. Задача криптоалгоритма состоит в том, чтобы получить такой шифртекст, из которого можно было бы получить все эти подготовленные сообщения. Получение каждого сообщения из шифртекста (расшифровка) должна осуществляться после предъявления соответствующего ключа. Эти ключи должны быть известны шифрующему алгоритму еще на этапе шифрования.

Назовем “точным шифрованием” строго детерминированный процесс, который при одинаковых исходных данных всегда будет давать одинаковый результат. Соответственно “размытым шифрованием” назовем такое обратимое преобразование данных, которое при каждом последующем использовании дает новый результат. Все известные автору криптоалгоритмы (кроме разновидностей ESS) осуществляют точное шифрование. Создать алгоритм точного шифрования, который позволял бы получить неоднозначность при расшифровке, автору не удалось. Поэтому им были предприняты попытки создания алгоритмов размытого шифрования, которые, в конце концов, увенчались успехом.

Криптоалгоритм, который позволяет получить неоднозначность при переборе ключей, был назван автором “мультибазисным”, поскольку ключи, на которых одновременно шифруются несколько сообщений, представлялись как базисы в n-мерном пространстве, где n – число шифруемых сообщений.

Мультибазисный криптоалгоритм содержит две сильно отличающихся друг от друга части – шифратор и дешифратор. Дешифратор – это довольно простой алгоритм, который позволяет получить расшифрованный текст, соответствующий введенному ключу. Шифратор – это сложный алгоритм поиска такого шифртекста, который позволял бы получить каждое из зашифровываемых сообщений. Во время работы шифратор рассчитывает очередной бит шифртекста как гипотезу.

Определение принимаемой гипотезы на том или ином шаге может происходить по одному из четырех сценариев:

1. Гипотеза может принимать значение как 1, так и 0.
2. Гипотеза может принимать значение 1, но не может принимать 0.
3. Гипотеза может принимать значение 0, но не может принимать 1.
4. Гипотеза не может принимать значение ни 1, ни 0.

Последний 4-й случай соответствует тому, что шифрование приходится приостановить, чтобы попробовать скорректировать те гипотезы в прошлом, которые допускают это (первый случай). Но иногда оказывается, что правкой гипотез, принятых в прошлом, невозможно исправить ситуацию – шифрование вновь и вновь останавливается. Такую ситуацию назовем тупиковой.

Попадание в тупиковую ситуацию обычно бывает спровоцировано схожими ключами, которые были использованы для одновременной зашифровки различных сообщений. Автор не нашел иных приемлемых способов определить, являются ли ключи достаточно различными, кроме как попытаться зашифровать на них сообщения. Перспективной, но не опробованной идеей является расчет корреляции пар ключей.

В общем, шифратор представляет собой реализацию вероятностной модели поведения по последовательному подбору бит шифртекста с коррекцией допущенных ошибок путем отката в прошлое. Детальное описание работы шифратора занимает слишком большой объем и поэтому здесь не рассмотрено.

Практическая сторона рассматриваемых подходов в криптографии довольно неоднозначна. Теоретики криптографии не учитывали, что количество одновременно зашифрованных осмысленных текстов может быть больше одного. Со стороны атакующего это означает то, что нельзя быть уверенным в том, что, подобрав один ключ, он не пропустит при этом другой. При этом возрастает количество ключей, которые требуется перебрать. Расшифровав все сообщения нужно будет затратить определенные силы на выявления истинного сообщения, что в некоторых случаях может оказаться невозможным. Со стороны обороняющегося это означает, что помимо смысловой информации нужно шифровать еще и дополнительную информацию, подобранную таким образом, чтобы проверка ее на корректность занимала у противника максимум ресурсов. Эта задача в общем нетривиальна и затраты на ее реализацию довольно высоки – необходимо заниматься сочинением набора подложных сообщений, которые были бы похожи на оригинальное сообщение. Видимо, помимо мультибазисных и совершенных шифров, не существует иных методов для получения неоднозначно расшифровываемого шифртекста.

3.5.2. КВАНТОВОЕ РАСПРЕДЕЛЕНИЕ КЛЮЧЕЙ

Итак, «квантовая криптография» или, если быть точнее, квантовое распределение ключей основано на нескольких технологических новшествах. Обратимся к статье [12].

Квантовые компьютеры. В 1994 году Питер Шор создал алгоритм факторизации целого числа, позволяющий найти решение приблизительно за $O((\log n)^2(\log\log n)(\log\log\log n))$ шагов, т.е. за полиномиальное время.

Такое решение возможно только с использованием квантового компьютера. Квантовый компьютер — это гипотетическое вычислительное устройство, существенно использующее при работе квантовомеханические эффекты, такие как квантовая суперпозиция и квантовый параллелизм.

Идея квантовых вычислений, впервые высказанная Ю. И. Маниным и Р. Фейнманом состоит в том, что квантовая система из L двухуровневых квантовых элементов (кубитов) имеет 2^L линейно независимых состояний, а значит, вследствие принципа квантовой суперпозиции, 2^L-мерное гильбертово пространство состояний. Операция в квантовых вычислениях соответствует повороту в этом пространстве. Таким образом, квантовое вычислительное устройство размером L кубит может выполнять параллельно 2^L операций.

Упрощенно говоря, квантовый компьютер – это цифровое устройство, в основе которого лежат аналоговые вычисления. Основу этих вычислений составляют операции над кубитами – квантовыми эквивалентами традиционных битов. Дело в том, что в квантовом компьютеры кубиты находятся в определенном состоянии, называемом связанным, когда состояние каждого кубита влияет на состояние других. В таблице 26

приведены различие физические реализации кубитов.

Таблица 26. Различные реализации кубитов

Физическая основа	Название	Носитель информации	«0»	«1»
Одиночный фотон	Поляризационное кодирование	Поляризация света	Горизонтальная	Вертикальная
	Количество фотонов	Количество фотонов	Вакуум	Одиночный фотон
	Временное кодирование	Время прибытие	Опережение	Запаздывание
Электроны	Спин электрона	Спин	Вверх	Вниз
	Количество электронов	Заряд	Нет электрона	Один электрон
Квантовая точка	Позиция электрона	Заряд	Электрон в левой точке	Электрон в правой точке
Когерентная основа света	Сжатый свет	Квадратура	Амплитудно-сжатое состояние	Фазово-сжатое состояние

В настоящее время предложено несколько различных путей реализации квантовых компьютеров. Это:

1. квантовые вычисления методом импульсного ядерного магнитного резонанса в молекулярных жидкостях;
2. с использованием в качестве элементной базы квантовых компьютеров ионов в ловушках в вакууме, спины одиночных электронов в квантовых точках в двумерном газе в полупроводниковых гетероструктурах, атомы в резонаторах электромагнитного поля;
3. на состояниях сверхпроводников, разделенных переходами Джозефсона и различающихся числом зарядов.

Существуют практические реализации квантового алгоритма Шора. Созданный квантовый компьютер основан на явлении ядерно-магнитного резонанса и состоял из семи кубитов, чего хватило для разложения числа 15 на простые множители 3 и 5.

По словам Артура Экерта, основателя Центра квантовых вычислений в Кембриджском университете, полноценных квантовых компьютеров следует ожидать уже через 10 лет.

Квантовое распределение ключей. Напомним, что единственной системой, для которой доказано, что она обеспечивает совершенную секретность, является система Вернама, т.е. метод одноразовых блокнотов.

Квантовая физика известна, как наука не только очень интересная, но и как крайне противоинтуитивная и местами даже причудливая. Известно, что квантовая физика накладывает некоторые ограничения, говорит о вещах, которые сделать невозможно. Например:

1. Невозможно измерить систему, не внеся в нее изменения;
2. Невозможно одновременно определить координаты и момент частицы со сколь угодно высокой точностью;
3. Невозможно измерить поляризацию фотона одновременно в горизонтально-вертикальном и диагональном базисах;
4. Невозможно скопировать неизвестное квантовое состояние.

Но эти ограничения оказываются фундаментом для создания систем квантового распределения ключей, т.е. для создания систем Вернама.

Главным отличием классической информатики от квантовой является возможность копировать информацию. В квантовой информатике, вследствие того что она базируется на квантовой физике, копирование неизвестного состояния – невозможно. Копирование информации в данном случае эквивалентно измерению состояния системы, а это в свою очередь изменяет состояние системы, т.е. «разрушает» исходное состояние.

Вуттерс и Цурек в 1982 году доказали теорему о невозможности копирования неизвестного состояния.

Пусть ψ – исходное состояние кубита, $|b\rangle$ - подготовленный кубит, «чистый лист», на который будет происходить копирование, $|0\rangle$ - исходное состояние копировальной машины. Идеальная машина должна произвести $\psi\otimes|b\rangle\otimes|0\rangle\rightarrow\psi\otimes\psi\otimes|f_\psi\rangle$, где $|f_\psi\rangle$ - конечное состояние копировальной машины, которое, вероятно, будет зависеть от ψ. В обозначениях протокола BB84 это можно записать $|\uparrow,b,0\rangle\rightarrow|\uparrow,\uparrow,f_\uparrow\rangle$ и $|\downarrow,b,0\rangle\rightarrow|\downarrow,\downarrow,f_\downarrow\rangle$. Здесь $\downarrow,\uparrow,\leftarrow,\rightarrow$ - это квантовый “1” и “0”. $\downarrow$ и $\leftarrow$ соответствуют “0” соответственно в горизонтально-вертикальном и диагональном базисах, а $\uparrow$ и $\rightarrow$ - “1” в тех же базисах.

Но тогда $|\rightarrow,b,0\rangle=(1/2)^{-1/2}(|\uparrow\rangle+|\downarrow\rangle)\otimes|b,0\rangle\rightarrow(1/2)^{-1/2}(|\uparrow,\uparrow,f_\uparrow\rangle+|\downarrow,\downarrow,f_\downarrow\rangle)$.

Мы видим, что конечное состояние отличается от идеальной копии $|\rightarrow,\rightarrow,f_\rightarrow\rangle$, при любых $|f_\psi\rangle$. Следовательно, невозможно создать идеальную квантовую копию, т.к. не может существовать идеальная квантовая копировальная машина.

Хотя квантовая криптография и является одним из самых разработанных в прикладном плане направлений квантовой физике, все еще остается множество нерешенных задач. Такими задачами можно назвать – усиление секретности, увеличение достоверности, коррекция ошибок, создание повторителей, усилителей, передача кубитов на большие расстояния.

К одним из самых перспективных направлений можно отнести исследования квантовой телепортации, создание воздушных квантовых каналов, изучение использования спутников, как источников связанных кубитов.

Контрольные вопросы и задания

1. Поясните принципы встраивания средств криптографической защиты информации в информационно-телекоммуникационную систему.
2. Назовите преимущества и недостатки СКЗИ программной, аппаратной и программно-аппаратной реализации.
3. Каковы алгоритмы шифрования, которые могут быть использованы в виртуальных частных сетях на территории РФ.
4. Укажите преимущества и недостатки централизованной и децентрализованной схемы электронной цифровой подписи.
5. Перечислите основные элементы инфраструктуры открытых ключей.
6. Поясните и сравните принципы использования криптографических ключей в СКЗИ программной, аппаратной и программно-аппаратной реализации.
7. В чем заключается преимущество внедрения многоключевой криптографии? Квантовой криптографии? Каковы недостатки таких систем?
8. Что общего между обычной и цифровой подписью? Чем они различаются?
9. Почему в криптографических системах, основанных на открытых ключах, нельзя использовать одинаковые ключи для шифрования и для цифровой подписи?
10. В системе аутентификации, основанной на схеме RSA, пользователь А выбрал открытый ключ e=7 и N=77. Если он получил от В число 23, то что А должен ответить, чтобы идентифицировать себя?
11. В схеме подписи, основанной на RSA, пользователи А и В имеют открытые ключи e_A=3, N_A=15; e_B=7, N_B=77 соответственно. Пользователь А хочет послать сообщение M=4 как подпись к некоторому тексту. Какое целое число он посылает?
12. Пусть $\overline{F_q}$ – алгебраическое замыкание F_q. Для поля К, $F_q \subset K \subset \overline{F_q}$, обозначим Е(К) – множество К-рациональных точек. Если m – количество точек Е(F_q), то должны выполняться следующие условия:

$$\begin{cases} q+1-2\cdot\sqrt{q} \in m \le q+1+2\sqrt{q}, \\ P \in E(F_q) \Rightarrow [m]P = 0. \end{cases} \qquad (*)$$

Пусть Р – точка эллиптической кривой Е/F_q и порядок этой точки больше $4\sqrt{q}$. Сколько различных целых чисел m удовлетворяют условиям (*)?

13. Количество значений блочно-итерационной хэш-функции h не больше количества образов шаговой функции хэширования ǽ. Доказать, что имеется $\sum_{i=0}^{N}(-1)^i \cdot \left(\frac{N}{i}\right) \cdot (N-i)^R$

различных сюръективных отображений $A^m \times A^n \to A^n$, где $N = |A|^n$, $R = |A|^{n+m}$.

14. Опишите различия между MD4 и MD5. Насколько защищенность MD5 выше по сравнению с MD4 и почему?
15. Почему нельзя в качестве хэш-функций использовать линейные отображения?
16. Можно ли использовать в качестве бесключевой хэш-функции ключевую хэш-функцию с фиксированным ключом?

ЗАКЛЮЧЕНИЕ

Криптографические методы и средства защиты информации выделены в группу дисциплин, которая объединяет как теоретические основы, так и практические реализации криптографического обеспечения информационной безопасности. В качестве полного руководства по описанным здесь методам и средствам защиты информации, а также как справочный материал пособие по этой дисциплине использоваться без дополнительных, указанных в списке литературы и рабочей программе дисциплины источников, не может. Целью издания является создание системного представления о предмете изучения.

Именно поэтому изучение основных принципов криптографической защиты информации, методов и средств, а также мер, реализующих упомянутые принципы, в группе дисциплин «Криптографические методы и средства защиты информации» является частью общей программы подготовки специалистов по специальностям 090105 «Комплексное обеспечение информационной безопасности автоматизированных систем», 090106 «Информационная безопасность телекоммуникационных систем».

В пособии раскрыты основные понятия, концепция и принципы организации криптографических средств обеспечения информационной безопасности, порядок решения различных задач обеспечения информационной безопасности в предметной области.

В процессе работы над пособием были по возможности учтены последние изменения в законодательстве, но, принимая во внимание его постоянное совершенствование, планируется издание соответствующих дополнений.

Необходимо отметить, что пособие ориентировано в первую очередь на специфику комплексного обеспечения информационной безопасности автоматизированных систем.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

Алферов, А. П. Основы криптографии : учеб. пособие / А. П. Алферов, А. Ю. Зубов, А. С. Кузьмин, А. В. Черемушкин. – М : Гелиос АРВ, 2001. – 480 с., ил.

Андреев, Н. Н. Основоположник отечественной засекреченной телефонной связи / Н. Н. Андреев, А. П. Петерсон, К. В. Прянишников, А. В. Старовойтов // Радиотехника, 1998. – № 8. – С. 8–12.

Анин, Б. Ю. Защита компьютерной информации. / Б. Ю. Анин. – СПб. : БХВ-Петербург, 2000. – 384 с.

Бабаш, А. В. Криптография-М / А. В. Бабаш, Г. П. Шанкин. – СОЛОН-Р, 2002. – 512 с.

Введение в криптографию / под общ. ред. В. В. Ященко. – 3-е изд., доп. – М. : Изд-во МЦНМО : ЧеРо, 2000. – 288 с.

Винокуров, А. Страничка классических блочных шрифтов [Электронный ресурс] / Андрей Винокуров. – Электрон. дан. – Режим доступа: http://www.enlight.ru/crypto/articles/ib/ib03.htm. – Загл. с экрана.

Домарев, В. В. Защита информации и безопасность компьютерных систем / В. В. Домарев. – Киев : Диасофт, 1999. – 480 с.

Жданов, О. Н. Алгоритм RSA : метод. указания к выполнению лаб. работ / О. Н. Жданов, И. А. Лубкин ; Сиб. гос. аэрокосмич. ун-т. – Красноярск, 2007. – 32 с. + 1 электрон. опт. диск (CD-ROM).

Золотарев, В. В. Программно-аппаратные средства обеспечения информационной безопасности / В. В. Золотарев ; Сиб. гос. аэрокосмич. ун-т. – Красноярск, 2007. – 112 с.

Зубов, А. Ю. Совершенные шифры / А. Ю. Зубов. – М. : Гелиос АРВ, 2003. – 117 с.

Коломиец, И. В. Проблема квантового распределения ключей / И. В. Коломиец // Актуал. проблемы безопасности информ. технологий : сб. науч. тр. / под общ. ред. О. Н. Жданова, В. В. Золотарева ; Сиб. гос. аэрокосмич. ун-т. – Красноярск, 2007. – с. 45–50.

Краковский, П. С. Введение неоднозначности в процесс дешифрования и мультибазисная криптография / П. С. Краковский // Актуал. проблемы безопасности информ. технологий : сб. науч. тр. / под общ. ред. О. Н. Жданова, В. В. Золотарева ; Сиб. гос. аэрокосмич. ун-т. – Красноярск, 2007. – с. 50–52.

Кукарцев, А. М. Операционный анализ криптоалгоритмов / А. М. Кукарцев, С. А. Старовойтов, В. С. Шестаков // Актуал. проблемы безопасности информ. технологий : сб. науч. тр. / под

общ. ред. О. Н. Жданова, В. В. Золотарева ; Сиб. гос. аэрокосмич. ун-т. – Красноярск, 2007. – с. 56–62.

Лукашин, И. В. Криптография? Железно! / И. В. Лукашин // Мир ПК – 2003. – № 3. – С. 100–109.

Межутков, А. А. Практическое руководство по методам и средствам криптографической защиты информации [Электронный ресурс] / А. А. Межутков. – Электрон. дан. (1 CD). – СПб. : Digital security, 2003.

Организация частных сетей и виртуальных частных сетей [Электронный ресурс]. – Электрон. дан. – Режим доступа : http://networkaccess.ru/articles/iptel/common/. – Загл. с экрана.

Панасенко, С. П. Принципы использования ключей шифрования / С. П. Панасенко // BYTE/Россия. – 2003. – № 9. – С. 65–68.

Панасенко, С. П. Аппаратные шифраторы / С. П. Панасенко, В. В. Ракитин // Мир ПК. – 2002. – № 8. – С. 77–83.

Прасолов, В. В. Эллиптические функции и алгебраические уравнения / В. В. Прасолов, Ю. П. Соловьев. – М. : Факториал, 1997. – 288 с.

Соколов, А. В. Защита от компьютерного терроризма / А. В. Соколов, О. М. Степанюк. – СПб. : БХВ-Петербург : Арлит, 2002. – 496 с.

Столлингс, В. Криптография и защита сетей: принципы и практика / В. Столлингс ; пер. с англ. – 2-е изд. – М. : Вильямс, 2001. – 672 с.

Титце, У. Полупроводниковая схемотехника : справ. руководство / У. Титце, К. Шенк ; пер. с нем. – М. : Мир, 1982. – 512 с., ил.

Уязвимость и информационная безопасность телекоммуникационных технологий : учеб. пособие / А. А. Новиков, Г. Н. Устинов ; под ред. Г. Н. Устинова. – М. : Радио и связь, 2003. – 296 с.

Шеннон, К. Теория связи в секретных системах / К. Шеннон // Работы по теории информации и кибернетике. – М. : Изд-во иностран. лит., 1963. – С. 333–402.

Щепинов, А. С. Заметки о совершенных шифрах [Электронный ресурс] / А. С. Щепинов. – Электрон. дан. – Режим доступа: http://www.cryptography/msg.html-mid=1169602.htm. – Загл. с экрана.

Шнайер, Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си / Брюс Шнайер. – М. : Триумф, 2003. – 816 с., ил.

Diffie, D. New directions in cryptography / D. Diffie, M. Hellman // IEEE Transactions on information theory. – 1976. – v. IT-22. – N 6. – P. 644–654.

Koblitz, N. A course in number theory and cryptography / N. Koblitz. – N.Y. : Springer-Verlag, 1987. – 208 p.

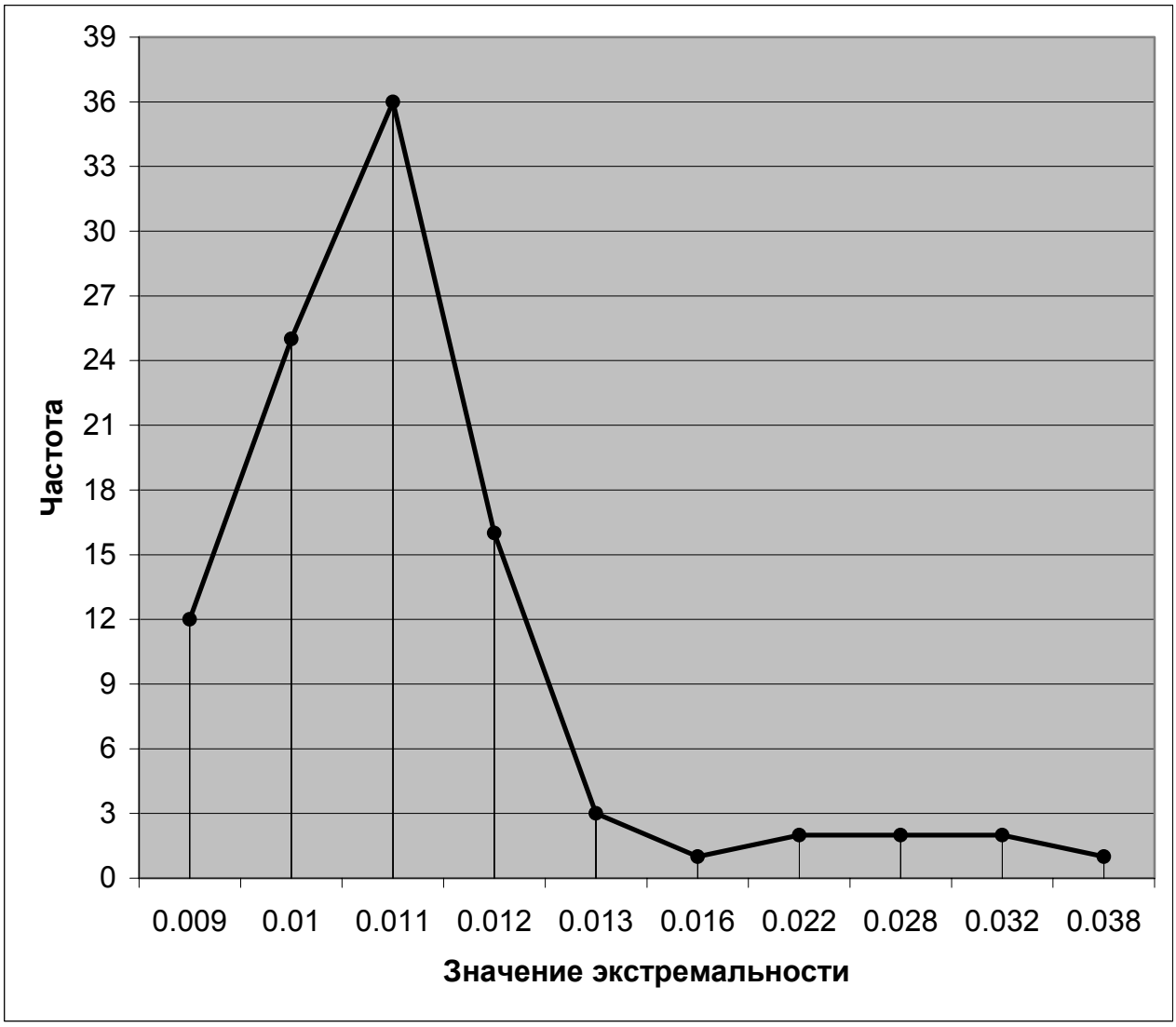
Odlyzko, A. Discrete logarithms and smooth polynomials / A. Odlyzko // Contemp. math. – 1994. – v. 168. – P. 269–278.

Rivest, R. A method for obtaining digital signatures and public keycryptosystems / R. Rivest, A. Shamir, L. Adleman // Communications of the ACM. – 1978 (febr.). – P. 120–126.

Schoof, R. Elliptic curves over finite fields and the computation of square roots modp / R. Schoof // Math. comp. – 1985. – v. 44. – P. 483–494.

ПРИЛОЖЕНИЕ

Результаты исследований алгоритма DES на лавинный эффект



Полигон частот выборки для первой итерации

Выборка для первой итерации характеризуется следующими числовыми характеристиками:

Выборочное среднее $\bar{\varepsilon}_в = 0.0116$;

Выборочная дисперсия $D_в = 0.000000010$;

Выборочное средне квадратичное отклонение выборки $\sigma_в = 0.00010$;

Исправленная выборочная дисперсия $S^2 = 0.000000010$;

Исправленное выборочное средне квадратичное отклонение выборки $S = 0.0001$;

Размах вариации $R = 0.029$;

Мода вариационного ряда $M_o^* = 0.011$;

Медиана вариационного ряда $M_e^* = 0.011$.

Приведенные числовые характеристики выборки позволяют оценить ее как хорошую.

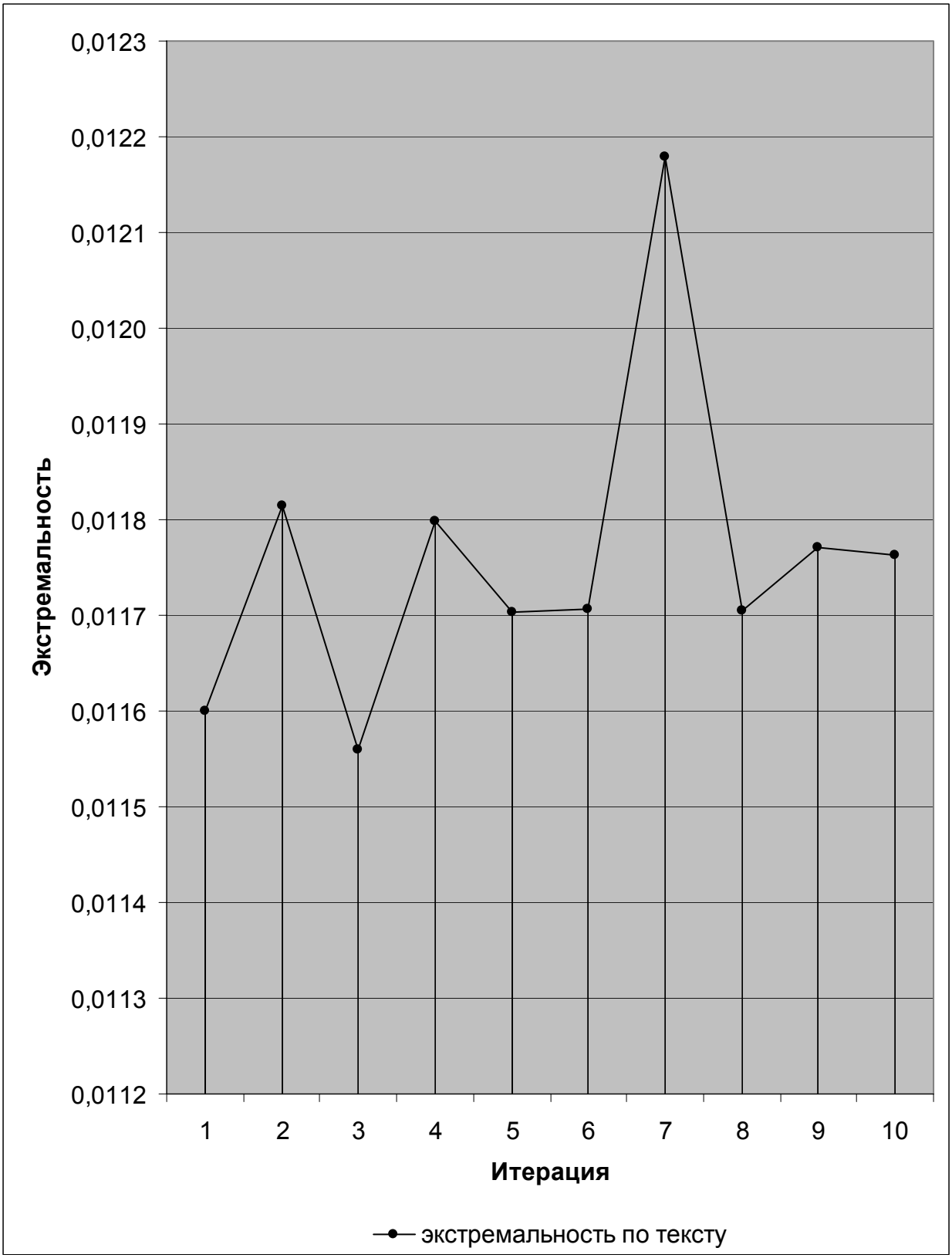


График экстремальности по тексту алгоритма DES

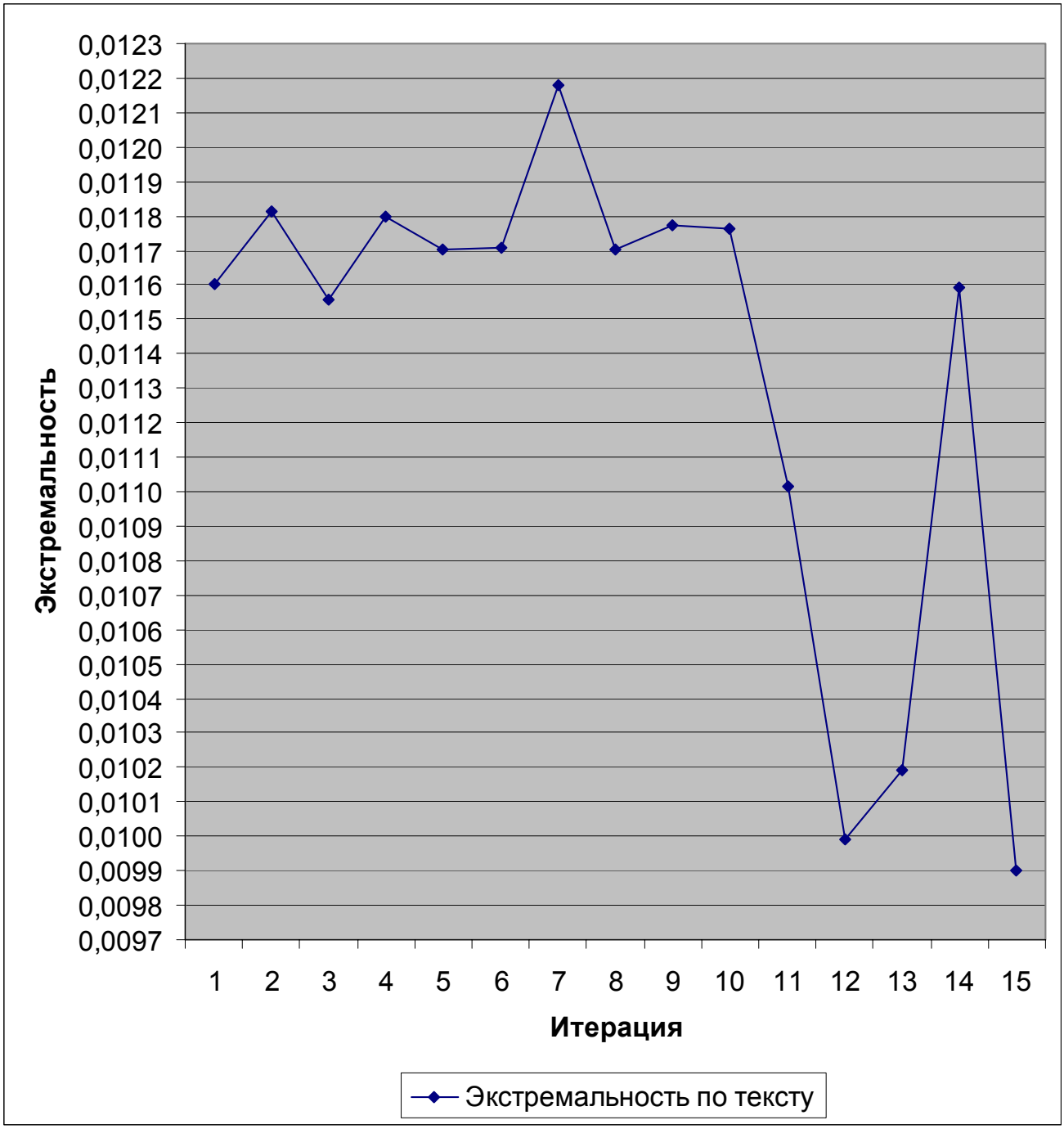


График экстремальности по тексту 15 итераций алгоритма DES

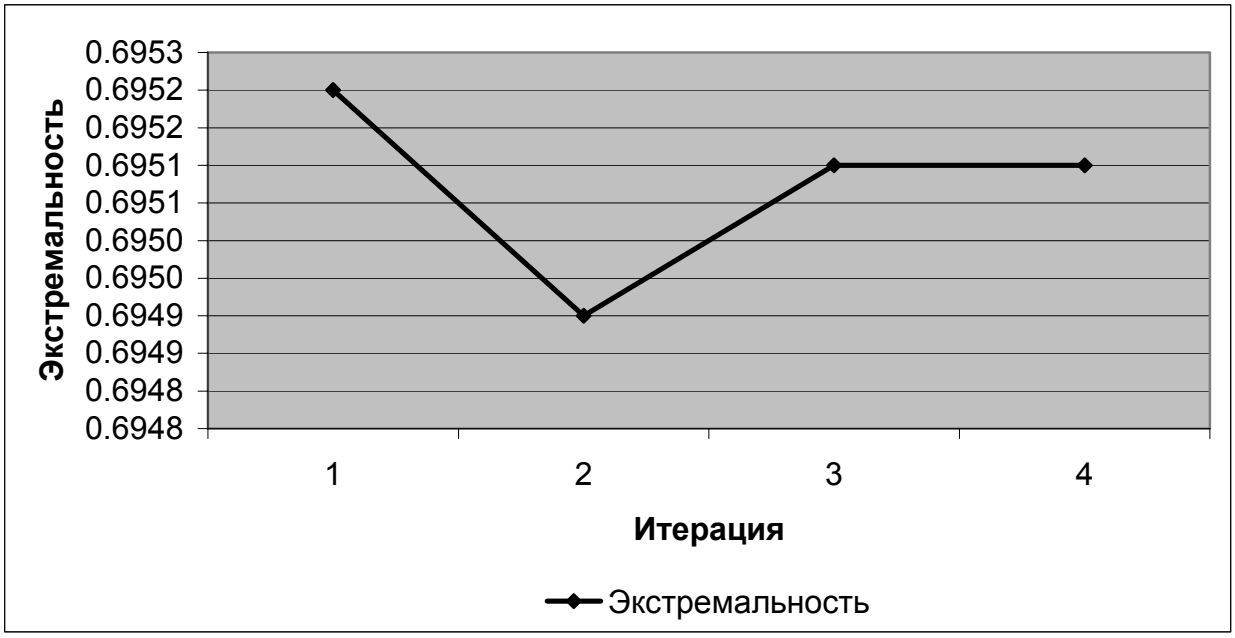


График экстремальности по тексту 4 итераций алгоритма ECC