Министерство образования и науки
Российской Федерации

НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»

М.А. ИВАНОВ, И.В. ЧУГУНКОВ

КРИПТОГРАФИЧЕСКИЕ МЕТОДЫ ЗАЩИТЫ ИНФОРМАЦИИ В КОМПЬЮТЕРНЫХ СИСТЕМАХ И СЕТЯХ

Под редакцией М.А. Иванова

Рекомендовано УМО «Ядерные физика и технологии»
в качестве учебного пособия
для студентов высших учебных заведений

Москва 2012

УДК 681.3
ББК 32.81
И 20

Иванов М.А., Чугунков И.В. **Криптографические методы защиты информации в компьютерных системах и сетях:** Учебное пособие / Под ред. М.А. Иванова. М.: НИЯУ МИФИ, 2012. – 400 с.: ил.

Излагаются вопросы применения криптографических методов для обеспечения безопасности информации. Доказывается, что только на основе их использования удается успешно решать задачи защищенного взаимодействия удаленных абонентов, в том числе обеспечения секретности информации, аутентичности субъектов и объектов информационного взаимодействия.

Предназначено для студентов вузов и аспирантов, обучающихся по компьютерным специальностям, а также специальностям, связанным с обеспечением безопасности компьютерных технологий.

Рекомендуется использовать при изучении дисциплин «Методы и средства защиты компьютерной информации», «Безопасность информационных систем» для студентов, обучающихся по специальности «Вычислительные машины, комплексы, системы и сети». Может быть полезно разработчикам и пользователям компьютерных систем.

Подготовлено в рамках Программы создания и развития НИЯУ МИФИ.

Рецензент д-р техн. наук, проф. С.В. Дворянкин

ISBN 978-5-7262-1676-8

Оглавление

Список используемых сокращений …… 9
Список используемых терминов …… 10
Принятые обозначения …… 17
Введение …… 20

ЧАСТЬ 1. ВВЕДЕНИЕ В КРИПТОЛОГИЮ …….. 43

ГЛАВА 1. ОСНОВЫ КРИПТОЛОГИИ …….. 43

1.1. Основные термины и определения ……. 43
1.2. Оценка надежности криптоалгоритмов ……. 45
1.3. История криптологии ……... 48
1.4. Классификация методов шифрования информации 49
1.5. Шифры замены ……. 53
1.6. Шифры перестановки ……... 54
1.7. Шифр Ф. Бэкона …… 56
1.8. Блочные составные шифры …….. 58
1.9. Абсолютно стойкий шифр. Гаммирование ……. 64
1.10. Поточные шифры ……. 70

ГЛАВА 2. КРИПТОСИСТЕМЫ С СЕКРЕТНЫМ КЛЮЧОМ …… 74

2.1. Модель симметричной криптосистемы …… 74
2.2. Классификация угроз противника. Основные свойства криптосистемы ……. 76
2.3. Классификация атак на криптосистему с секретным ключом ……... 76
2.4. Режимы использования блочных шифров …… 78
2.5. Российский стандарт криптографической защиты ….. 87
2.6. Американский стандарт криптографической защиты …. 93
2.7. Вероятностное блочное шифрование …… 103
2.8. Синхронные поточные шифры …….. 105
2.9. Самосинхронизирующиеся поточные шифры ……. 114

ГЛАВА 3. ХЕШ-ФУНКЦИИ ……. 118

3.1. Требование к качественной хеш-функции …….. 118

3.2. Итеративная хеш-функция ……. 122
3.3. Secure Hash Algorithm (SHA) ……. 122
3.4. Хеш-функции на основе симметричных блочных криптоалгоритмов ……... 128

ГЛАВА 4. МЕТОДЫ АУТЕНТИФИКАЦИИ ИНФОРМАЦИИ …….. 130

4.1. Аутентичность. Задача аутентификации информации ……... 130
4.2. Имитозащита информации. Контроль целостности потока сообщений ……... 131
4.3. Криптографические методы контроля целостности … 135
4.4. Код аутентификации сообщений …….. 135
4.5. Код обнаружения манипуляций с данными ……. 136
4.6. Код HMAC …….. 139
4.7. Код CMAC …….. 141
4.8. Идентификация, аутентификация и авторизация …… 142
4.9. Аутентификация субъекта ……. 145
4.10. Аутентификация объекта ……. 149

ГЛАВА 5. КРИПТОСИСТЕМЫ С ОТКРЫТЫМ КЛЮЧОМ …… 153

5.1. Односторонние функции …… 154
5.2. Модель криптосистемы с открытым ключом …….. 156
5.3. Открытое распределение ключей …… 158
5.4. Электронная цифровая подпись …… 160
5.5. Криптосистема, основанная на задаче об укладке рюкзака ……. 162
5.6. Криптосистема RSA …… 167
5.7. Криптосистема Эль-Гамаля ……... 171
5.8. Гибридные криптосистемы …… 172
5.9. Генераторы ПСЧ на основе односторонних функций с секретом ……. 175

ГЛАВА 6. КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ ……. 180

6.1. Основные понятия ……... 180
6.2. Доказательства с нулевым разглашением …… 181

6.2.1. Пещера нулевого знания …… 183
6.2.2. Протокол Фиата–Шамира …….. 183
6.2.3. Протокол Фейга–Фиата–Шамира ……. 187
6.3. Протоколы подбрасывания монеты …….. 188
6.4. Протоколы битовых обязательств …… 192
6.5. Протоколы разделения секрета …….. 193
6.6. Протоколы электронной подписи ……. 195
6.6.1. Протокол электронной подписи RSA ……. 198
6.6.2. Протокол электронной подписи Шнорра ……... 199
6.6.3. Классификация атак на схемы электронной подписи …….. 201
6.6.4. Процедура разрешения споров …….. 202
6.6.5. Особые схемы электронной подписи …… 204
6.7. Протоколы аутентификации удаленных абонентов … 204
6.7.1. Симметричные методы аутентификации субъекта ……... 204
6.7.2. Схема Kerberos …… 207
6.7.3. Несимметричные методы аутентификации субъекта ……... 211

ГЛАВА 7. УПРАВЛЕНИЕ КЛЮЧАМИ ……... 217

7.1. Разрядность ключа …….. 217
7.2. Генерация ключей ……... 220
7.3. Неоднородное ключевое пространство ……. 221
7.4. Хранение ключей …… 222
7.5. Распределение ключей ……... 226
7.6. Время жизни ключей …….. 228

ЧАСТЬ 2. ОСНОВЫ ТЕОРИИ, ПРИМЕНЕНИЯ И ОЦЕНКИ КАЧЕСТВА ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ ……..... 230

ГЛАВА 8. ПРИНЦИПЫ ПОСТРОЕНИЯ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ …….. 230

8.1. Стохастические методы защиты информации ……. 231
8.2. Требования к генераторам ПСЧ …… 233
8.3. Классификация генераторов ПСЧ ……. 235

8.4. Генераторы ПСЧ на регистрах сдвига
с линейными обратными связями ……. 244
8.4.1. Последовательные РСЛОС …… 244
8.4.2. Параллельные РСЛОС …… 247
8.4.3. Примеры двоичных РСЛОС …….. 250
8.4.4. Основы теории конечных полей …… 253
8.5. Аддитивные генераторы ПСЧ ……... 258
8.6. Стохастическое преобразование информации.
R-блоки ……. 261
8.6.1. Модификация существующих алгоритмов
поточного шифрования …… 265
8.6.2. Нелинейные М-последовательности на основе
R-блоков 266

ГЛАВА 9. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ
СТОХАСТИЧЕСКИХ МЕТОДОВ
В ЗАДАЧАХ ЗАЩИТЫ ИНФОРМАЦИИ
В ЭЛЕКТРОННЫХ ПЛАТЕЖНЫХ СИСТЕМАХ …….. 269

9.1. Цифровые деньги …… 270
9.2. Защита информации в ЭПС на основе
цифровых денег …….. 274
9.3. Слепая электронная подпись ……. 275
9.4. Структура централизованной платежной системы
на основе цифровой наличности.
Анализ жизненного цикла цифровой купюры ……. 277
9.5. Механизмы обеспечения безопасности информации
в ЭПС на основе протокола SET ……... 281

ГЛАВА 10. ИССЛЕДОВАНИЕ СТАТИСТИЧЕСКОЙ
БЕЗОПАСНОСТИ ГЕНЕРАТОРОВ ПСЧ ……. 287

10.1. Тесты, применяемые для оценки качества
генераторов ПСЧ ……... 287
10.2. Руководство по статистическому тестированию
НИСТ …….. 288
10.3. Повышение эффективности оценочных тестов ……. 295
10.4. Комплекс программных средств оценки качества
генераторов ПСЧ ……... 301

ЧАСТЬ 3. ЭЛЛИПТИЧЕСКАЯ КРИПТОГРАФИЯ ……. 305

ГЛАВА 11. ЭЛЛИПТИЧЕСКИЕ КРИВЫЕ
И КРИПТОГАФИЯ ……. 305

11.1. Основные понятия и определения …….. 305
11.2. Группа точек эллиптической кривой ……. 306
11.3. Сложение точек эллиптической кривой ……. 309
11.4. Скалярное умножение точек эллиптической кривой ……. 310
11.5. Задача дискретного логарифмирования на эллиптической кривой ……. 311
11.6. Эллиптические кривые над конечными полями $GF(p)$ ……. 313
11.7. Эллиптические кривые над конечными полями $GF(2^m)$ …… 314
11.8. Параметры криптографической схемы …….. 317
11.9. Схема гибридного шифрования ECIES …….. 318
11.10. Эллиптические алгоритмы генерации ПСЧ ……. 320
11.10.1. Эллиптические алгоритмы формирования ПСЧ на основе линейных конгруэнтных генераторов …… 321
11.10.2. Эллиптические алгоритмы формирования ПСЧ на основе регистров сдвига с линейными обратными связями ……. 323
11.11. Сравнение систем на основе эллиптических кривых с другими асимметричными криптосистемами …… 325

ГЛАВА 12. КРИПТОСИСТЕМЫ, ОСНОВАННЫЕ
НА СВОЙСТВАХ ЭЛЛИПТИЧЕСКИХ КРИВЫХ ……. 328

12.1. Схема симметричного шифрования на эллиптических кривых ……. 328
12.2. Схема асимметричного шифрования на эллиптических кривых ……. 330
12.3. Схема электронной цифровой подписи на эллиптических кривых …… 331
12.4. Схема ЭЦП на эллиптических кривых (ECDSA) ….. 333

12.5. Протокол выработки общего секретного ключа (ECKEP) ……. 334
12.6. Схема гибридного шифрования (ECKAP) …….. 336
12.7. Российский стандарт на ЭЦП ГОСТ Р 34.10-2001…. 338

ГЛАВА 13. АТАКИ НА ECDLP ОБЩЕГО ВИДА …….. 345

13.1. Полный перебор (Exhaustive Search) …….. 345
13.2. Атака Полига–Хеллмана (Pohlig–Hellman Attack) …. 345
13.3. Алгоритм маленьких и больших шагов Шэнкса (Baby-step Giant-step) …… 348
13.4. ρ-метод Полларда ……..... 349
13.5. λ-метод Полларда …….. 352
13.6. Дискретное логарифмирование методом индексного исчисления …… 352

Заключение …… 355

Список использованных источников информации ……... 362

Контрольно-измерительные материалы по курсу «Методы и средства защиты компьютерной информации» .. 366

Приложение 1. Неприводимые многочлены над $GF(p)$, p – простое ……... 387
Приложение 2. Примитивные многочлены над $GF(2)$ …… 392
Приложение 3. Криптоалгоритмы с архитектурой «Куб» ….. 395

Список используемых сокращений

АНБ – Агентство национальной безопасности США
ИС – информационная система
КС – компьютерная система
НИСТ – Национальный институт стандартов и технологий США
НСД – несанкционированный доступ (к информации)
ОБИ – обеспечение безопасности информации
ОС – операционная система
ПО – программное обеспечение
ПС – программная система
ПСП – псевдослучайная последовательность
ПСЧ – псевдослучайные числа
ССЗ – самое слабое звено
ЭПС – электронная платежная система
ЭЦП – электронная цифровая подпись
AES – Advanced Encryption Standard
CRC – cyclic redundancy code
DES – Data Encryption Standard
ECCS – Elliptic Curves Cryptosystem
ECDLP – Elliptic Curves Digital Logarithm Problem
MAC – Message Authentication Code
MDC – Manipulation Detection Code
OAEP – Optimal Asymmetric Encryption Padding

Список используемых терминов

Авторизация – 1) (в ЭПС) процесс проверки достаточности средств на карточном счету для проведения операции по карте; в процессе авторизации данные о карте и о запрашиваемой сумме передаются в банк, выпустивший карту, где проверяется состояние счета клиента; 2) процесс проверки прав, предоставляемых конкретному пользователю (какую информацию он может использовать и каким образом (читать, записывать, модифицировать или удалять), какие программы может выполнять, какие ресурсы ему доступны и др.)

Альтернативная статистическая гипотеза – гипотеза о том, что проверяемая последовательность является неслучайной

Анализ рисков – процесс определения рисков безопасности, их величины и значимости

Атака – действие или последовательность действий, использующие уязвимости системы для нарушения ее безопасность системы

Атака перехвата и повтора – попытка вторжения в систему или подделки финансовых средств оплаты с использованием ранее использованных реквизитов доступа или других ранее использованных данных

Аутентификация объекта (информационного взаимодействия) – процедура установления целостности массива данных или полученного сообщения; а также в большинстве случаев установление их принадлежности конкретному автору

Аутентификация субъекта (информационного взаимодействия) – процедура установления подлинности пользователя (абонента сети), программы или устройства

Аутентичность информации (аутентичность объекта информационного взаимодействия) – свойство информации быть подлинной, достоверной; означающее, что информация была создана законными участниками информационного процесса и не подвергалась случайным или преднамеренным искажениям

Генератор ПСЧ – устройство или программа, формирующие последовательность ПСЧ; именно качеством используемых генераторов ПСЧ определяется надежность стохастических методов защиты

Графический статистический тест – тест, при проведении которого статистические свойства последовательностей отображаются в виде графических зависимостей, по которым делают выводы о свойствах исследуемой последовательности

Дешифрование – процесс преобразования закрытых (зашифрованных) данных в открытые при неизвестном ключе (по сути, взлом криптоалгоритма)

Доступность (информации) – гарантия того, что злоумышленник не сможет помешать работе авторизованных пользователей, иначе говоря, обеспечение своевременного доступа пользователей к необходимой им информации или компонентам системы

Задача дискретного логарифмирования – задача, на сложности решения которой основана стойкость криптосистем Эль-Гамаля и ECCS

Задача факторизации целых чисел – задача, на сложности решения которой основана стойкость криптосистемы RSA

Зашифрование – процесс преобразования открытых данных в закрытые (зашифрованные)

Защита информации – комплекс мероприятий, направленных на обеспечение информационной безопасности

Информационная безопасность – защищенность информации и поддерживающей инфраструктуры от случайных и умышленных деструктивных воздействий

Конфиденциальность – необходимость предотвращения утечки (разглашения) какой либо информации

Криптоалгоритм – алгоритм криптографического преобразования

Криптоанализ – научная дисциплина, оценивающая надежность криптосистем, разрабатывающая способы анализа стойкости шифров и способы их вскрытия

Криптографический примитив – «строительный блок» (операция или процедура), используемый для решения какой-либо частной задачи криптографической защиты информации (например, генератор ПСЧ, хеш-генератор и пр.)

Криптографический протокол – протокол, использующий криптографическое преобразование

Криптографическое преобразование – специальное преобразование (шифрование, хеширование) сообщений или хранимых данных для решения задач обеспечения секретности или аутентичности (подлинности) информации

Криптография – научная дисциплина, разрабатывающая способы преобразования информации с целью ее защиты от противника, обеспечения ее секретности, аутентичности (целостности, подлинности) и неотслеживаемости

Криптология – научная дисциплина, включающая в себя два основных раздела, цели которых прямо противоположны: криптографию и криптоанализ

Криптостойкость (стойкость к криптоанализу) – способность криптосистемы, противостоять различным атакам на нее

Метод грубой силы (brute force attack) – метод преодоления криптографической защиты перебором большого числа вариантов (например, перебор всех возможных ключей для расшифрования сообщения или всех возможных паролей для неавторизованного входа в систему)

Нарушение целостности информации динамическое – нарушение атомарности транзакций, переупорядочение, дублирование или внесение дополнительных сообщений

Нарушение целостности информации статическое – несанкционированное изменение данных или ввод злоумышленником неверных (искаженных или фальсифицированных) данных

Недекларируемые возможности – функциональные возможности аппаратуры или ПО, не описанные или не соответствующие описанным в документации

Неотслеживаемость (untracebility) – свойство транзакции, когда покупатель не только анонимен, но и два платежа, совершенных одним и тем же лицом, не могут быть увязаны между собой ни при каких условиях

Несанкционированный доступ к информации – ознакомление, обработка, копирование, модификация или уничтожение информации в нарушение установленных правил разграничения доступа

Несимметричное шифрование (шифрование с открытым ключом) – криптоалгоритм, использующий для за- и расшифрования различные ключи, соответственно открытый и закрытый (секретный); при этом знание открытого ключа не дает возможности

вычислить закрытый; используется для реализации криптографических протоколов, в частности протокола электронной подписи

Нулевая статистическая гипотеза – гипотеза о том, что проверяемая последовательность является случайной

Обеспечение доступности – исключение возможности атак, вызывающих отказ в обслуживании (DoS-атак)

Обеспечение секретности – исключение возможности пассивных атак на передаваемые или хранимые данные

Оценочный статистический тест – тест, при проведении которого статистические свойства последовательностей определяются числовыми характеристиками; на основе оценочных критериев делаются заключения о степени близости свойств анализируемой и истинно случайной последовательностей

Ошибка 1-го рода (для статистического теста) – ситуация, когда тест указывает на неслучайность последовательности в то время, как она является случайной

Ошибка 2-го рода (для статистического теста) – ситуация, когда тест указывает на случайность последовательности в то время, как она является неслучайной

Политика информационной безопасности – совокупность мер противодействия угрозам ИБ, целью применения которых является уменьшение риска либо за счет уменьшения вероятности осуществления угрозы, либо за счет уменьшения последствий реализации угрозы

Поточный (потоковый) *шифр* – криптоалгоритм, обеспечивающий шифрование в реальном масштабе времени или близком к нему; обеспечивает преобразование каждого элемента входного потока данных на своем элементе потока ключей (гаммы); главное требование к поточному шифру – высокое быстродействие; именно поточные шифры используются при программной реализации симметричных криптоалгоритмов

Протокол – многораундовый обмен сообщениями между участниками, направленный на достижение конкретной цели (выработку общего секретного ключа, аутентификацию абонентов и пр.); протокол включает в себя: характер и последовательность действий каждого из участников, спецификацию форматов пере-

сылаемых сообщений, спецификацию синхронизации действий участников, описание действий при возникновении сбоев

Псевдослучайная последовательность (ПСП) – последовательность псевдослучайных чисел, которая по своим статистическим свойствам не отличается от истинно случайной последовательности, но в отличие от последней может быть повторена необходимое число раз

Разграничение доступа – совокупность реализуемых правил доступа к ресурсам системы

Разделение доступа (привилегий) – разрешение доступа только при одновременном предъявлении полномочий всех членов группы

Рассеивание – свойство криптографического преобразования, назначение которого – скрыть статистические особенности открытого текста за счет распространения влияния каждого отдельного элемента открытого текста на множество элементов зашифрованного текста

Расшифрование – процесс преобразования закрытых (зашифрованных) данных в открытые при известном ключе

Риск – вероятность того, что конкретная атака будет осуществлена с использованием конкретной уязвимости

Секретность информации – гарантия невозможности доступа к информации для неавторизованных пользователей

Сертификат – документ, удостоверяющий какой-либо факт, например подлинность открытого ключа

Симметричное шифрование (шифрование с секретным ключом) – криптоалгоритм, использующий для за- и расшифрования один и тот же секретный ключ

Статистические методы – методы крипто- и стегоанализа, основанные на проведении статистических тестов, обнаруживающих закономерности в проверяемой информационной последовательности

Стеганография – научная дисциплина, разрабатывающая методы скрытия факта передачи или хранения секретной информации

Стегоанализ – научная дисциплина, оценивающая надежность стегосистем, разрабатывающая способы анализа стойкости стеганографических методов защиты и способы их вскрытия

Стохастический метод защиты – 1) в широком смысле – метод, основанный на использовании генераторов псевдослучайных чисел или хеш-генераторов; 2) в узком смысле – метод защиты, основанный на использовании стохастических сумматоров (*R*-блоков), т. е. сумматоров с непредсказуемым результатом работы; для не знающих ключевой информации результат работы стохастического сумматора выглядит случайным; метод сочетает в себе эффективную реализацию (как программную, так и аппаратную) и высокую крипто- и имитостойкость

Угроза информационной безопасности – потенциально возможное событие, действие (воздействие), процесс или явление, которое посредством воздействия на информацию или другие компоненты ИС может прямо или косвенно привести к нанесению ущерба. Различают угрозы секретности (конфиденциальности) информации, угрозы аутентичности (целостности, подлинности) информации и угрозы доступности информации

Управление рисками ИБ – процедура постоянной (пере)оценки рисков и выбора эффективных и экономичных защитных механизмов для их нейтрализации

Уровень риска – функция вероятности реализации соответствующей угрозы ИБ и размера возможного ущерба

Хеш-генератор – устройство (или программа), реализующее операцию хеширования

Хеширование – необратимое сжатие данных с использованием хеш-функции

Хеш-образ – результат действия хеш-функции; иногда называемый дайджестом сообщения или массива данных

Хеш-функция – необратимая функция сжатия, принимающая на входе массив данных произвольной длины и дающая на выходе хеш-образ фиксированной длины; результат действия хеш-функции зависит от всех бит исходного массива данных и их взаимного расположения; используется для организации парольных систем, реализации протокола электронной подписи, формирования контрольного кода целостности (кода MDC)

Целостность информации – отсутствие случайных или умышленных (несанкционированно внесенных) искажений информации

Цель защиты информации – уменьшение размеров ущерба от различных нарушений ИБ до допустимых значений

Цифровая подпись – см. электронная подпись

Шифр – совокупность алгоритмов за- и расшифрования

Шифрование – процедура за- или расшифрования

Электронная подпись (цифровая подпись, электронная цифровая подпись) – результат шифрования хеш-образа подписываемого массива данных на секретном ключе подписывающего; электронная подпись в отличие от обычной допускает неограниченное копирование подписанной информации и может существовать отдельно от нее; процедура проверки электронной подписи для своего выполнения не требует никакой секретной информации

Электронная цифровая подпись – см. электронная подпись

MAC (Message Authentication Code) – код аутентификации сообщений; в отличие от кода MDC для своего вычисления требует знания секретного ключа

MDC (Manipulation Detection Code) – код обнаружения манипуляций с данными, в отличие от кода MAC для своего вычисления не требует знания секретного ключа

P-value – вероятность того, что совершенный генератор случайных чисел произвел бы последовательность менее случайную, чем исследуемая, для типа неслучайности, проверяемого статистическим оценочным тестом

R-блок (R – random) – стохастический сумматор, т. е. сумматор с непредсказуемым результатом работы (см. стохастические методы защиты)

S-блок (S – substitution) – блок замены (подстановки)

Принятые обозначения

$c = c_1 c_2 \ldots c_i \ldots c_t$, $i = \overline{1, t}$, – шифротекст, зашифрованное сообщение или массив данных

c_i – i-й блок шифротекста

c_0 – синхропосылка

D – функция расшифрования

D_A – функция расшифрования на секретном ключе $k_A^{(secret)}$ абонента A или секретная функция генерации подписи абонента A

D_{AB} – функция расшифрования на секретном ключе k_{AB}, разделяемом абонентами A и B

D_k – функция расшифрования на ключе k_d

E – функция зашифрования или эллиптическая кривая

E_A – функция расшифрования на открытом ключе $k_A^{(public)}$ абонента A или открытая функция проверки подписи абонента A

E_{AB} – функция зашифрования на секретном ключе k_{AB}, разделяемом абонентами A и B

$E(GF(p))$ – множество точек (x, y) эллиптической кривой E, удовлетворяющих условию $x, y \in GF(p)$, и бесконечно удаленная точка

E_k – функция зашифрования на ключе k_e

$E(Z_p)$ – множество точек (x, y) эллиптической кривой E, удовлетворяющих условию $x, y \in Z_p$, и бесконечно удаленная точка

$e = e_0 e_1 \ldots e_i \ldots e_{m-1}$, $i = \overline{0, (m-1)}$, $e_i \in \{0, 1\}$, – последовательность ошибок

$F_k(x)$ – односторонняя функция с секретом k

G – матрица, определяющая логику работы блока пространственного сжатия информации

g – генератор элементов конечного поля

$GF(p)$ – поле Галуа из p элементов, p – простое

$GF\left(p^n\right)$ – поле Галуа из p^n элементов, p – простое, n – натуральное

H – таблица стохастического преобразования, таблица замен ГОСТ 28147-89

$h(x)$ – хеш-функция

iv – вектор инициализации

$K\{M\}$ – шифрование документа M с использованием ключа K

k – ключ шифра

k_d – ключ расшифрования

k_e – ключ зашифрования

$k_A^{(public)}$ – открытый ключ абонента A

$k_A^{(secret)}$ – закрытый ключ абонента A

nrb – неповторяющийся блок данных

$m = m_1 m_2 ... m_i ... m_t,\ i = \overline{1, t}$, или $M = M_1 M_2 ... M_i ... M_t,\ i = \overline{1, t}$, – открытый текст, открытое сообщение или массив данных

m_i или M_i – i-й блок открытого текста

m_0 или M_0 – синхропосылка

$\{M\}K$ – формирование ЭЦП на документе M с использованием ключа K

O – бесконечно удаленная точка эллиптической кривой

$O(n)$ – асимптотическая оценка сложности алгоритма (асимптотическую скорость возрастания количества операций при увеличении n), где n – параметр сложности исходных данных

$Q_i(t)$ – состояние i-го регистра генератора псевдослучайных последовательностей в момент времени t

$q_i(t)$ – состояние i-го триггера генератора псевдослучайных последовательностей в момент времени t

s – контрольный код или электронная подпись

$sign_A(M)$ – электронная подпись абонента A на документе M

T – сопровождающая матрица линейной последовательностной машины

$u \leftarrow a$ – присвоить переменной u значение a

$|x|$ – разрядность числа x

$]x[$ – наименьшее целое $n \geq x$

$[x]$ – наибольшее целое $n < x$

x_A – случайный запрос абонента A

y_A – результат преобразования запроса x_A

Z_p – множество целых чисел от 0 до $p - 1$

Z_p^* – множество целых чисел от 1 до $p - 1$

$\Phi(x) = \alpha_N x^N + ... + \alpha_i x^i + ... + \alpha_1 x + \alpha_0$, $i = \overline{0, N}$, $\alpha_i \in GF\left(p^n\right)$, – многочлен (полином) с коэффициентами из поля $GF\left(p^n\right)$

$\gamma = \gamma_1 \gamma_2 ... \gamma_i ... \gamma_m$, $i = \overline{1, m}$, – гаммирующая последовательность (гамма шифра)

$\varphi(n)$ – функция Эйлера, n – натуральное

$\varphi(x)$ – характеристический многочлен генератора псевдослучайных последовательностей

ω – примитивный элемент поля Галуа

$\oplus$ – поразрядное сложение по модулю 2

$\|$ – конкатенация

Введение

Проблема информационной безопасности. Под информационной безопасностью понимается защищенность информации и поддерживающей инфраструктуры (в том числе компьютерных систем (КС), систем электро- и теплоснабжения, средств коммуникаций и обслуживающего персонала; в последнем случае речь идет, естественно о компетенции) от случайных и умышленных деструктивных воздействий, которые могут нанести неприемлемый ущерб субъектам информационного взаимодействия. Очевидно, что застраховаться от всех видов ущерба невозможно. Тем более невозможно сделать это экономически целесообразным образом, когда стоимость защитных механизмов не превышает размер возможного ущерба. Защищаться следует только от неприемлемого ущерба, которым может являться, например, нанесение вреда здоровью людей, хотя чаще порог неприемлемости имеет материальное выражение. Защита информации – это комплекс мероприятий, направленных на обеспечение информационной безопасности. Целью защиты является уменьшение размеров ущерба до допустимых значений [9].

Трактовка проблем, связанных с информационной безопасностью, для разных категорий субъектов может существенно различаться. Например, для режимных государственных организаций в первую очередь необходимо обеспечить секретность информации, для банковских систем – доступность и аутентичность информации. Для медицинских систем актуальны все три задачи – обеспечение конфиденциальности (врачебная тайна), обеспечение целостности и доступности (предписанные медицинские процедуры, это та информация, нарушение целостности которой может оказаться в буквальном смысле смертельным, как и невозможность оперативно получить своевременную медицинскую помощь или консультацию опытного специалиста).

Компьютерные системы как объекты защиты информации. Жизнь современного общества немыслима без повсеместного использования КС, связанных с вводом, хранением, обработкой и выводом информации. Всеобщая компьютеризация, помимо очевидных выгод, несет с собой и многочисленные проблемы, наиболее сложной из которых является проблема инфор-

мационной безопасности. Высочайшая степень автоматизации, к которой стремится человечество, широкое внедрение дешевых компьютерных систем массового применения и спроса делают их чрезвычайно уязвимыми по отношению к деструктивным воздействиям, ставят современное общество в зависимость от степени безопасности используемых информационных технологий.

Появление персональных компьютеров расширило возможности не только пользователей, но и нарушителей. *Важнейшей характеристикой любой компьютерной системы, независимо от ее сложности и назначения, становится безопасность циркулирующей в ней информации*.

Информационная безопасность давно стала самостоятельным направлением исследований и разработок. Однако, несмотря на это, проблем не становится меньше. Это объясняется появлением всё новых компьютерных технологий, которые не только создают новые проблемы информационной безопасности, но и представляют, казалось бы, уже решенные вопросы совершенно в новом ракурсе. Кроме того, появление новых компьютерных технологий, новых математических методов дают в руки нарушителей и создателей разрушающих программных воздействий (РПВ) все новые и новые возможности [29].

Главная причина трудоемкости решения задачи обеспечения безопасности информации (ОБИ) в современных условиях – всё большее отстранение пользователей от процессов управления и обработки информации и передача его полномочий программному обеспечению (ПО), обладающему некоторой свободой в своих действиях и поэтому очень часто работающему вовсе не так, как предполагает пользователь.

Как отмечается в [11, 22, 25, 28, 30], трудоемкости решения задачи защиты информации также способствуют:

увеличение объемов информации, накапливаемой, хранимой и обрабатываемой с помощью компьютерной техники;

сосредоточение в единых базах данных информации различного назначения и принадлежности;

расширение круга пользователей, имеющих доступ к ресурсам компьютерной системы и находящимся в ней массивам данных;

усложнение режимов функционирования технических средств компьютерной системы;

увеличение количества технических средств и связей в КС;

повсеместное использование зарубежной элементной базы и ПО;

повсеместное распространение сетевых технологий, объединение локальных сетей в глобальные;

появление новых транспортных средств передачи данных;

появление общедоступной, не находящейся ни в чьем владении сети Интернет, практически во всех сегментах которой отсутствуют какие-либо контролирующие или защитные механизмы;

бурное развитие ПО, в большинстве своем не отвечающего минимальным требованиям безопасности;

наличие различных стилей программирования, появление новых технологий программирования, затрудняющих оценку качества используемых программных продуктов;

невозможность в современных условиях при осуществлении кредитно-финансовой деятельности (сферы, чрезвычайно привлекательной для злоумышленников) обойтись без активного взаимного информационного обмена среди субъектов этой деятельности.

Предмет защиты. Ценность информации является критерием при приятии любого решения о ее защите. Известно [4] следующее разделение информации по уровню важности:

жизненно важная незаменимая информация, наличие которой необходимо для функционирования компьютерной системы, организации и т.п.;

важная информация – информация, которая может быть заменена или восстановлена, но процесс восстановления очень труден или связан с большими затратами;

полезная информация – информация, которую трудно восстановить, однако компьютерная система или организация могут эффективно функционировать и без нее;

несущественная информация.

Ценность информации обычно изменяется со временем и зависит от степени отношения к ней различных групп лиц, участвующих в процессе обработки информации [4]:

источников или поставщиков информации;

держателей или обладателей, собственников информации;

владельцев систем передачи, сбора и обработки информации;

законных пользователей или потребителей информации;

нарушителей, стремящихся получить информацию, к которой в нормальных условиях не имеют доступа.

Отношение этих групп лиц к значимости одной и той же информации может быть различным: для одной важная, для другой – нет. Например:

важная оперативная информация, такая, например, как список заказов на данную неделю или график производства, может иметь высокую ценность для держателя, тогда как для источника (например, заказчика) или нарушителя низка;

персональная информация, например медицинская, имеет значительно бóльшую ценность для источника (лица, к которому относится информация), чем для держателя или нарушителя;

информация, используемая руководством для выработки решений, например о перспективах рынка, может быть значительно более ценной для нарушителя, чем для источника или ее держателя, который уже завершил анализ этих данных.

Существует деление информации по уровню *секретности, конфиденциальности*. Признаками секретной информации является наличие, во-первых, *законных пользователей*, которые имеют право владеть этой информацией, во-вторых, *незаконных пользователей* (нарушителей, противников), стремящихся овладеть этой информацией с тем, чтобы обратить ее себе во благо, а законным пользователям – во вред. Для наиболее типичных, часто встречающихся ситуаций такого типа введены даже специальные понятия: государственная тайна, военная тайна, коммерческая тайна, юридическая тайна, врачебная тайна и т.п.

Угрозы безопасности и методы защиты информации в КС. Цель создания систем безопасности – предупреждение или оперативное устранение последствий *умышленных* (преднамеренных) и *случайных деструктивных воздействий*, следствием которых могут быть *разрушение*, *модификация* или *утечка* инфор-

мации, а также *дезинформация*. В том случае, если объект *атаки* противника какой-то из компонентов системы (аппаратные средства или ПО) можно говорить о *прерывании*, когда компонент системы становится недоступен, теряет работоспособность или попросту утрачивается в результате обычной кражи; *перехвате* и (или) *модификации*, когда противник получает доступ к компоненту и (или) возможность манипулировать с ним; *подделке*, когда противнику удается добавить в систему некий компонент или процесс (разрушающее программное воздействие), файлы или записи в них [28].

Эффективная система ОБИ должна обеспечивать [22]:

секретность всей информации или наиболее важной ее части;

аутентичность субъектов и объектов информационного взаимодействия;

правильность функционирования компонентов системы в любой момент времени, в том числе отсутствие недекларируемых возможностей (НДВ);

своевременный доступ пользователей к необходимой им информации или компонентам системы;

защиту авторских прав, прав собственников информации, возможность разрешения конфликтов;

разграничение ответственности за нарушение правил информационных взаимоотношений;

оперативный контроль правильности реализации алгоритма управления, в том числе контроль хода выполнения программ;

непрерывный анализ защищенности процессов управления, обработки и передачи информации.

Как будет показано далее, при создании электронных платежных систем, помимо перечисленных функций, необходимо обеспечивать неотслеживаемость информации, например для обеспечения анонимности участников финансовых транзакций.

Следует отметить два важных факта. Во-первых, как уже отмечалось выше, в зависимости от объекта защиты возможна различная расстановка приоритетов среди перечисленных свойств системы безопасности. В случае защиты государственных секретов наивысший приоритет имеет секретность информации. При

защите же, например, банковской платежной системы наиболее важными свойствами следует признать обеспечение своевременного доступа пользователей к необходимым им ресурсам системы и достоверность информации. Во-вторых, имеет место трудноразрешимое противоречие между эффективным выполнением системой своих основных функций и степенью обеспечения в ней необходимого уровня безопасности. Чем более высокие требования предъявляются к безопасности системы, тем большее количество ее ресурсов оказывается задействованным для обеспечения этих требований, соответственно тем неудобнее пользователям, работающим в этой системе. И наоборот, чем больше ресурсов выделяется для эффективной работы пользователей, тем меньше их остается для решения задачи обеспечения безопасности.

Назначение средств обеспечения секретности и конфиденциальности – защитить информацию от пассивных атак. При этом иногда устанавливается несколько уровней защиты в зависимости от важности содержимого сообщений.

Конфиденциальную информацию можно разделить на предметную и служебную, к последней относятся, например, пароли пользователей. Раскрытие служебной информации особенно опасно, поскольку последствием этого будет являться получение НСД ко всей информации, включая предметную.

В случае единичного сообщения назначение средств обеспечения аутентичности – проверка того, что источник данного сообщения – именно тот субъект, за которого выдает себя отправитель. При интерактивном взаимодействии эти средства, во-первых, должны гарантировать, что оба участника информационного взаимодействия аутентичны (т.е. действительно те, за кого себя выдают), а во-вторых, не должны допускать несанкционированного влияния на информационный обмен какой-либо третьей стороны.

Различают статическую и динамическую целостность данных. В тех случаях, когда злоумышленник вводит неверные (искаженные или фальсифицированные) данные или изменяет данные, имеет место статическое нарушение целостности. Потенциально уязвимы с точки зрения нарушения целостности не только

данные, но и программы. Внедрение вредоносного ПО – пример подобного нарушения.

Угрозами динамической целостности являются нарушение атомарности транзакций, переупорядочение, дублирование или внесение дополнительных сообщений.

Назначение средств защиты целостности – гарантировать то, что принятые сообщения в точности соответствуют отправленным и не содержат изъятий, дополнений, повторов и изменений в порядке следования фрагментов. Может контролироваться целостность как частей сообщения, так и сообщения в целом, а также потока сообщений. Дополнительной функцией соответствующих средств (помимо основной – оперативного обнаружения нарушений целостности), может стать и восстановление искаженной информации [9].

Назначение средств управления доступом – исключить из процессов информационного взаимодействия незаконных участников (субъектов и объектов) и предотвратить выход законных участников за пределы своих полномочий. В процессе аутентификации субъекта выделяют три стадии: идентификация (проверка подлинности идентификаторов субъекта), собственно аутентификация и авторизация (проверка того, какие права предоставлены данному субъекту, какие ресурсы он может использовать, какие действия может совершать и пр.).

Аутентификация пользователей (субъектов) может быть основана на следующих принципах [25]:

предъявлении пользователем пароля;

предъявлении пользователем доказательств, что он обладает секретной ключевой информацией, в том числе возможно хранящейся на смарт-карте;

предъявлении пользователем некоторых признаков, неразрывно связанных с ним;

установлении подлинности пользователя некой третьей, доверенной стороной.

Если сообщение было отправлено не внушающим доверия отправителем, получатель должен иметь возможность доказать, что сообщение действительно отправлено. И наоборот, если сообщение отправлено не внушающему доверия получателю, от-

правитель должен иметь возможность доказать, что сообщение этим адресатом получено.

Назначение соответствующих средств обеспечения доступности – обеспечить своевременный доступ пользователей к необходимой им информации и ресурсам информационной системы. Задача обеспечения доступности – комплексная, для ее решения применяются как методы защиты от НСД, так и специализированные методы защиты от разрушающих программных воздействий (РПВ) [29].

Очень часто имеет место взаимодействие различных аспектов информационной безопасности. НСД к служебной информации, например, может являться предпосылкой для последующего нарушения целостности или доступности.

Конфиденциальность, целостность и доступность информации могут быть нарушены как в результате преднамеренных действий злоумышленника, так и в результате объективных воздействий со стороны среды.

Причинами случайных деструктивных воздействий, которым подвергается информация в процессе ввода, хранения, обработки, вывода и передачи, могут быть [22]:

отказы и сбои аппаратуры;

помехи на линиях связи от воздействий внешней среды;

ошибки человека как звена системы;

ошибки разработчиков аппаратного и (или) программного обеспечения;

аварийные ситуации.

К методам защиты от случайных воздействий можно отнести:

помехоустойчивое кодирование;

контролепригодное и отказоустойчивое проектирование;

процедуры контроля исправности, работоспособности и правильности функционирования аппаратуры;

самоконтроль или самотестирование;

контроль хода программ и микропрограмм.

Преднамеренные угрозы связаны с действиями нарушителя (злоумышленника), который при этом может воспользоваться как штатными (законными), так и другими каналами доступа к информации в КС (рис. В.1). При этом согласно [22] в результате действий нарушителя, которым может являться как незакон-

ный, так и законный пользователь, информация подвергается следующим угрозам:

кража носителей информации и оборудования;

несанкционированное копирование программ и данных;

уничтожение или несанкционированная модификация информации или ПО;

несанкционированный доступ (НСД) к секретной или конфиденциальной информации (хранимой или передаваемой по каналам связи);

представление себя в качестве другого лица с целью уклонения от ответственности, либо отказа от обязательств, либо с целью использования прав другого лица для:

- отправки фальсифицированной информации;
- искажения имеющейся информации;
- НСД с помощью фальсификации или кражи идентификационных данных или их носителей;
- фальсифицированной авторизации транзакций или их подтверждения;

приведение компонентов системы в неработоспособное состояние или состояние неправильного функционирования;

уклонение от ответственности за созданную информацию;

фальсификация источника информации или степени ответственности, например заявление о получении некоторой информации от другого абонента, хотя на самом деле информация была создана самим нарушителем;

фальсификация времени отправки (получения) или самого факта отправки (получения) информации;

отказ от факта получения или передачи сообщения, в частности отказ от факта получения или передачи сообщения в определенный момент времени;

расширение своих полномочий нарушителем, например на получение доступа, создание информации, ее распространение и т.п.;

несанкционированное создание учетных записей или изменение (ограничение или расширение) полномочий других пользователей;

использование скрытых каналов передачи данных, например сокрытие наличия некоторой тайной информации в другой информации (стеганографическое скрытие информации);
появление побочных каналов утечки секретной информации, например о промежуточных результатах работы криптоалгоритмов;
внедрение в линию связи между другими абонентами в качестве активного тайного ретранслятора;
дискредитация защищенного протокола информационного взаимодействия, например путем разглашения сведений, которые согласно этому протоколу должны храниться в секрете;
изменение функций ПО (обычно с помощью добавления скрытых функций);
провоцирование других участников информационного взаимодействия на нарушение защищенного протокола, например путем предоставления неправильной информации;
препятствование взаимодействию других абонентов, например путем скрытого вмешательства, вызывающего отказ в обслуживании или прекращение легального сеанса как якобы нелегального и пр.;
разрыв связи и дальнейшая работа с системой под видом законного пользователя;
приведение системы в состояние, требующее незапланированных затрат ресурсов (например, на обслуживание поступающих сообщений и запросов, ведение оперативного контроля, восстановление работоспособности, устранение попыток нарушения безопасности и т.п.);
фальсификация сообщений (например, выдача себя за другого пользователя; санкционирование ложных обменов информацией; навязывание ранее переданного сообщения; приписывание авторства сообщения, сформированного самим нарушителем, другому лицу и т.п.);
нарушение протокола обмена информацией с целью его дискредитации;
имитация работы системы, анализ поведения интересующего компонента или анализ трафика с целью получения ин-

формации об идентификаторе пользователя, поиска каналов утечки информации, каналов скрытого влияния на объект, правилах вступления в связь и т.п.

Рис. В.1. Каналы доступа к информации в КС

На рис. В.2 показаны виды атак на протоколы информационного обмена.

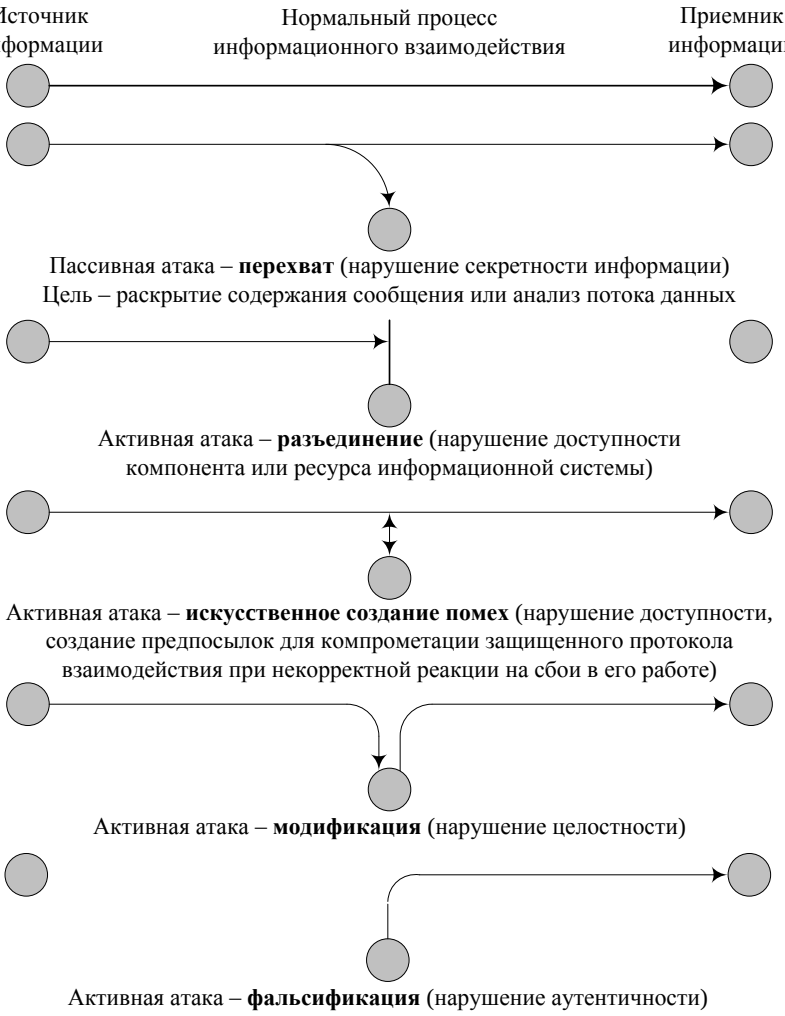


Рис. В.2. Виды атак на протоколы информационного взаимодействия

Для каждого типа угроз обычно можно предложить одну или несколько мер противодействия, целью применения которых является уменьшение риска либо за счет уменьшения вероятности осуществления угрозы, либо за счет уменьшения последствий реализации угрозы. В совокупности указанные меры образуют политику безопасности. Основные характеристики каждой меры

противодействия – эффективность и стоимость, именно они являются основой для проведения рациональной с экономической точки зрения политики безопасности. При оценке угроз со стороны противника следует учитывать также стоимость их реализации. «Нормальный» противник не будет расходовать на реализацию угрозы больше средств, чем он может получить от последствий ее исполнения. Поэтому одной из целей мер противодействия может стать увеличение цены нарушения безопасности системы до уровня, превышающего оценку достигаемого противником выигрыша. С учетом различных экономических и социальных санкций, которые ждут нарушителя в случае обнаружения его действий, эффективной мерой защиты может являться всего лишь оперативное обнаружение факта реализации угрозы.

Вторая группа методов защиты (рис. В.3) значительно обширнее и включает в себя:

методы защиты от несанкционированного доступа (НСД) к информации;

методы защиты от РПВ;

организационные методы защиты, направленные на защиту от НСД, защиту от РПВ, совершенствование защиты и восстановление информации.

Помимо рассмотренных на рис. В.3 можно выделить также следующие методы защиты информации [22]:

законодательные меры;

принципы и правила работы в системе, уменьшающие риск нарушения безопасности, например регулярное создание резервных копий системы или наиболее важных ее компонентов, установление определенной процедуры копирования;

формирование этических норм для пользователей, своего рода «кодекса поведения», согласно которому считаются неэтичными любые умышленные или неумышленные действия, которые могут нарушить нормальную работу системы.

На рис. В.4 и В.5 показаны соответственно модель системы защиты информационного взаимодействия по каналам связи и модель системы защиты от РПВ.

Наиболее распространенные угрозы ИБ. Самые частые и самые опасные с точки зрения размера ущерба – непреднамеренные ошибки программистов, законных пользователей, системных администраторов, администраторов безопасности, обслуживающего персонала. Во многих случаях такие ошибки и являют-

ся собственно угрозами. По некоторым данным, до 65 % потерь – следствие непреднамеренных ошибок, причины которых – безграмотность и небрежность в работе [9].

Рис. В.3. Классификация методов защиты КС от умышленных деструктивных воздействий

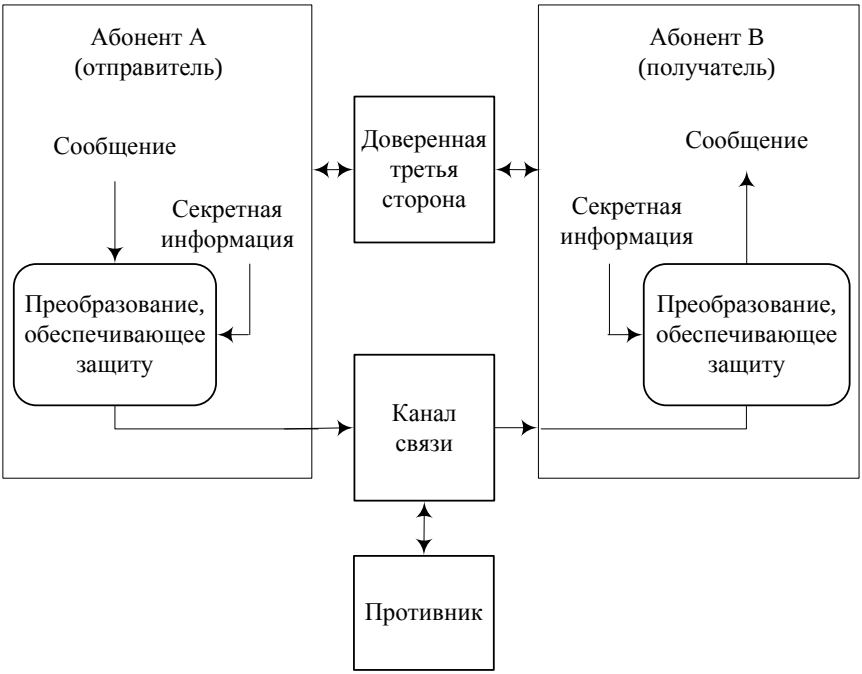


Рис. В.4. Модель системы защиты удаленного взаимодействия двух абонентов

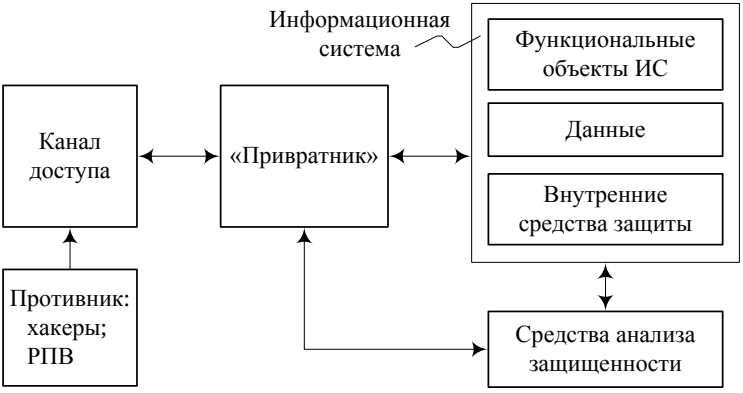


Рис. В.5. Модель системы защиты от РПВ

Самые радикальные способы борьбы с непреднамеренными ошибками – обучение пользователей основам информационной безопасности, максимальная автоматизация и строгий контроль.

Административный уровень информационной безопасности. К административному уровню ИБ относятся действия общего характера, осуществляемые руководством организации. Главная цель этих мер заключается в разработке программы работ в области ИБ и обеспечении ее выполнения путем выделения необходимых ресурсов и контроля за состоянием дел. Политика безопасности (иначе говоря, стратегия организации в области ИБ) строится на основе анализа рисков, которые признаются реальными для информационной системы организации.

Политику безопасности следует рассматривать на трех уровнях детализации. К верхнему уровню относят решения, затрагивающие организацию в целом. Цели на этом уровне формулируются в терминах «доступность», «аутентичность» и «конфиденциальность».

К среднему уровню относят такие вопросы, касающиеся ИБ, как отношение к передовым (но, скорее всего, недостаточно проверенным) технологиям, подключение к Интернету, применение нелицензионного ПО и т.п.

Политика безопасности нижнего уровня относится к конкретным информационным сервисам. Решаются вопросы, касающиеся того, кто имеет право доступа к объектам, поддерживаемым сервисом; при каких условиях можно читать и модифицировать данные; как организовать удаленный доступ к сервису [9].

Управление рисками. Использование информационных систем связано с некоторой совокупностью рисков. Когда возможный ущерб неприемлемо велик, необходимо принимать экономически оправданные меры защиты. Периодическая переоценка рисков необходима для контроля эффективности деятельности в области ИБ и учета изменений обстановки.

С количественной точки зрения уровень риска суть функция вероятности реализации соответствующей угрозы и размера возможного ущерба.

Управление рисками включает в себя (пере)оценку рисков и выбор эффективных и экономичных защитных механизмов для

нейтрализации рисков. По отношению к выявленным рискам возможны следующие действия:

ликвидация риска (например, за счет устранения причины);

уменьшение риска до допустимого уровня;

принятие риска и выработка плана действий при реализации соответствующей угрозы;

переадресация риска (например, путем заключения страхового соглашения).

Процесс управления рисками можно разделить на следующие этапы:

выбор анализируемых объектов и уровня детализации их рассмотрения;

анализ угроз и их последствий;

выявление уязвимых мест в защите;

оценка рисков;

выбор защитных мер;

оценка остаточного риска [9].

Процедурный уровень информационной безопасности. Рассмотрим меры безопасности, которые ориентированы на человека – самое слабое звено любой информационной системы.

На процедурном уровне можно выделить следующие меры:

обучение пользователей основам информационной безопасности;

управление персоналом путем реализации двух принципов: разделения обязанностей и минимизации привилегий;

физическая защита, основным принципом которой является непрерывность защиты во времени и пространстве;

поддержание работоспособности компонентов КС, в том числе поддержание ПО и документации в актуальном состоянии, резервное копирование программ и данных, контроль за проведением регламентных работ;

реагирование на нарушение политики информационной безопасности с целью локализации инцидента и уменьшения наносимого вреда, выявления нарушителя и предупреждения повторных нарушений;

планирование восстановительных работ с целью подготовки к различного рода внештатным ситуациям, уменьшения

ущерба в случае их возникновения и сохранение работоспособности КС хотя бы в минимальном объеме [9].

Криптографические методы ОБИ. Анализ угроз безопасности в КС, тенденции развития информационных технологий дают все основания сделать вывод о постоянном возрастании роли криптографических методов при решении задач аутентификации в распределенных системах, обеспечения секретности данных при их передаче по открытым каналам связи, юридической значимости результатов информационного обмена. В публикациях по теме информационной безопасности криптографической защите уделяется особое внимания, так как специалисты считают ее наиболее надежной, а в некоторых ситуациях единственно возможным механизмом защиты.

В стандартных криптографических протоколах широко используется модель угроз уязвимой среды Долева–Яо, согласно которой злоумышленник может [21, 30]:

перехватить любое сообщение, передаваемое по сети;

являться законным пользователем и вступать в контакт с любым другим пользователем;

получать сообщения от любого пользователя;

посылать сообщения любому пользователю, маскируясь под любого другого пользователя.

Криптографические методы защиты информации являются предметом исследования современной *криптологии*, включающую в себя два основных раздела, цели которых прямо противоположны:

1) *криптографию*, разрабатывающую способы преобразования (шифрования или хеширования) информации с целью ее защиты от злоумышленников, обеспечения ее *секретности*, *аутентичности* и *неотслеживаемости*;
2) *криптоанализ*, связанный с оценкой надежности криптосистем, анализом стойкости шифров, разработкой способов их вскрытия.

В настоящее время стандартный способ доказательства высокой стойкости криптоалгоритмов заключается в формальном доказательстве того факта, что атака на алгоритм эквивалентна решению одной из хорошо известных трудноразрешимых задач. Доказательство представляет собой эффективное математиче-

ское преобразование или последовательность преобразований, сводящих атаку на алгоритм к решению задачи-эталона.

Практическая стойкость – свойство алгоритма, доказанное путем формального сведения адаптивных атак к решению трудноразрешимых задач

Современная криптография в значительной степени основана на понятии вычислительной сложности. Задачи разделяются на два класса: разрешимые и трудноразрешимые. Криптографический алгоритм должен быть разработан так, чтобы стать разрешимым для законного пользователя и трудноразрешимым для атакующего. Математическая задача является эффективно разрешимой, если время, необходимое для ее решения, полиномиально зависит от ее размера. Криптографический протокол (процедура обмена данными между участниками) считается эффективным, если количество сеансов (раундов) связи ограничено полиномом малой степени: константой (степень 0) или линейной функцией (степень 1).

Поскольку система ОБИ функционирует во вражеском окружении, при этом даже законный пользователь может иметь злой умысел, ее разработчик должен учитывать множество дополнительных факторов. Выделим следующие [21, 30].

1. Необходимо ясно формулировать все необходимые предположения. Система ОБИ взаимодействует с окружением, которое должно удовлетворять определенным условиям или предположениям. Их нарушение создает предпосылки для проведения атак на КС. Особенно трудно проверить предположение, не сформулированное явно. Иначе говоря, все предположения, на которых основана система ОБИ, должны быть явно указаны.

Например, многие протоколы используют неявные предположения о том, что компьютер может генерировать качественные случайные числа. Однако на практике это часто бывает не так, в результате становится возможной атака на генераторы случайных чисел (low-entropy attack), формирующие ключевую информацию.

2. Необходимо ясно формулировать все предполагаемые услуги по ОБИ, иначе говоря, четко указывать, для решения каких задач предназначен тот или иной алгоритм или протокол. Очень часто требуется уточнение и самих решаемых задач. Например,

когда речь о конфиденциальности информации, может, помимо секретности, предполагаться наличие свойств невидимости или анонимности участников и неотслеживаемости информационных потоков.

Неправильная идентификация задач, стоящих перед алгоритмом или протоколом, может привести соответственно к неправильному использованию криптографических примитивов.

3. Необходимо ясно выделять частные случаи математических задач, на которых основывается стойкость алгоритма или протокола. Например, существуют частные случаи трудноразрешимых задач, которые относительно просто решить.

Например, задача факторизации большого составного целого числа в принципе трудноразрешима. Однако факторизация большого составного целого числа $n = pq$, в котором q является простым числом, следующим за большим простым числом p, вовсе не является трудноразрешимой задачей. Она эффективно решается! Алгебраические структуры, которые используются в криптоалгоритмах и протоколах, такие как группы и поля, имеют специальные варианты, не представляющие никакой вычислительной сложности. Например, элементы, имеющие малый мультипликативный порядок в группе или конечном поле. Другим примером являются вырожденные (суперсингулярные, аномальные) виды эллиптических кривых. Если разработчик алгоритма или протокола не знает о существовании таких особых ситуаций или неточно описал их в спецификации, появляются предпосылки для успешной атаки. Необходимо знать математические особенности решаемой задачи и явно описывать потенциально опасные ситуации.

Ясность и простота описания алгоритма или протокола существенно облегчают его анализ, при этом увеличивается вероятность его правильной реализации и соответственно уменьшается вероятность взлома.

Важной составляющей криптографических исследований является их публичная проверка. Криптографические алгоритмы, протоколы и системы в целом печально известны своей уязвимостью и подверженности ошибкам. После того как результаты криптографических исследований и разработок опубликованы, многочисленные эксперты начинают их испытывать. В результа-

те вероятность, что в ходе таких проверок обнаружатся ошибки в проектировании или реализации, допущенные разработчиками, существенно возрастает. Если же алгоритм сохраняется в секрете, его в лучшем случае проверят несколько экспертов. При этом вероятность выявления ошибок снижается. Возможно ситуация, когда разработчик знает о существовании ошибки и может скрытно воспользоваться этим в своих интересах [21, 30].

Современная наука предоставляет все необходимые алгоритмы, методы и средства, позволяющие построить систему защиты, затраты на взлом которой таковы, что у противника с ограниченными финансовыми и техническими возможностями для получения интересующей его информации остаются только два пути – использование, во-первых, человеческого фактора, а во-вторых, особенностей конкретной реализации (чаще программной) алгоритмов и протоколов удаленного взаимодействия. Именно такой вывод можно сделать, анализируя примеры реальных успешных атак на КС в защищенном исполнении. Известны лишь единичные случаи взлома с использованием исключительно математических методов. В то же время различных примеров взломов реальных систем так много, что их анализом вынуждены заниматься целые компании.

Система защиты в целом не может быть надежнее отдельных ее компонентов. Иными словами, для того чтобы преодолеть систему защиты, достаточно взломать или использовать для взлома самый ненадежный из ее компонентов. Очень часто причинами ненадежности реальных систем защиты являются особенности программной реализации.

Правило слабого звена. Система безопасности надежна настолько, насколько надежно ее самое слабое звено. Слабое звено всегда рвется первым, прочность остальных звеньев после этого уже не имеет значения. Чтобы повысить безопасность системы, необходимо улучшить ее самое слабое звено (ССЗ), иначе говоря, надо определить, из каких звеньев состоит система безопасности и какое из них самое слабое. Для этого необходимо построить и проанализировать иерархическую древовидную структуру, так называемое *дерево атак* [30].

С формальной точки зрения укрепление любого звена, кроме самого слабого, – пустая трата времени. На практике это не так:

злоумышленник может не знать, какое звено самое слабое;

ССЗ может различаться для разных типов злоумышленников;

прочность звена зависит от навыков злоумышленника и имеющихся у него средств.

Таким образом, необходимо укреплять любое звено, которое в определенной ситуации может оказаться самым слабым [30].

Особенностью проектирования системы безопасности является противоборствующее окружение:

противники умны, коварны и изворотливы;

они играют не по правилам;

их поведение непредсказуемо;

неизвестно, кем является злоумышленник, какими знаниями, опытом и ресурсами он обладает, какова его цель и когда он собирается нападать;

противник обладает преимуществом в несколько лет исследований и может воспользоваться новыми технологиями, которых не существовало на момент проектирования;

при этом злоумышленнику, имеющему все перечисленные преимуществами, достаточно найти всего лишь одно слабое звено в атакуемой системе, в то время как разработчикам надо защищать ее всю [30].

Заниматься проектированием безопасных систем и уметь находить слабые места в собственных разработках сможет лишь тот, кто сам начнет думать, как злоумышленник.

При работе в области практической криптографии очень полезна *параноидальная модель*. Например, только такая модель должна использоваться при создании электронных платежных систем (ЭПС), объединяющих покупателей, продавцов и банки. Для каждого участника ЭПС предполагается, что остальные участники объединились с целью его обмануть. Если криптосистема способна выстоять в такой ситуации, у нее есть шанс выжить в реальном мире.

Важно знать, от чего мы пытаемся защищаться. В чем состоит угроза? Так, например, большинство компаний защищают свои информационные системы (ИС) межсетевыми экранами (МЭ), в то время как наиболее разрушительные атаки обычно исходят

изнутри. В этой ситуации МЭ бесполезны. Это пример неудачного составления *модели угроз* [30].

Стохастические методы защиты. Стохастическими методами защиты в широком смысле принято называть методы защиты компьютерных систем, прямо или косвенно основанные на использовании генераторов псевдослучайных чисел (ПСЧ) и хеш-генераторов, при этом эффективность защиты в значительной степени определяется качеством используемых алгоритмов генерации ПСЧ и алгоритмов хеширования. Криптографические методы защиты, таким образом, являются всего лишь частным случаем стохастических методов.

Генераторы ПСЧ и хеш-генераторы успешно решают практически все упомянутые выше задачи, стоящие перед разработчиками систем ОБИ. При реализации большинства методов защиты используются генераторы ПСЧ и (или) хеш-генераторы. Иначе говоря, эти методы являются стохастическими. Более того, наиболее перспективный метод защиты, а именно: метод внесения неопределенности в работу программных систем (реализация которого в принципе невозможна без использования генераторов ПСЧ), является универсальным. Он может использоваться совместно с любым другим методом защиты, автоматически повышая его качество.

В книге обосновывается утверждение об универсальности стохастических методов (которые имеют двойное назначение, поскольку используются как при атаках, так и при организации защиты КС) и определяющей роли генераторов ПСЧ в системах ОБИ. После обоснования утверждения о том, что роль генераторов ПСЧ при построении системы ОБИ является решающей, возникает возможность подойти к решению задач защиты КС с единых исходных позиций. Ранее научные дисциплины, в той или иной степени связанные с решением задач ОБИ, развивались обособленно друг друга. Речь в первую очередь идет о технической диагностике, теории помехоустойчивого кодирования, крипто- и стеганографии, информационной безопасности, технологии безопасного программирования.

ЧАСТЬ 1. ВВЕДЕНИЕ В КРИПТОЛОГИЮ

ГЛАВА 1. ОСНОВЫ КРИПТОЛОГИИ

1.1. Основные термины и определения

Принято считать, что криптография – наука о шифрах. Это далеко не так. Возможности криптографии значительно шире. Среди них:

обеспечение конфиденциальности информации путем шифрования передаваемых сообщений и хранимых данных для защиты от утечки информации;

обеспечение аутентичности (целостности и подлинности) объектов информационного взаимодействия (передаваемых сообщений и хранимых данных), т.е. обнаружение их случайных или преднамеренных искажений и защита от навязывания фальсифицированной информации;

обеспечение аутентичности (подлинности) субъектов информационного взаимодействия (удаленных абонентов, технических средств и носителей информации);

защита программ от несанкционированного копирования и распространения;

защита от несанкционированного доступа (НСД) к информации, в частности путем организации парольных систем.

Криптография включает в себя следующие основные разделы:

криптосистемы с секретным ключом (классическая криптография);

криптосистемы с открытым ключом (современная криптография);

криптографические протоколы;

управление ключами.

Основной целью *криптографической защиты* или *криптографического закрытия* информации является защита от утечки информации, которая обеспечивается путем обратимого однозначного преобразования сообщений или хранящихся данных в форму, непонятную для посторонних или *неавторизованных* лиц. Преобразование, обеспечивающее криптозащиту, называется *шифрованием*. Защита от модификации информации и навя-

зывания ложных данных, т.е. *имитозащита*, обеспечивается выработкой *имитоприставки*. Последняя представляет собой информационную последовательность, полученную по определенным правилам из открытых данных и ключа.

В отличие от криптографии, которая считает, что сообщение в зашифрованном виде может быть доступно незаконному пользователю, *стеганография* скрывает сам факт хранения или передачи секретной информации. Компьютерная стеганография, позволяющая прятать секретные файлы внутри графических и звуковых файлов, основывается на свойстве этих файлов видоизменяться без потери функциональности и на неспособности человеческих органов чувств различать незначительные изменения в цвете изображения или качестве звука.

Введем некоторые понятия, необходимые в дальнейшем:

алфавит – конечное множество используемых для шифрования информации знаков;

текст – упорядоченный набор из элементов алфавита;

шифр – совокупность обратимых преобразований множества открытых данных на множество закрытых (зашифрованных) данных, заданных алгоритмом криптографического преобразования (*криптоалгоритмом*);

ключ – сменный элемент шифра, применяемый для закрытия отдельного сообщения, т.е. конкретное секретное состояние параметров криптоалгоритма, обеспечивающее выбор одного варианта преобразования из совокупности возможных; именно ключом в первую очередь определяется безопасность защищаемой информации, и поэтому применяемые в качественных шифрах преобразования сильно зависят от ключа;

зашифрование – преобразование открытых данных в закрытые (зашифрованные) с помощью определенных правил, содержащихся в шифре; *расшифрование* – обратный процесс;

шифрование – процесс зашифрования или расшифрования;

криптограмма – зашифрованный текст;

криптосистема – совокупность алгоритмов зашифрования и расшифрования информации, а также генерации ключей;

дешифрование – процесс преобразования закрытых данных в открытые при неизвестном ключе и (или) неизвестном алгоритме (*вскрытие* или *взламывание шифра*);
синхропосылка – исходный параметр криптоалгоритма;
раскрытие криптоалгоритма – результат работы криптоаналитика, приводящий к возможности эффективного определения любого зашифрованного с помощью данного алгоритма открытого текста;
стойкость криптоалгоритма – способность шифра противостоять всевозможным попыткам его раскрытия, т.е. *атакам* на него.

1.2. Оценка надежности криптоалгоритмов

Все современные шифры базируются на принципе Кирхгофа, согласно которому секретность шифра обеспечивается секретностью ключа, а не секретностью алгоритма шифрования. В некоторых ситуациях (например, в военных, разведывательных и дипломатических ведомствах) нет никаких причин делать общедоступным описание сути криптосистемы. Сохраняя такую информацию в тайне, можно дополнительно повысить надежность шифра. Однако полагаться на секретность этой информации не следует, так как рано или поздно она будет скомпрометирована. Поэтому анализ надежности таких систем всегда должен проводиться исходя из того, что *противник имеет всю информацию о применяемом криптоалгоритме, ему не известен только реально использованный ключ*. Таким образом, можно сформулировать общее правило: *при создании или при анализе стойкости криптосистем не следует недооценивать возможностей противника. Их лучше переоценить, чем недооценить*.

Стойкость криптосистемы зависит от сложности алгоритмов преобразования, длины ключа, а точнее, от объема *ключевого пространства*, метода реализации: при программной реализации необходимо дополнительно защищаться от *разрушающих программных воздействий* (закладок, вирусов и т.п.). Хотя понятие стойкости шифра является центральным в криптографии, коли-

чественная оценка криптостойкости – проблема до сих пор нерешенная.

Методы оценки качества криптоалгоритмов, используемые на практике [4]:

всевозможные попытки их вскрытия;

анализ сложности алгоритма дешифрования;

оценка статистической безопасности шифра.

В первом случае многое зависит от квалификации, опыта, интуиции *криптоаналитиков* и правильной оценки возможностей противника. Обычно считается, что противник знает шифр, имеет возможность его изучения, знает некоторые характеристики открытых защищаемых данных, например тематику сообщений, их стиль, стандарты, форматы и т.п. В [4] приводятся следующие примеры возможностей противника, который:

может перехватывать все зашифрованные сообщения, но не имеет соответствующих им открытых текстов;

может перехватывать все зашифрованные сообщения и добывать соответствующие им открытые тексты;

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

Во втором случае оценку стойкости шифра заменяют оценкой минимальной сложности алгоритма его вскрытия. Однако получить строго доказуемые оценки нижней границы сложности алгоритмов рассматриваемого типа не представляется возможным. Иными словами, всегда возможна ситуация, когда алгоритм вскрытия шифра, сложность которого анализируется, оказывается вовсе не самым эффективным.

Сложность вычислительных алгоритмов можно оценивать числом выполняемых элементарных операций, при этом, естественно, необходимо учитывать их стоимость и затраты на их выполнение. В общем случае это число должно иметь строгую нижнюю оценку и выходить за пределы возможностей современных компьютерных систем. *Качественный шифр невозможно раскрыть способом более эффективным, чем полный перебор по всему ключевому пространству, при этом разработчик дол-*

жен рассчитывать только на то, что у противника не хватит времени и ресурсов, чтобы это сделать.

Алгоритм полного перебора по всему ключевому пространству – это пример так называемого *экспоненциального* алгоритма. Если сложность алгоритма выражается неким многочленом (полиномом) от n, где n – число элементарных операций, такой алгоритм носит название *полиномиального*. Пример полиномиального алгоритма – алгоритм умножения столбиком двух n-разрядных двоичных чисел, требующий выполнения n^2 элементарных, битовых, операций умножения [4].

В третьем случае считается, что надежная криптосистема с точки зрения противника является «черным ящиком», входная и выходная информационные последовательности которого взаимно независимы, при этом выходная зашифрованная последовательность является *псевдослучайной*. Поэтому смысл испытаний заключается в проведении статистических тестов, устанавливающих зависимость изменений в зашифрованном тексте от изменений символов или бит в исходном тексте или ключе, а также анализирующих, насколько выходная зашифрованная последовательность по своим статистическим свойствам приближается к истинно случайной последовательности.

Случайность текста шифрограммы можно приближенно оценивать степенью ее сжатия при использовании алгоритма Лемпела–Зива, применяемого в архиваторах IBM PC. Если степень сжатия больше 10 %, то криптосистема несостоятельна.

Необходимые условия стойкости криптосистемы, проверяемые статистическими методами:

отсутствие статистической зависимости между входной и выходной последовательностями;

выходная последовательность по своим статистическим свойствам должна быть похожа на истинно случайную последовательность;

при неизменной входной информационной последовательности любое (в том числе незначительное) изменение ключа должно приводить к существенному непредсказуемому из-

менению выходной последовательности (в среднем должно измениться 50 % бит);

при неизменном ключе любое (в том числе незначительное) изменение входной последовательности должно приводить к существенному непредсказуемому изменению выходной последовательности (в среднем должно измениться 50 % бит);

не должно быть зависимостей между ключами, последовательно используемыми в процессе шифрования.

Существует много различных криптоалгоритмов, при этом нет ни одного, подходящего для всех случаев. В каждой конкретной ситуации *выбор криптоалгоритма* определяется следующими факторами:

особенностью защищаемой информации (документы, исходные тексты программ, графические файлы и т.п.);

особенностями среды хранения или передачи информации;

ценностью информации, характером защищаемых секретов, временем обеспечения секретности;

объемами информации, скоростью ее передачи, степенью оперативности ее предоставления пользователю;

возможностями собственников информации, владельцев средств сбора, обработки, хранения и передачи информации по ее защите;

характером угроз, возможностями противника.

1.3. История криптологии

Криптография и криптоанализ имеют многовековую историю развития и применения [4, 10, 11, 27]. Можно выделить три периода развития криптологии:

1) эра донаучной криптологии (длилась с незапамятных времен до середины XX в.), когда последняя была занятием чудаков-одиночек, среди которых были полководцы, цари, ученые, дипломаты, священнослужители;
2) эра классической криптографии (криптографии с секретным ключом) – становление криптологии как научной дисциплины, связанной с разработкой шифров и оценкой

их стойкости (1949–1976 гг.); начало этому периоду положила работа К. Шеннона «Теория связи в секретных системах» (Шеннон К.Э. Работы по теории информации и кибернетике. М.: ИЛ, 1963, с. 243–332) (в [4] имеется фрагмент этой работы);

3) эра современной криптографии, эра бурного развития наряду с классической криптографией криптографии с открытым ключом, появление которой стимулировало рождение новых направлений в математике; начало этому периоду положила работа У. Диффи, М. Хеллмана «Новые направления в криптографии» (IEEE Trans. Inform. Theory, IT-22, 1976, pp. 644–654).

1.4. Классификация методов шифрования информации

Основные объекты изучения классической криптографии показаны на рис. 1.1, где A и B – законные пользователи, W – противник или криптоаналитик. Учитывая, что схема на рис. 1.1,*а* фактически является частным случаем схемы на рис. 1.1,*б* при $B = A$, в дальнейшем будет рассматриваться только она.

Процедуры зашифрования (encryption) и расшифрования (decryption) можно представить в следующем виде:

$$c = E_k(m),$$
$$m = D_k(c),$$

где m и c – открытый (plaintext) и зашифрованный (ciphertext) тексты, k_e и k_d – ключи зашифрования и расшифрования, E_k и D_k – функции зашифрования с ключом k_e и расшифрования с ключом k_d соответственно, причем для любого открытого текста m справедливо

$$D_k(E_k(m)) = m.$$

На рис. 1.2 приведена классификация методов шифрования информации. Различают два типа алгоритмов шифрования *симметричные* (*с секретным ключом*) и *асимметричные* (*с откры-*

тым ключом). В первом случае обычно ключ расшифрования совпадает с ключом зашифрования, т.е.

$$k_e = k_d = k,$$

либо знание ключа зашифрования позволяет легко вычислить ключ расшифрования. В асимметричных алгоритмах такая возможность отсутствует: для зашифрования и расшифрования используются разные ключи, причем знание одного из них не дает практической возможности определить другой. Поэтому если получатель A информации сохраняет в секрете ключ расшифрования $k_{dA} = k_A^{(secret)}$, ключ зашифрования $k_{eA} = k_A^{(public)}$ может быть сделан общедоступным

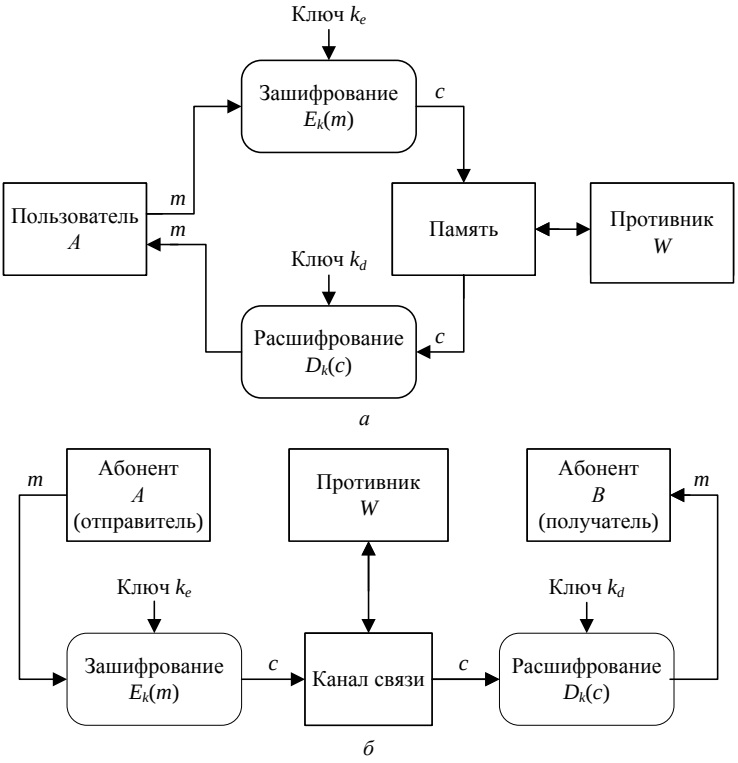


Рис. 1.1. Криптозащита:
а – при хранении; *б* – при передаче информации по каналу связи

В процессе шифрования информация делится на порции величиной от одного до сотен бит. Поточные (иначе называемые потоковыми) шифры могут оперировать порциями данных любой разрядности, так существуют поточные шифры, обрабатывающие байты (например, RC4), порции данных размерностью 64 бита (например, Chameleon), но чаще всего поточные шифры оперируют с битами открытого и закрытого текстов. Существенное отличие между этими двумя методами шифрования заключается в том, что в блочных шифрах для шифрования всех порций данных (блоков фиксированной длины) используется один и тот же ключ, а в поточных – для каждой порции используется свой ключ той же размерности. Иначе говоря, в поточных шифрах имеет место зависимость результата шифрования порции информации от ее позиции в тексте, а в некоторых случаях и от результатов шифрования предыдущих порций текста. Таким образом, при реализации поточной криптосистемы возникает необходимость в элементах памяти, изменяя состояние которых, можно вырабатывать последовательность (поток) ключевой информации. Блочную же криптосистему можно рассматривать как зависящую от ключа подстановку на множестве значений блоков открытого текста [2].

Основной критерий эффективности при проектировании поточных шифров – быстродействие (чаще всего в ущерб криптостойкости), при проектировании блочных шифров – криптостойкость (чаще всего в ущерб быстродействию). Требование высокой скорости преобразования при использовании поточных шифров определяется областью их использования – шифрованием данных, требующих оперативной доставки потребителю (например, аудио- и видеоинформации). В случае поточных криптоалгоритмов шифрование осуществляется в реальном масштабе времени или близком к нему. В блочных же шифрах преобразование информации начинается только при поступлении всей порции информации, равной разрядности блока.

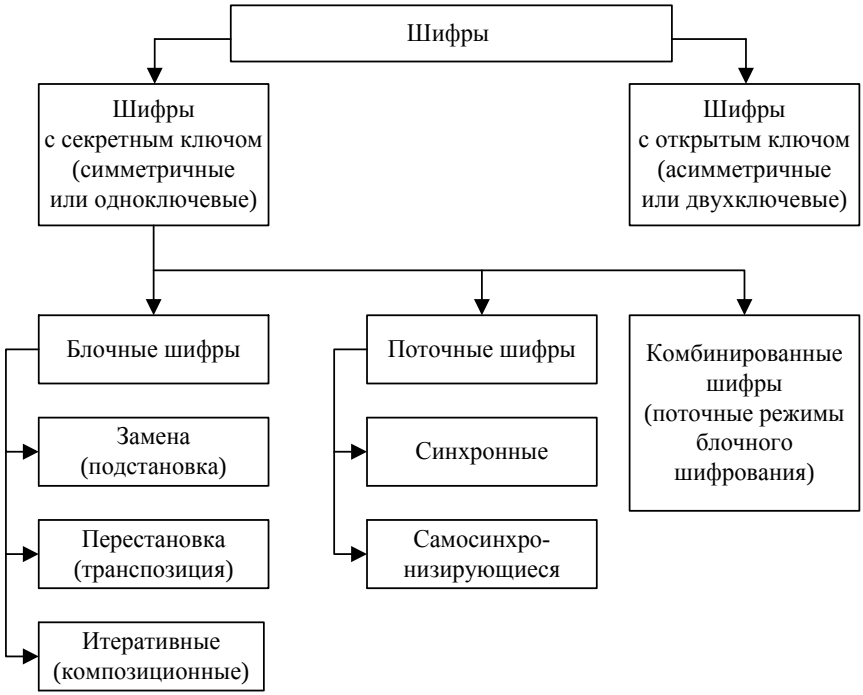


Рис. 1.2. Классификация методов шифрования информации

Учитывая то, что в случае применения классических блочных шифров одинаковым блокам открытого текста соответствуют одинаковые блоки шифротекста, а это является серьезным недостатком, на практике получили наибольшее распространение комбинированные методы шифрования, использующие один из следующих принципов:

«сцепления» блоков;

формирование потока ключей (*гаммы* шифра) с помощью так называемых *генераторов псевдослучайных чисел* (ПСЧ), в качестве функции обратной связи (функции переходов) или функции выхода которых применяется функция зашифрования блочного шифра.

1.5. Шифры замены

Наиболее известны шифры *замены* или *подстановки*, особенностью которых является замена символов (или слов, или других частей сообщения) открытого текста соответствующими символами, принадлежащими алфавиту шифротекста. Различают *одноалфавитную* и *многоалфавитную* замену. Вскрытие одноалфавитных шифров основано на учете частоты появлений отдельных букв или их сочетаний (биграмм, триграмм и т.п.) в данном языке. Классические примеры вскрытия таких шифров содержатся в рассказах Эдгара По («Золотой жук») и Артура Конан Дойла («Пляшущие человечки»).

Пример многоалфавитного шифра замены – так называемая *система Виженера*. Шифрование осуществляется по таблице, представляющей собой квадратную матрицу размерностью $n \times n$, где n – число символов используемого алфавита. Таблица Виженера для русского языка (алфавит Z_{33} – 32 буквы и пробел) имеет размер 33 × 33. 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»

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

ГРУЗИТЕ АПЕЛЬСИНЫ БОЧКАМИ
ВЕНТИЛЬВЕНТИЛЬВЕНТИЛЬВЕНТ

В результате зашифрования, начальный этап которого показан на рис. 1.3, получим шифротекст

«ЕХ РЭЯБЕЫЧУДККТИСЙЩРМЕЩЬЗ»

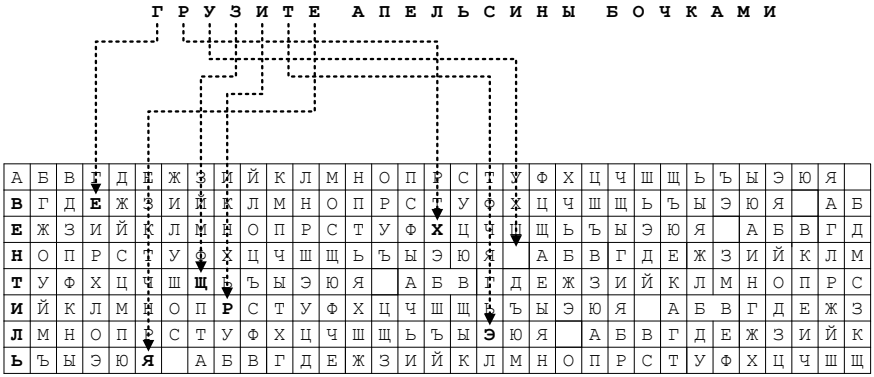


Рис. 1.3. Принцип шифрования по таблице Виженера

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 строки 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.

1.6. Шифры 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. 1.4. 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 изготовления 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.

Рассмотрим *усложненную перестановку по таблице* [11]. Пример таблицы для реализации этого метода шифрования показан на рис. 1.5. Таблица представляет собой матрицу размерностью 6 × 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»

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 4 6 2 5 3 1.

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

Рис. 1.4. Пример поворотной решетки

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. 1.5. Пример шифрования методом усложненной перестановки по таблице

1.7. Шифр Ф. Бэкона

Особое место занимает двухлитерный шифр Ф. Бэкона, который за счет присущей ему избыточности может при соответствующем выборе ключевой информации иметь «второе» и даже «третье дно» (рис. 1.6). Рассмотрим его реализацию для символов английского алфавита (столбец 1 на рис. 1.6). Структура ключевой информации имеет следующий вид. Каждому символу исходного алфавита ставится во взаимно-однозначное соответствие двухлитерный пяти символьный код (столбец 2 на рис. 1.6). Затем каждому символу исходного алфавита ставится в соответствие одна из двух литер «a» или «b», на рис. 1.7 в качестве примера показаны три возможных варианта выбора: k_1, k_2 и k_3. Общее число возможных вариантов выбора равно $2^{26} - 1$, при этом предпочтительными являются те из них, которые содержат приблизительно одинаковое количество литер «a» и «b». Указанное соответствие выбирается либо по принципу удобства запоминания (столбцы 3 и 4), либо случайным образом (столбец 5). Очевидно, что второй способ выбора k более предпочтителен.

Итак, предположим, что выбрана ключевая информация, показанная на рис. 1.6, при этом только один из ключей k – истинный, предположим, что таковым является k_1.

Символы алфавита	Первая форма представления				Вторая форма представления			
	Двухлитерный код	k_1	k_2	k_3	Таблица замен	k_1	k_2	k_3
1	2	3	4	5	6	7	8	9
A	abbbb	a	b	b	10000	1	0	0
B	babbb	a	a	b	01000	1	1	0
C	bbabb	a	b	a	00100	1	0	1
D	abbab	a	a	b	10010	1	1	0
E	babba	a	b	a	01001	1	0	1
F	ababb	a	a	a	10100	1	1	1
G	aabab	a	b	b	11010	1	0	0
H	baaba	a	a	b	01101	1	1	0
I	bbaab	a	b	b	00110	1	0	0
J	abbaa	a	a	a	10011	1	1	1
K	aabba	a	b	a	11001	1	0	1
L	aaabb	a	a	b	11100	1	1	0
M	aaaab	a	b	a	11110	1	0	1
N	aaaaa	b	a	b	11111	0	1	0
O	baaaa	b	b	b	01111	0	0	0
P	bbaaa	b	a	a	00111	0	1	1
Q	bbbaa	b	b	a	00011	0	0	1
R	abbba	b	a	b	10001	0	1	0
S	aabbb	b	b	b	11000	0	0	0
T	baabb	b	a	a	01100	0	1	1
U	abaab	b	b	b	10110	0	0	0
V	aabaa	b	a	a	11011	0	1	1
W	aaaba	b	b	a	11101	0	0	1
X	baaab	b	a	a	01110	0	1	1
Y	abaaa	b	b	b	10111	0	0	0
Z	babaa	b	a	b	01011	0	1	0

Рис. 1.6. Две формы задания шифра Ф. Бэкона

За исходное сообщение примем слово CAT. Шифротекст формируется за два шага. На первом шаге каждый символ исходного текста заменяется его двухлитерным кодом, в рассматриваемом случае получим bbabb abbbb baabb. На втором шаге в

соответствии с ключом k_1 вместо литер «a» и «b» записываются символы исходного алфавита, при этом очевидно, что число способов записи и получения в результате шифровки очень велико. Иначе говоря, одному исходному тексту соответствует огромное множество различных шифротекстов. Выбираем шифротекст следующего вида: NOCVW ARTVZ PFGVY. При желании можно попытаться получить осмысленный шифротекст.

Процедура расшифрования также выполняется за два шага, но в обратной последовательности. На первом шаге, имея шифротекст NOCVW ARTVZ PFGVY, заменяем символы исходного алфавита на литеры «a» и «b» в соответствии с ключом k_1. После этого на втором шаге находим в столбце 2 полученные двухлитерные коды и записываем соответствующие им символы исходного алфавита. После чего получаем правильный исходный текст CAT.

При появлении необходимости задействовать «второе дно» истинным ключом объявляется k_2. В результате на первом шаге шифротексту NOCVW ARTVZ PFGVY соответствует последовательность кодов abbab baaaa aabab, которая на втором шаге превращается в ложный исходный текст DOG.

При появлении необходимости задействовать «третье дно» истинным ключом объявляется k_3. В результате на первом шаге шифротексту NOCVW ARTVZ PFGVY соответствует последовательность кодов bbaaa bbaab aabab, которая на втором шаге превращается в ложный исходный текст PIG.

Столбцы 6–9 на рис. 1.6 содержат более привычное задание шифра за счет замены литеры «a» на «1» и литеры «b» на «0».

1.8. Блочные составные шифры

В общем случае детерминированный шифр G определяется следующим образом:

$$G = (M, C, K, F),$$

где M – множество входных значений, C – множество выходных значений, K – пространство ключей, F – функция зашифрования:

$$F: M \times K \to C.$$

Пусть составной шифр определяется семейством преобразований G_i, имеющих общие пространства входных и выходных значений, т.е. $M_i = C_i = M$, функции F_i, определяемые ключевыми элементами $k_i \in K_i$. На основе этого семейства с помощью операции *композиции* можно построить шифр, задаваемый отображением

$$F: M \times (K_1 \times K_2 \times ... \times K_r) \to M,$$

причем

$$F = F_r \bullet ... \bullet F_2 \bullet F_1,$$

а ключом является вектор

$$(k_1, k_2, ..., k_r) \in K_1 \times K_2 \times ... \times K_r.$$

Преобразование F_i называется i-м *раундом шифрования*, ключ k_i – *раундовым ключом*. В некоторых случаях раундовые ключи получаются из ключа всей системы с помощью *алгоритма выработки раундовых ключей* (при этом размер ключа системы существенно меньше суммарного размера всех раундовых ключей). Если ключевые пространства K_i и преобразования F_i для всех раундов совпадают, такой составной шифр называется *итеративным*, представляющим собой композицию одной и той же криптографической функции, используемой с разными ключами [2].

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. В [10, 27] отмечается, что действие таких шифров аналогично «алгоритму», к которому прибегают, когда месят тесто:

РАСКАТАТЬ
СЛОЖИТЬ
РАСКАТАТЬ
СЛОЖИТЬ
РАСКАТАТЬ
СЛОЖИТЬ
. . . . . .

Многократное использование этих преобразований (рис. 1.7) позволяет обеспечить два свойства, которые должны быть присущи стойким шифрам: *рассеивание* (diffusion) и *перемешивание* (confusion). Рассеивание предполагает распространение влияния одного знака открытого текста, а 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, и не позволяет восстанавливать неизвестный ключ по частям.

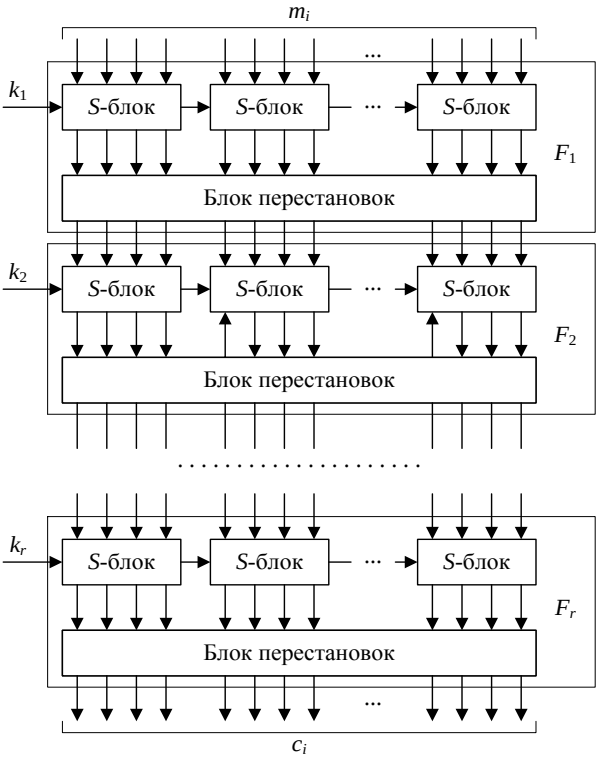


Рис. 1.7. Вариант простейшего итеративного шифра (*SP*-сеть)

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 сколь-нибудь значительное количество 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 преобразования 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. Каждое раундовое преобразование может являться слабым с криптографической точки зрения. Единственное ограничение при построении составного шифра заключается в запрете на использование в двух соседних раундах шифрования преобразований, имеющих общую *прозрачность*.

Пусть $F: x \to y,\ x, y \in M,$ и на множестве M определены преобразования g и h. Если $F(g(x)) = h(y),$ F прозрачно для g, а g – для F.

Примерами прозрачных операций могут служить операции циклического сдвига, замены и т. п. Если два преобразования, выбранные в качестве соседних раундов, имеют общую прозрачность g, и при этом существует простое преобразование, не прозрачное для g, данное преобразование следует поместить между двумя раундами шифрования и полученная композиция уже не будет прозрачной для g. Такие преобразования, чаще всего не зависящие от ключа, называются *буферами*. Помимо внутренних

иногда применяют и внешние буфера, выполняющие преобразования, зависящие или не зависящие от ключа.

Важное достоинство многих составных шифров – 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 становится известными каких-либо промежуточные результаты преобразований. По этой причине использование такой информации при криптоанализе составных шифров некорректно [2].

Рассмотрим структуру раунда блочного шифра, получившую название *петли Фейстеля* (рис. 1.8). Представим шифруемый блок данных (открытого m_i или закрытого c_i текста) длиной n в виде пары полублоков в два раза меньшего размера:

$$\left| m_i \right| = \left| c_i \right| = n,\ m_i = c_i = (L, R), \left| L \right| = \left| R \right| = n/2.$$

Зашифруем старший полублок L (Left) блока m_i с помощью младшего R (Right), используя некоторую функцию f_i, зависящую от раундового ключа k_i, и обратимую бинарную операцию $\circ$ над $n/2$-битовыми блоками данных. Для подготовки к следующему аналогичному раунду осуществимим перестановку частей блока m_i: $L \circ f_i(R) \leftrightarrow R$. Таким образом, раундовая функция зашифрования (рис. 1.9) будет иметь вид

$$F_i(m_i) = F_i\ (L, R) = (R, L \circ f_i(R)),$$

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 $\bullet$ – операция, обратная $\circ$. Композиционный шифр, использующий раундовые функции такого вида, называется шифром Фейстеля. В подавляющем большинстве шифров рассматриваемой структуры в качестве операций $\circ$ и $\bullet$ используется поразрядное сложение по модулю два (XOR).

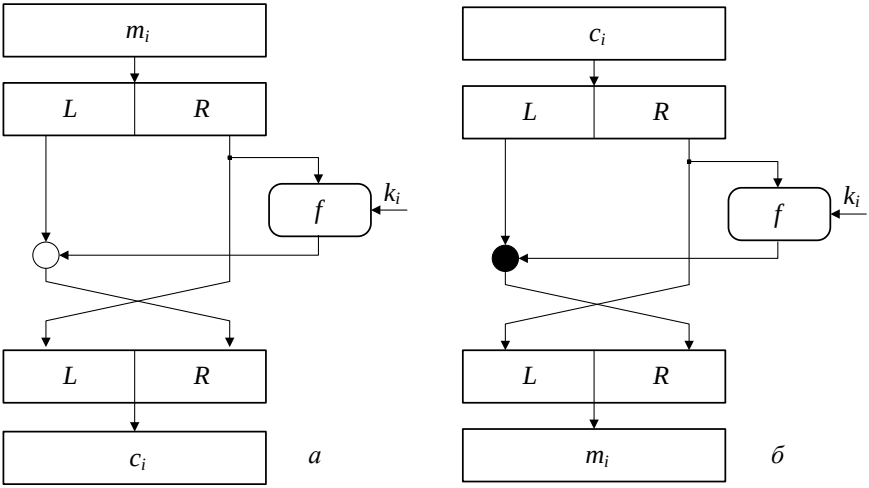


Рис. 1.8. Схема сбалансированной петли Фейстеля:
а – зашифрование, *б* – расшифрование

Первыми широко известными практическими реализациями итеративного блочного шифра были разработанные сотрудниками фирмы IBM (Х. Фейстелем, Д. Копперсмитом и др.) 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 использованием 56-разрядного ключа, из которого по специальному *фиксированному* алгоритму, использующему перестановки и сдвиги, вырабатываются раундовые ключи. Применяемые преобразования – поразрядное сложение по модулю два, подстановки и перестановки, число раундов равно 16. Перед началом первого раунда выполняется начальная фиксированная перестановка *IP*, после 16-го раунда – обратная перестановка IP^{-1}. Следуя рекомендациям Шеннона, в каждом раунде осуществляется один шаг перемешивания (с использованием соответствующего раундового ключа и *S*-блоков), после которого следует шаг рассеивания, не зависящий от ключа.

Интересно отметить: в первоначальной схеме, предложенной IBM, все шестнадцать 48-разрядных раундовых ключей выбирались независимо, т.е. размер ключа был равен 768 битам. Однако, по требованию Агенства национальной безопасности США

(АНБ), во-первых, размер ключа был уменьшен до 64 битов, из которых только 56 – секретные, во-вторых, в алгоритме определены перестановки лишь специального вида, не зависящие от ключа. А это наводило критиков данного 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) по схемам, показанным на рис. 1.9.

1.9. Абсолютно стойкий шифр. Гаммирование

Простейшей и в то же время наиболее надежной из всех схем шифрования является так называемая *схема однократного использования* (рис. 1.10), изобретение которой чаще всего связывают с именем Г.С. Вернама [10, 27]. Формируется t-разрядная случайная двоичная последовательность – ключ шифра, известный отправителю и получателю сообщения. Отправитель производит побитовое сложение по модулю два ключа и t-разрядной двоичной последовательности, соответствующей пересылаемому сообщению:

$$c_i = m_i \oplus k_i,\ i = \overline{1,\ t},$$

где m_i, k_i, c_i – очередные i-е биты соответственно исходного сообщения, ключа и зашифрованного сообщения; t – число битов открытого текста. Процесс расшифрования сводится к повторной генерации ключевой последовательности и наложения ее на зашифрованные данные. Уравнение расшифрования имеет вид

$$m_i = c_i \oplus k_i,\ i = \overline{1,\ t}.$$

Клодом Шенноном доказано, что если ключ является фрагментом истинно случайной двоичной последовательностью с равномерным законом распределением (причем его длина равна длине исходного сообщения) и только один раз, после чего уничтожается, такой шифр – *абсолютно стойкий*, его невозможно раскрыть, даже если криптоаналитик располагает неограниченными запасами времени и вычислительных ресурсов. Дейст-

вительно, противнику известно только зашифрованное сообщение c, при этом все различные ключевые последовательности k возможны и равновероятны, а значит, возможны и любые сообщения m, т. е. *криптоалгоритм не дает никакой информации об открытом тексте*.

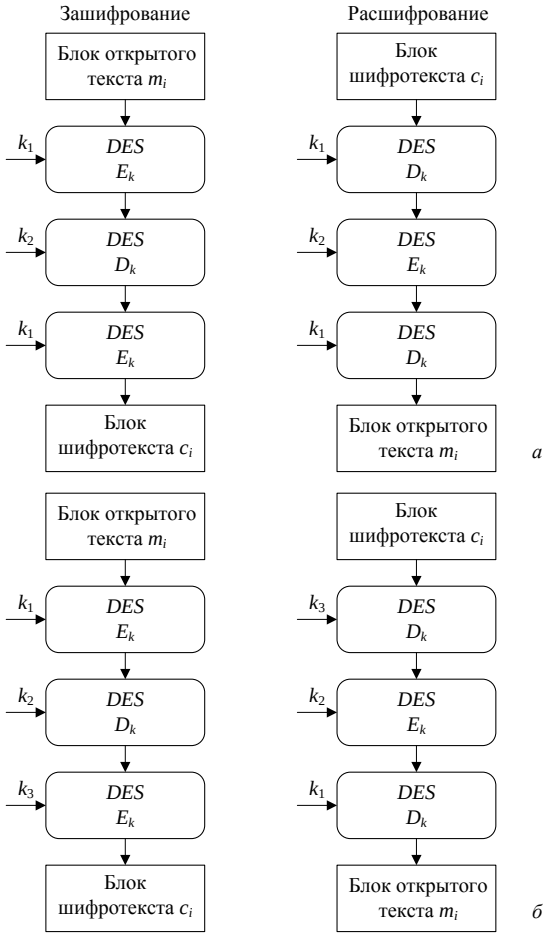


Рис. 1.9. Схемы трехкратного использования алгоритма DES: *а* – с двумя ключами; *б* – с тремя ключами

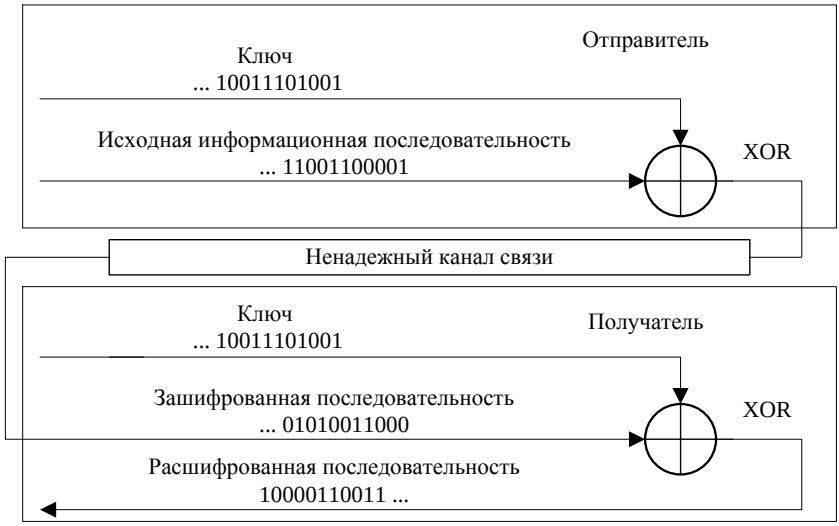


Рис. 1.10. Схема однократного использования

Целью противника может являться раскрытие криптосистемы, нахождение ключа, в крайнем случае, дешифрование какого-либо закрытого сообщения. Однако его может удовлетворить получение даже некоторой вероятностной информации об исходном тексте сообщения. Например, известный криптоаналитику факт написания текста некоторого сообщения на английском языке предоставляет ему некоторую *априорную* информацию об этом сообщении даже до анализа шифровки. В данном случае он заранее знает, что слово «HELLO» – *более вероятное* начало сообщения, чем набор букв «FGHKM». Поэтому одной из целей криптоанализа может стать увеличение информации, относящейся к каждому возможному сообщению таким образом, чтобы правильный текст был более вероятен. Предположим, противник перехватил шифровку «ABCCD» и знает (или предполагает), что использованный шифр – это шифр простой замены. Анализ шифровки позволяет сделать вывод: исходное сообщение состоит из пяти букв, причем на третьей и четвертой позициях стоит одна и та же буква, а остальные отличны от нее и различны между собой. Противник

не может считать, что это сообщение «HELLO», поскольку имеются и другие возможные сообщения, например «TEDDY». Однако *апостериорные* вероятности этих открытых текстов возрастают относительно их *априорных* вероятностей. В то же время *апостериорная* вероятность таких открытых текстов, как «PEACE» или «GATES», снижается до нуля вне зависимости от их *априорной* вероятности. По Шеннону, в совершенно секретных криптосистемах после анализа закрытых текстов *апостериорные* вероятности возможных открытых текстов остаются такими же, какими были их *априорные* вероятности [3].

Необходимые и достаточные условия абсолютной стойкости шифра:

полная случайность ключа;
равенство длин ключа и открытого текста;
однократное использование ключа.

Абсолютная стойкость рассмотренной схемы оплачивается слишком большой ценой, она чрезвычайно дорога и непрактична. Основной ее недостаток – равенство объема ключевой информации и суммарного объема передаваемых сообщений. Применение схемы оправдано лишь в нечасто используемых каналах связи для шифрования исключительно важных сообщений. Существует большое число модификаций представленной схемы, наиболее известная из которых основана на использовании одноразовых шифровальных блокнотов (рис. 1.11).

Таким образом, построить эффективный криптоалгоритм можно, лишь отказавшись от абсолютной стойкости. Возникает задача разработки теоретически нестойкого шифра, для вскрытия которого противнику потребовалось бы выполнить число операций, осуществимое на современных и ожидаемых в ближайшей перспективе вычислительных средствах за разумное время. В первую очередь следует иметь схему, которая использует ключ небольшой разрядности, в дальнейшем выполняющий функцию «зародыша», порождающего значительно более длинную ключевую последовательность.

Данный результат может быть достигнут при использовании *гаммирования*, схема которого показана на рис. 1.12. Гаммированием называют процедуру наложения на входную информационную последовательность *гаммы* шифра, т.е. последовательности с выходов *генератора псевдослучайных чисел*. Последовательность чисел называется *псевдослучайной*, если по своим статистическим свойствам она неотличима от истинно *случайной*, но, в отличие от последней, является детерминированной, т.е. знание алгоритма формирования дает возможность ее повторения необходимое число раз. Если символы входной информационной последовательности и гаммы представлены в двоичном виде, наложение чаще всего реализуется с помощью операции поразрядного сложения по модулю два. Надежность шифрования методом гаммирования определяется качеством генератора гаммы.

Различают гаммирование с *конечной* и *бесконечной* гаммами. В первом случае источник гаммы – аппаратный или программный генератор ПСЧ. В качестве примера бесконечной гаммы можно привести последовательность цифр в десятичной записи числа $\pi = 3{,}1415926\ldots$.

Если множеством используемых для шифрования знаков является алфавит, отличный от бинарного ($Z_2 = \{0, 1\}$), например, алфавит Z_{33} – русские буквы и пробел, то его символы и символы гаммы заменяются цифровыми эквивалентами, которые затем суммируются по модулю N:

$$c_i = (m_i + \gamma_i) \bmod N,\ i = \overline{1,\ t},$$

где m_i, γ_i, c_i – очередной i-й знак исходного сообщения, гаммы и шифротекста соответственно; t – число знаков открытого текста; N – число символов в алфавите. Возможно использование при гаммировании и других логических операций.

Буква	А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М
Код	0	1	2	3	4	5	6	7	8	9	10	11	12
Буква	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ
Код	13	14	15	16	17	18	19	20	21	22	23	24	25
Буква	Ь	Ы	Ъ	Э	Ю	Я							
Код	26	27	28	29	30	31	32						

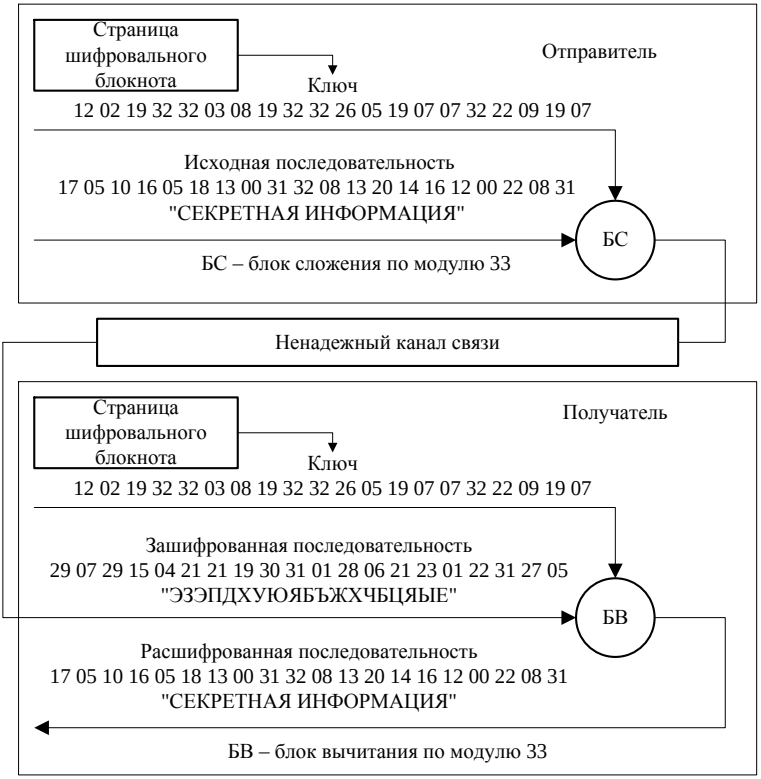


Рис. 1.11. Система однократного использования
на основе шифровального блокнота

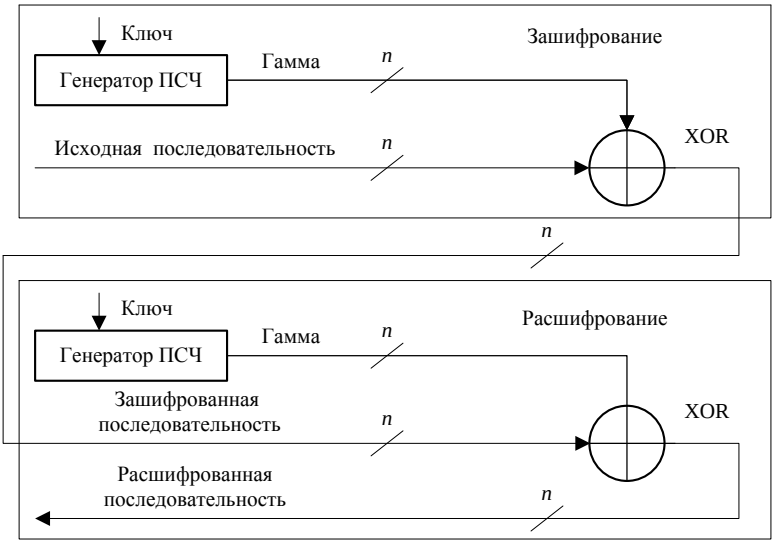


Рис. 1.12. Шифрование информации методом гаммирования

1.10. Поточные шифры

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 генераторов ПСЧ, реализованного на основе *N*-разрядного регистра сдвига с линейной обратной связью (Linear Feedback Shift Register (LFSR)), называются *скремблерами*, а сам процесс преобразования – скремблированием (рис. 1.13 и 1.14).

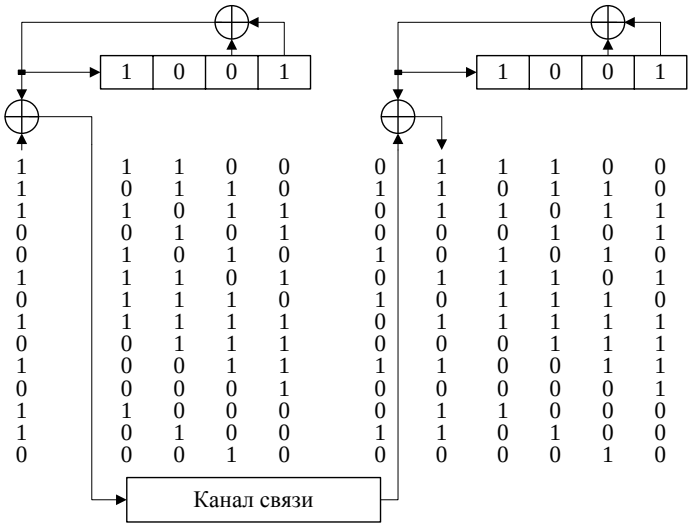


Рис. 1.13. Синхронное поточное шифрование с использованием LFSR

В синхронных поточных шифрах (см. рис. 1.13) 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.

В самосинхронизирующихся поточных шифрах элементы входной последовательности зашифровываются с учетом N предшествующих элементов (рис. 1.14), которые принимают участие в формировании ключевой последовательности. В самосинхронизирующихся шифрах имеет место эффект размножения ошибок, в то же время, в отличие от синхронных, восстановление синхронизации происходит автоматически через N элементов зашифрованной последовательности.

В последующих главах будут рассмотрены более эффективные схемы генераторов ПСЧ и наиболее известные современные поточные шифры.

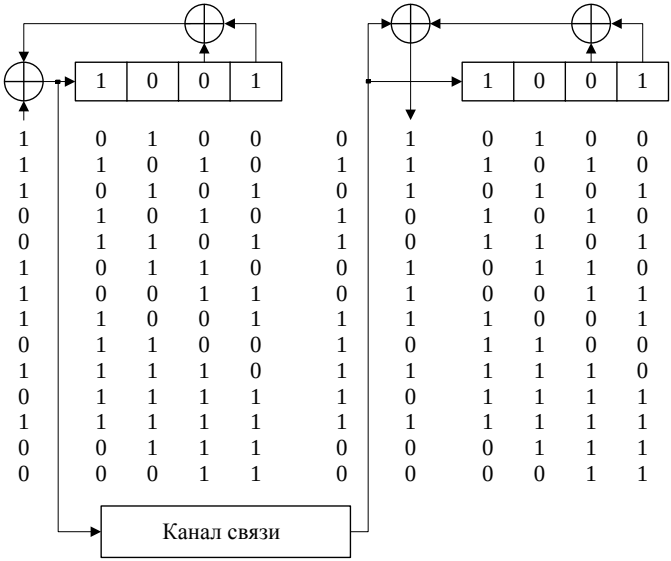


Рис. 1.14. Самосинхронизирующееся поточное шифрование с использованием LFSR

Контрольные вопросы

1. Какие свойства шифрования методом гаммирования вы знаете?
2. Перечислите методы оценки стойкости шифров.
3. Назовите свойства синхронного поточного шифрования вам известны.
4. Какие свойства самосинхронизирующегося поточного шифрования вам известны?
5. Перечислите свойства блочного составного шифра.
6. Какие требования предъявляются к абсолютно стойкому шифру?

7. В чем состоит принцип построения блочных составных шифров К. Шеннона?
8. Перечислите свойства петли Фейстеля.
9. Сформулируйте правило Кирхгофа.

ГЛАВА 2. КРИПТОСИСТЕМЫ С СЕКРЕТНЫМ КЛЮЧОМ

2.1. Модель симметричной криптосистемы

В системе, показанной на рис. 2.1, в информационных отношениях принимают участие три действующих лица: отправитель (абонент *A*) и получатель информации (абонент *B*), а также противник (*W*). Современные одноключевые криптосистемы предполагают использование взаимно-обратных преобразований *E* (encryption) и *D* (decryption) блоков данных фиксированной длины. Для задания блочной криптосистемы необходимо определить:

числовые параметры криптоалгоритма – разрядность *n* шифруемых блоков данных, объем ключевой информации, размер раундового ключа, число раундов шифрования;

раундовую функцию *F* шифрования;

алгоритм получения раундовых ключей из исходного ключа k_{AB}.

Задача абонента *A* заключается в том, чтобы передать получателю конфиденциальное сообщение *m*, состоящее из *t* блоков длины *n*, т. е.

$$m = m_1 m_2 \ldots m_i \ldots m_t,\ i = \overline{1,\ t}.$$

Задача абонента *B* заключается в том, чтобы, получив переданное сообщение

$$c = c_1 c_2 \ldots c_i \ldots c_t,$$

понять его содержание. Для того чтобы только получатель мог прочитать посланное сообщение, отправитель преобразует открытый текст *m* с помощью функции зашифрования *E* и секретного (известного только *A* и *B*) ключа k_{AB} в шифротекст *c*:

$$c = E_{AB}(m),$$

который и поступает в канал связи. Получатель восстанавливает исходный текст сообщения с помощью функции расшифрования *D* и того же секретного ключа k_{AB}:

$$m = D_{AB}(c).$$

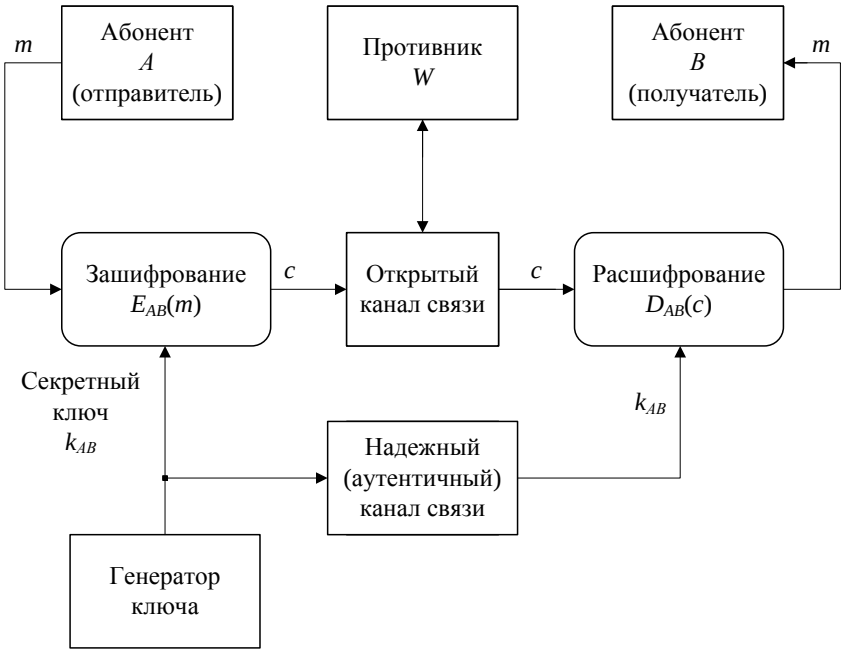


Рис. 2.1. Модель криптосистемы с секретным ключом

Для реализации такого информационного обмена должен существовать надежный канал, по которому происходит предварительный обмен секретными ключами, а у одного из его законных участников должен быть генератор, формирующий качественные ключи k_{AB}, обеспечивающие гарантированную стойкость системы.

Цель противника – воспрепятствовать осуществлению намерений законных участников информационного обмена. В общем случае противник может перехватывать зашифрованные сообщения, модифицировать их и даже посылать фальсифицированные сообщения стороне *B* якобы от имени другой, в рассматриваемом случае – от стороны *A*.

2.2. Классификация угроз противника. Основные свойства криптосистемы

Таким образом, имеют место три возможных типа угроз со стороны противника:

1) *нарушение секретности информации* – дешифрование (полное или частичное) переданного сообщения или получение информации о его сути;
2) *нарушение целостности информации* – внесение в сообщение искажений, которые законный получатель не смог бы обнаружить;
3) *нарушение подлинности информации* – формирование ложных сообщений, которые законный получатель *B* принял бы за подлинные, пришедшие от *A*.

Дешифрование переданного сообщения противником возможно в случае вычисления им секретного ключа, либо нахождения алгоритма, функционально эквивалентного D_{AB} и не требующего знания k_{AB}.

Соответственно задачей криптографа является обеспечение требуемого уровня крипто- и имитостойкости системы. *Криптостойкость* – это защищенность криптосистемы от несанкционированного ознакомления с содержимым зашифрованных сообщений. *Имитостойкость* – защищенность криптографической системы от навязывания ложных данных [8].

2.3. Классификация атак на криптосистему с секретным ключом

В симметричной криптосистеме различают пять уровней атак со стороны криптоаналитика [4]:

1) *атака на основе только шифротекста* (ciphertext-only attack): противнику известны *n* шифротекстов, зашифрованных на одном и том же ключе *k*;
2) *атака на основе известного* (*невыбранного*) *открытого текста* (known-plaintext attack): противнику известны *n* шифротекстов, зашифрованных на одном и том же ключе *k*, а также соответствующие им открытые тексты;

3) *атака на основе выбранного открытого текста* (chosen-plaintext attack): противник может выбрать необходимое число открытых текстов и получить соответствующие им шифрограммы (при этом в случае *простой* атаки такого типа все открытые тексты могут быть выбраны до получения первой шифрограммы; в случае *адаптивной* атаки противник выбирает очередной открытый текст, зная шифрограммы всех предыдущих);
4) *атака на основе выбранного шифротекста* (chosen-ciphertext attack): противник может выбрать необходимое количество шифротекстов и получить соответствующие им открытые тексты (в случае *простой* атаки такого типа все шифрограммы должны быть выбраны до получения первого открытого текста; в случае адаптивной атаки противник выбирает очередную шифрограмму, зная открытые тексты всех предыдущих);
5) *атака на основе выбранного текста* (chosen-text attack): противник может атаковать криптосистему с двух сторон, т. е. выбирать шифрограммы и дешифровать их, а также выбирать открытые тексты и шифровать их (атака такого типа может быть простой, адаптивной, простой с одной стороны и адаптивной с другой).

В каждом случае противник должен либо определить ключ k_{AB}, либо выполнить дешифрование некоторого нового шифротекста, зашифрованного на том же ключе, что и сообщения, предоставленные ему для исследования на начальной стадии криптоанализа.

Атаки перечислены в порядке возрастания их силы. Различие в степени действенности, например атак первых трех уровней, можно показать на примере шифра простой (одноалфавитной) замены. При анализе на основе только шифротекста для раскрытия этого простейшего шифра требуется провести некоторую работу, аналогичную той, которую провели герои вышеупомянутых произведений Э. По и А. Конан Дойла. В случае атаки на основе известного открытого текста раскрытие шифра становится тривиальным в особенности после того, как в открытых текстах встретятся большинство символов используемого алфавита. При атаке на основе выбранного открытого текста в случае ис-

пользования английского алфавита шифр будет вскрыт сразу после получения шифровки $E_k(ABC...XYZ)$ [3].

2.4. Режимы использования блочных шифров

Для различных ситуаций, встречающихся на практике, разработано значительное количество режимов шифрования [2, 12, 21, 28, 30].

Наиболее очевидное решение задачи закрытия сообщений, состоящих из нескольких блоков, заключается в независимом шифровании каждого блока на одном и том же ключе k_{AB}. Данная классическая схема блочного шифрования (рис. 2.2) известна под названием режима *электронной кодовой книги* (Electronic Code Book (ECB)). В ГОСТ 28147-89 [12] данный режим назван *режимом простой замены*.

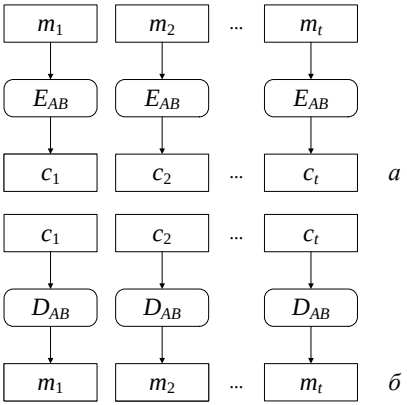


Рис. 2.2. Шифрование в режиме ECB: *а* – зашифрование, *б* – расшифрование

Уравнения зашифрования и расшифрования в режиме ECB имеют вид

$$c_i = E_{AB}(m_i),$$
$$m_i = D_{AB}(c_i),$$
$$i = \overline{1,\ t}.$$

Режим имеет три существенных недостатка. Так как блоки шифруются независимо друг от друга, при зашифровании двух

или более одинаковых блоков получаются одинаковые блоки шифротекста и наоборот. Данное свойство режима ECB позволяет противнику делать выводы о тождественности тех блоков открытого текста, которым соответствуют одинаковые блоки шифротекста. В тех случаях, когда длина исходного сообщения не кратна n, где n – разрядность блока данных шифра, возникает проблема дополнения последнего блока до нужного размера. Дополнение последнего неполного блока некоей фиксированной комбинацией битов в некоторых случаях может позволить противнику методом перебора определить этот неполный блок. И, наконец, данный режим нечувствителен к выпадению или вставке целого числа блоков шифротекста.

Отмеченные недостатки ограничивают область использования режима ECB только шифрованием ключевой информации, объем которой обычно кратен n, при этом качественные ключи не могут содержать повторяющихся блоков. Применение режима ECB оправдано также в базах данных, когда требуется произвольный доступ для чтения (записи) к различным полям.

Все остальные режимы реализуют комбинированные схемы шифрования (см. рис. 1.2) и обеспечивают зависимость каждого блока шифротекста не только от соответствующего блока открытого текста, но и от его позиции. На рис. 2.3 показана схема шифрования в режиме *сцепления блоков шифротекста* (Ciphertext Block Chaining (CBC)). Уравнения зашифрования и расшифрования в режиме CBC имеют вид

$$c_i = E_{AB}\left(m_i \oplus c_{i\text{-}1}\right),$$
$$m_i = D_{AB}\left(c_i\right) \oplus c_{i\text{-}1},$$
$$i = \overline{1,\ t},$$

где секретность n-разрядного блока c_0 (синхропосылки) не является обязательной.

Отличительные особенности режима CBC – зависимость при зашифровании i-го блока шифротекста от всех предшествующих блоков открытого текста и зависимость при расшифровании каждого блока открытого текста m_i только от двух блоков ($c_{i\text{-}1}$ и c_i) шифротекста. Первое свойство делает пригодным использование режима для решения задач контроля целостности информации.

Второе свойство делает режим самосинхронизирующимся: одиночная ошибка (ошибка при передаче одного блока) может привести к неправильному расшифрованию только двух блоков. Как будет видно в дальнейшем, этот режим используется и в криптосистемах с открытым ключом в том случае, если размер шифруемого сообщения больше, чем размер блока.

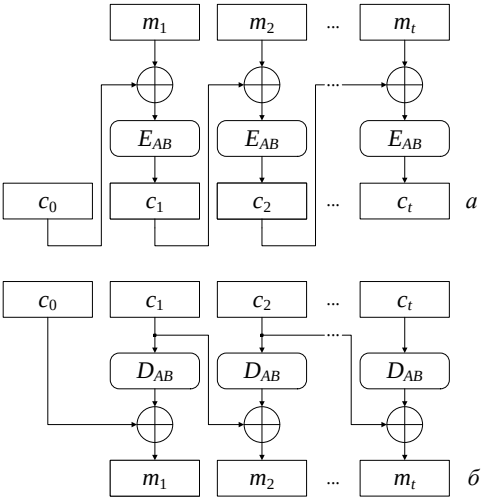


Рис. 2.3. Режим шифрования CBC: *а* – зашифрование, *б* – расшифрование

Схема режима *обратная связь по шифротексту* (Ciphertext Feedback (CFB)) показана на рис. 2.4. Схема, приведенная ранее на рис. 1.14, может рассматриваться как простейший частный случай данной схемы. Уравнения зашифрования и расшифрования имеют вид:

$$c_i = m_i \oplus E_{AB}^{(\tau)}\left(s_{i\text{-}1}\right),$$
$$m_i = c_i \oplus E_{AB}^{(\tau)}\left(s_{i\text{-}1}\right),$$
$$s_i = \left(2^{\tau} s_{i\text{-}1} + c_i\right) \bmod 2^n,$$
$$i = \overline{1,\ t},$$

где n – разрядность регистра сдвига; τ – разрядность шифруемых блоков данных ($1 \le \tau \le n$); s_0 – начальное состояние

регистра сдвига (синхропосылка); $E_{AB}^{(\tau)}(s_{i-1})$ – старшие τ битов n-разрядной шифрограммы $E_{AB}(s_{i-1})$.

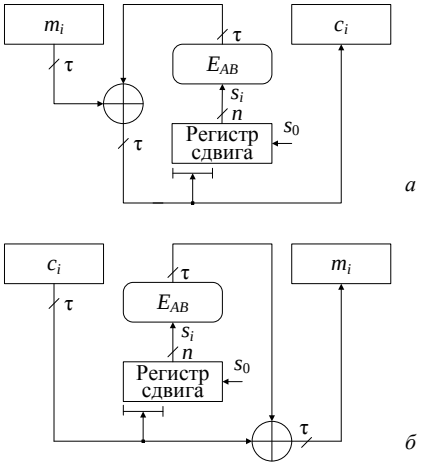


Рис. 2.4. Режим шифрования CFB: *а* – зашифрование, *б* – расшифрование

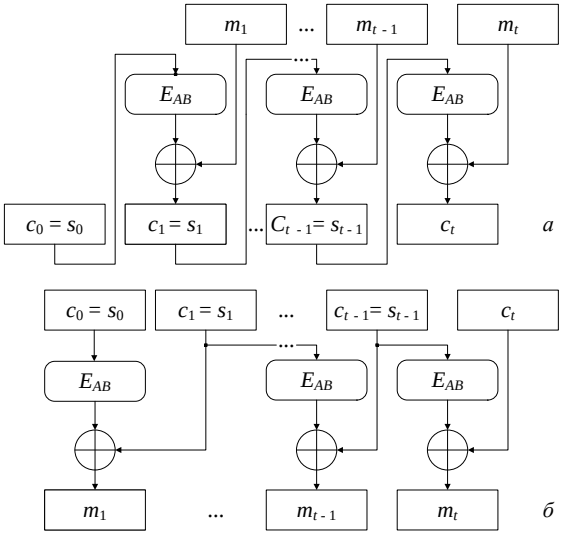


Рис. 2.5. Режим шифрования CFB при $\tau = n$:
а – зашифрование, *б* – расшифрование

Схема шифрования в режиме CFB при $\tau = n$ показана на рис. 2.5. Уравнения зашифрования и расшифрования в этом случае принимают вид

$$c_i = m_i \oplus E_{AB}\left(c_{i-1}\right),$$
$$m_i = c_i \oplus E_{AB}\left(c_{i-1}\right),$$
$$i = \overline{1, t}.$$

В ГОСТ 28147-89 аналогичный режим назван *режимом гаммирования с обратной связью*. Свойства данной схемы шифрования аналогичны режиму CBC: при зашифровании каждый блок шифротекста зависит от всего предшествующего ему открытого текста, при расшифровании отсутствует эффект «размножения» ошибок.

Схема шифрования в режиме *обратной связи по выходу* (Output Feedback (OFB)) показана на рис. 2.6. Схема, приведенная ранее на рис. 1.13, может рассматриваться как простейший частный случай данной схемы. Гамма шифра снимается с выходов генератора псевдослучайных чисел, реализованного на основе n-разрядного регистра сдвига, в цепи обратной связи которого используется функция зашифрования E_{AB}. Уравнения зашифрования и расшифрования имеют вид

$$c_i = m_i \oplus \gamma_i,$$
$$m_i = c_i \oplus \gamma_i,$$
$$\gamma_i = E_{AB}^{(\tau)}\left(s_{i-1}\right),$$
$$s_i = \left(2^{\tau} s_{i-1} + \gamma_i\right) \bmod 2^n,$$
$$i = \overline{1, t},$$

где γ_i – очередной элемент гаммирующей последовательности; n – разрядность регистра сдвига; τ – разрядность шифруемых блоков данных; s_0 – начальное состояние регистра сдвига (синхропосылка); $E_{AB}^{(\tau)}\left(s_{i-1}\right)$ – старшие τ битов n-разрядной шифрограммы $E_{AB}\left(s_{i-1}\right)$. Последовательность $\gamma = \gamma_1\gamma_2 \ldots \gamma_i \ldots \gamma_t$ не зависит от открытого текста и поэтому всякий раз при фиксированных k_{AB} и s_0 будет вырабатываться одна и та же гамма. Данный факт требует при шифровании на одном ключе двух различных массивов данных использовать различные синхропосылки.

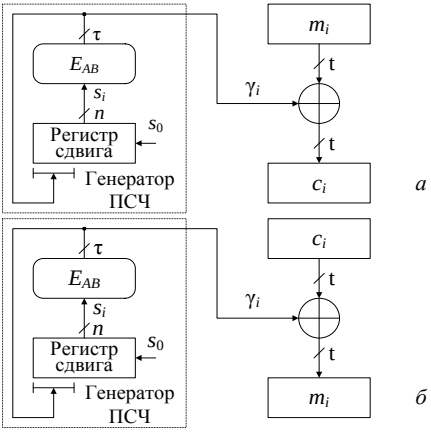


Рис. 2.6. Режим шифрования OFB: *а* – зашифрование, *б* – расшифрование

Схема шифрования в режиме OFB при $\tau = n$ показана на рис. 2.7. Уравнения зашифрования и расшифрования в этом случае принимают вид:

$$c_i = m_i \oplus E_{AB}\left(s_{i-1}\right),$$
$$m_i = c_i \oplus E_{AB}\left(s_{i-1}\right),$$
$$s_i = E_{AB}\left(s_{i-1}\right),$$
$$i = \overline{1,\ t}.$$

В схеме режима *счетчика* (Counter) (рис. 2.8) генератор ПСЧ имеет двухступенчатую структуру. Первая ступень это либо *n*-разрядный счетчик, изменение состояния которого задается формулой $s_i = s_{i-1} + 1$, либо генератор *n*-разрядных кодов с максимально возможным периодом выходной последовательности. Каждый *n*-разрядный двоичный набор с выхода счетчика или генератора кодов первой ступени поступает на вход функции зашифрования E_{AB}, результат действия которой – очередной элемент гаммы: $\gamma_i = E_{AB}^{(\tau)}\left(s_{i-1}\right)$. Так же, как и в режиме OFB, для обратимости процедур шифрования при зашифровании и расшифровании должна использоваться одна и та же синхропосылка s_0. Уравнения зашифрования и расшифрования в режиме счетчика имеют вид

$$c_i = m_i \oplus \gamma_i,$$
$$m_i = c_i \oplus \gamma_i,$$
$$i = \overline{1,\ t},\ 1 \le \tau \le n.$$

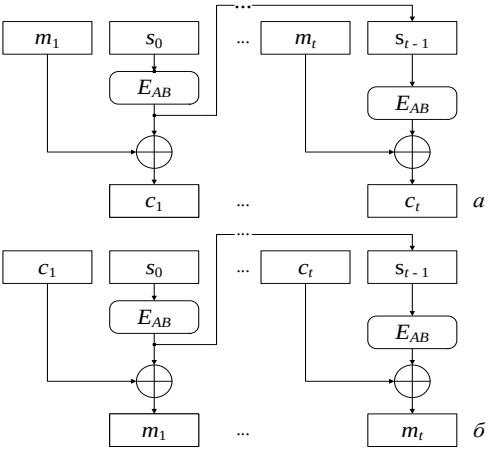


Рис. 2.7. Режим шифрования OFB при $\tau = n$: *а* – зашифрование, *б* – расшифрование

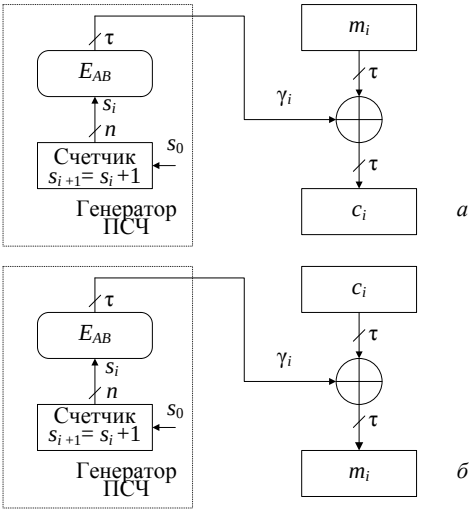


Рис. 2.8. Режим шифрования Counter: *а* – зашифрование, *б* – расшифрование

В ГОСТ 28147-89 аналогичный режим называется режимом *гаммирования*.

Режимы счетчика и OFB с точки зрения обеспечения высокого уровня криптозащиты наиболее эффективны: именно они, по своей сути, наиболее близки к схеме одноразового использования, т.е. к абсолютно стойкому шифру. С одинаковым на то основанием оба они могут называться режимом гаммирования и в качестве последнего имеют следующие свойства:

1) при зашифровании и расшифровании используется одна и та же функция E_{AB};
2) любой элемент (бит, байт, блок и т.п.) информационной последовательности шифруется независимо от других;
3) изменение любого бита шифротекста на противоположное значение приводит после расшифрования к аналогичному изменению соответствующего бита открытого текста:
$$\overline{b_1^{(c)}} = b_1^{(c)} \oplus 1 = \left(b_i^{(m)} \oplus \gamma_i\right) \oplus 1 = \left(b_i^{(m)} \oplus 1\right) \oplus \gamma_i = \overline{b_i^{(m)}} \oplus \gamma_i,$$
где $b_i^{(c)}$, $b_i^{(m)}$ – соответственно биты закрытого и открытого текста;
4) повторное наложение той же гаммы γ на зашифрованную последовательность c, дает на выходе исходную последовательность
$$m \oplus \gamma \oplus \gamma = m;$$
5) шифрование нулевой последовательности дает на выходе гамму
$$0 \oplus \gamma = \gamma;$$
6) если известны две последовательности $c^{(1)}$ и $c^{(2)}$, зашифрованные с использованием одной и той же гаммы, ее действие нейтрализуется следующим образом:
$$c^{(1)} \oplus c^{(2)} = m^{(1)} \oplus \gamma \oplus m^{(2)} \oplus \gamma = m^{(1)} \oplus m^{(2)},$$
после чего для дешифрования может быть использован частотный анализ;
7) если известны две последовательности $c^{(1)}$ и $c^{(2)}$, зашифрованные с использованием одних и тех же значений k

и s_1, т.е. с использованием одной и той же гаммы, и известна последовательность $m^{(1)}$, то последовательность $m^{(2)}$ может быть легко вычислена по формуле

$$m^{(2)} = c^{(1)} \oplus c^{(2)} \oplus m^{(1)},$$

так как справедливо соотношение

$$c^{(1)} \oplus c^{(2)} \oplus m^{(1)} = \gamma \oplus c^{(2)} = m^{(2)}.$$

Третье свойство режимов гаммирования дает возможность противнику, не обладающему секретным ключом, воздействуя лишь на биты шифротекста, вносить предсказуемые, а в некоторых случаях даже целенаправленные изменения в получаемый после расшифрования открытый текст. Однако, как справедливо отмечается в [8], это свойство гаммирования нельзя считать недостатком, так как задачей любого шифра является лишь обеспечение секретности, задачи обеспечения целостности и подлинности информации решаются с использованием других механизмов, которые будут рассмотрены в последующих главах. Возможность внесения предсказуемых изменений при использовании режимов гаммирования исчезает при использовании режима CFB (гаммирования с обратной связью).

Режим счетчика следует признать более качественным, чем режим OFB, учитывая, что во втором случае криптостойкая функция E_{AB}, используемая в цепи обратной связи генератора ПСЧ, вовсе не гарантирует максимально возможный период гаммы. Кроме того, если в режиме счетчика при реализации алгоритма E_{AB}, не используемого в цепи обратной связи генератора ПСЧ, возникает случайное искажение информации, то при этом (в отличие от режима OFB, где искажаются все последующие элементы гаммы) искажается только один элемент гаммы, а значит, неправильно расшифровывается только один соответствующий блок.

Приведенные пять режимов использования блочных шифров являются наиболее распространенными. Более того, первые четыре в свое время были официально утверждены Национальным бюро стандартов США в качестве режимов использования алгоритма DES.

Генератор ПСЧ, функционирующий в соответствие с уравнением

$$s_i = \left(2^{\tau} s_{i-1} + E_{AB}^{(\tau)}\left(s_{i-1}\right)\right) \bmod 2^n,$$

официально утвержден для использования в режиме Output Feedback и его модификациях. Стойкость генератора при $\tau < n$ можно повысить, если в каждом такте полностью менять содержимое регистра генератора ПСЧ, т.е. в качестве сигналов обратной связи использовать n-разрядный код $E_{AB}\left(s_{i-1}\right)$. Уравнение работы модифицированного генератора (рис. 2.9) имеет вид

$$s_i = E_{AB}\left(s_{i-1}\right).$$

2.5. Российский стандарт криптографической защиты

В данном разделе в качестве примера блочного итерационного шифра приведено краткое описание алгоритма криптографического преобразования данных ГОСТ 28147-89 [5–8], в дальнейшем изложении – просто ГОСТ.

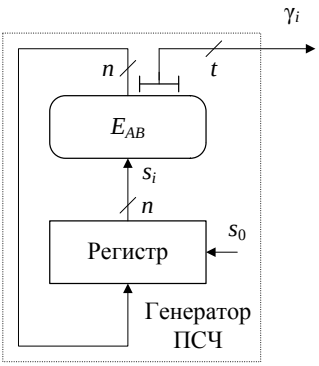


Рис. 2.9. Модифицированный генератор ПСЧ для режимов шифрования, подобных OFB

Достоинства ГОСТа:

удобство программной реализации на 32-разрядных процессорах;

регулярная структура устройства (реализующего алгоритм), облегчающая его интегральное исполнение;

криптостойкость, приемлемая для подавляющего числа приложений;

оригинальный качественный генератор ПСЧ (генератор гаммы);

большой запас «прочности» за счет значительного объема ключевой информации.

ГОСТ является классическим итерационным блочным шифром Фейстеля с разрядностью блоков данных, равным 64 битам. Чтобы разобраться в ГОСТе, необходимо понять:

характер используемой при шифровании ключевой информации,

структуру раундовой функции шифрования (так называемого *основного шага криптопреобразования*),

алгоритмы зашифрования E_k (базовые циклы 32-З и 16-З) и расшифрования D_k (базовый цикл 32-Р),

рекомендуемые режимы использования.

Ключевая информация ГОСТа. Ключевая информация представляет собой два массива данных: собственно ключ k и таблицу замен H (рис. 2.10). Ключ – это массив из восьми 32-разрядных элементов $k = \{k_m\}$, $m = \overline{0, 7}$. Таким образом, размер ключа составляет $8 \times 32 = 256$ битов или 32 байта. Таблица замен – набор из восьми одномерных массивов $H = \{H_m\}$, $m = \overline{0, 7}$, так называемых *узлов замены*, каждый из которых определяет логику работы четырехразрядного блока подстановок (*S*-блока) и по этой причине содержит шестнадцать 4-разрядных двоичных наборов (от 0 до 15), расположенных в произвольном порядке. Элемент m-го узла замен $S_m(j) = H_{j,\,m}$ – 4-разрядный код на выходе соответствующего m-го *S*-блока при поступлении на его вход двоичного кода числа j. Таким образом, объем таблицы замен равен $8 \times 16 \times 4 = 512$ битов или 64 байта. Ключ должен быть массивом статистически независимых битов, принимающих с равной вероятностью значения 0 и 1. Если для генерации ключевой информации используется генератор ПСЧ, то он должен обладать криптостойкостью, не меньшей, чем у самого ГОСТа. Таблица замен H (в отличие от ключа k) – долговре-

менный ключевой элемент. Она, например, может быть общей для всех процедур шифрования в рамках одной системы криптографической защиты.

Таблица замен H						
Адрес ячейки	Узел замен H_7	...	Узел замен H_m	...	Узел замен H_1	Узел замен H_0
0	$S_7(0)$		$S_m(0)$		$S_1(0)$	$S_0(0)$
1	$S_7(1)$		$S_m(1)$		$S_1(1)$	$S_0(1)$
⋮						
j	$S_7(j)$		$S_m(j)$		$S_1(j)$	$S_0(j)$
⋮						
15	$S_7(15)$		$S_m(15)$		$S_1(15)$	$S_0(15)$

Рис. 2.10. Таблица замен ГОСТ 28147-89

Раундовая функция шифрования ГОСТа. Структура основного шага криптопреобразования показана на рис. 2.11, где $d^{(t)}$ – входной блок, $d^{(t+1)}$ – выходной блок, x – шаговый ключ. В качестве исходных данных шаг получает 64-разрядный блок данных $d = (L, R)$ и 32-разрядный раундовый ключ x, в качестве которого используется один из элементов ключа k_m. В ходе выполнения шага левая (L) и правая (R) половины блока данных рассматриваются как отдельные 32-разрядные элементы данных, в качестве которых они подвергаются следующим преобразованиям:

1) сложению по модулю 2^{32} полублока R и элемента ключа x;
2) разбиению результата s на восемь четырехбитовых блоков, поблочной замене по таблице замен, формированию из получившихся блоков нового значения s;
3) циклическому сдвигу результата s на 11 разрядов влево;
4) поразрядному сложению по модулю два (XOR) результата s и полублока L;

5) элемент R становится новым значением элемента L, значение результата предыдущей операции становится новым значением элемента R.

Полученные значения элементов L и R выдаются в качестве результата шага.

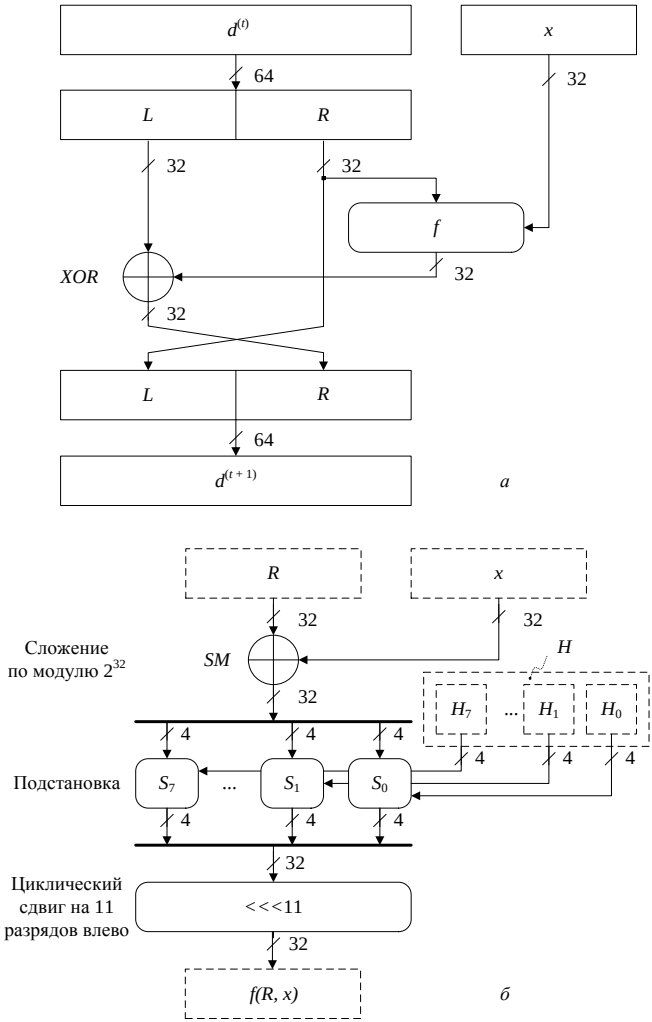


Рис. 2.11. ГОСТ 28147-89:
а – схема основного шага криптопреобразования, *б* – шаговая функция *f*

Базовые циклы ГОСТа. Базовые циклы 32-З, 32-Р (циклы шифрования E_k и D_k) и 16-З (цикл выработки имитоприставки E_k) отличаются друг от друга только числом повторений основного шага и порядком использования элементов ключа. В обозначении цикла первый элемент – число повторений основного шага (32 или 16), второй элемент – тип функции (З – зашифрование, Р – расшифрование). Кроме того, в конце циклов шифрования для взаимной обратимости предусмотрена дополнительная перестановка элементов L и R. Порядок использования элементов ключа имеет вид:

цикл зашифрования 32-З – три раза вперед, один раз назад
(k_0, k_1, ..., k_7, k_0, k_1, ... , k_7, k_0, k_1, ..., k_7, k_7, k_6, ..., k_0);
цикл расшифрования 32-Р – один раз вперед, три раза назад
(k_0, k_1, ..., k_7, k_7, k_6, ..., k_0, k_7, k_6, ..., k_0, k_7, k_6, ..., k_0);
цикл выработки имитоприставки 16-З – два раза вперед
(k_0, k_1, ..., k_7, k_0, k_1, ..., k_7).

Режимы шифрования ГОСТа. ГОСТ 28147-89 определяет три режима шифрования данных (простая замена, гаммирование и гаммирование с обратной связью) и режим выработки имитоприставки.

Простая замена заключается в применении цикла 32-З к блокам открытого текста при зашифровании и цикла 32-Р к блокам закрытого текста при расшифровании. Режим простой замены полностью совпадает с режимом ECB, описанным выше.

Гаммирование – наложение на открытые данные псевдослучайной гаммирующей последовательности. Режим гаммирования аналогичен режиму *Counter*, описанному выше. Блоки гаммы получаются в результате зашифрования в режиме простой замены элементов выходной последовательности 64-разрядного счетчика. Таким образом, генератор гаммы имеет ярко выраженную двухступенчатую структуру, при этом первая ступень, т.е. счетчик, обеспечивает близкий к максимально возможному значению 2^{64} период гаммирующей последовательности, а вторая (т.е. функция E_k) – необходимую криптостойкость. Схема счетчика ГОСТа, являющегося по сути двумя независимо работающими рекуррентными генераторами со взаимно простыми зна-

чениями периодов (2^{32} и $2^{32}-1$), формирующими правую и левую половины блоков, обладает следующими свойствами:

соседние 64-битовые значения, вырабатываемые генератором, для повышения криптостойкости отличаются друг от друга в каждом байте;

генератор легко реализуется как аппаратно, так и на 32-разрядных процессорах программно;

синхропосылка (начальное заполнение генератора ПСЧ) перед инициализацией генератора подвергается преобразованию по циклу 32-З;

период выходной последовательности равен $2^{32}\left(2^{32}-1\right)$.

Рекуррентные соотношения для старшей (Q_L) и младшей (Q_R) частей генератора имеют вид

$$Q_L(t+1)=\left(Q_L(t)+C_1\right) \bmod 2^{32};$$

$$Q_H(t+1)=\left(Q_H(t)+C_2-1\right) \bmod\left(2^{32}-1\right)+1,$$

где C_1 = 01010101h, C_2 = 01010104h.

Гаммирование с обратной связью полностью совпадает с вышеописанным режимом CFB. Его отличительная особенность – способ выработки очередного элемента гаммы. Первый элемент гаммы является результатом шифрования по циклу 32-З синхропосылки. Любой последующий элемент гаммы вырабатывается как результат преобразования по циклу 32-З предыдущего блока шифротекста, вследствие чего каждый блок шифротекста зависит от соответствующего и всех предыдущих блоков открытого текста.

Рассмотрим еще раз влияние искажений шифротекста на полученные в результате расшифрования открытые данные.

В режиме простой замены искажение блока шифротекста приводит при расшифровании к непредсказуемым изменениям соответствующего блока открытого текста.

В режиме гаммирования все изменения открытого текста предсказуемы.

В режиме гаммирования с обратной связью искажение блока шифротекста после расшифрования приводит к изменениям двух блоков открытого текста, при этом один из них оказывается искаженным предсказуемым образом, а другой – непредсказуемым.

Непредсказуемые изменения в расшифрованном массиве данных могут быть обнаружены лишь в случае избыточности этих данных. Такая избыточность, к сожалению, имеет место практически только для текстов на естественных или искусственных языках. Поэтому для обнаружения случайных или умышленных искажений информационной последовательности требуется специальный режим, в ГОСТ он назван *режимом выработки имитоприставки*.

Имитоприставка – это добавляемый к зашифрованным данным контрольный код, зависящий от открытых данных и ключевой информации. Для противника вычислительно неразрешимы две задачи:

1) вычисление имитоприставки для заданной незашифрованной информационной последовательности;
2) подбор открытых данных под заданную имитоприставку.

Схема формирования имитоприставки – зашифрование информации в режиме CFB, при этом E_k – функция зашифрования по циклу 16-3. В качестве имитоприставки используется часть последнего зашифрованного блока, обычно его младшие 32 разряда. В общем случае вероятность успешного навязывания ложных данных равна 2^{-N}.

2.6. Американский стандарт криптографической защиты

История конкурса на стандарт криптографической защиты XXI в. – Advanced Encryption Standard (AES). В 1997 г. НИСТ объявил о начале программы по принятию нового стандарта криптографической защиты, стандарта XXI в., для закрытия важной информации правительственного уровня, на замену существующему с 1974 г. алгоритму DES, на тот момент самому распространенному криптоалгоритму в мире. DES устарел по многим параметрам: длине ключа, удобству реализации на современных процессорах, быстродействию и другим, за исключением самого главного – стойкости. За 25 лет интенсивного криптоанализа не было найдено методов вскрытия этого шифра, существенно отличающихся по эффективности от полного перебора по ключевому пространству.

Требования к кандидатам были следующие:

криптоалгоритм должен быть открыто опубликован;

криптоалгоритм должен быть симметричным блочным шифром, допускающим размеры ключей в 128, 192 и 256 бит;

криптоалгоритм должен быть предназначен как для аппаратной, так и программной реализации;

криптоалгоритм не должен быть запатентован, в противном случае патентные права должны быть аннулированы;

криптоалгоритм подвергается исследованиям по таким параметрам, как *стойкость, стоимость, гибкость*.

Стойкость. Это самый важный критерий в оценке алгоритма. Оценивались: способность шифра противостоять различным методам криптоанализа, статистическая безопасность и относительная защищенность по сравнению с другими кандидатами. Оценивалась стойкость к атаке методом полного перебора с учетом прогнозируемого роста вычислительных мощностей.

Стоимость. Не менее важный критерий, если принимать во внимание одну из основных целей NIST – широкую область использования и доступность AES. Стоимость зависит от вычислительной эффективности (в первую очередь быстродействия) на различных платформах, удобства программной и аппаратной реализации, низким требованиям к памяти, простоты (простые алгоритмы легче реализовывать, они более прозрачны для анализа)

Гибкость. Включает в себя способность алгоритма обрабатывать ключи больше оговоренного минимума (128 бит), надежность и эффективность выполнения в разных средах, возможность реализации других криптографических функций.

Другими словами, AES должен быть существенно более эффективным с точки зрения практической реализации (в первую очередь скорости шифрования и формирования ключей), иметь больший запас прочности, чем TripleDES, при этом не уступая ему в стойкости.

Реализуемость в Smart-картах. Важная область использования AES – smart-карты, при этом главной проблемой является небольшой объем доступной памяти. NIST исходил из допущения, что некоторые дешевые карты могут иметь всего 256 байтов

RAM (для вычисляемых данных) и 2000 байтов ROM (для хранения алгоритмов и констант). Существуют два основных метода формирования раундовых ключей:

1) вычисление на начальном этапе работы криптоалгоритма и хранение в памяти;
2) вычисление раундовых ключей «на лету».

Ясно, что второй вариант уменьшает затраты RAM, и поэтому наличие такой возможности в криптоалгоритме является его несомненным достоинством.

На конкурс были приняты 15 алгоритмов, разработанных криптографами 12 стран – Австралии, Бельгии, Великобритании, Германии, Израиля, Канады, Коста-Рики, Норвегии, США, Франции, Южной Кореи и Японии.

В финал конкурса вышли следующие алгоритмы: MARS, RC6, TWOFISH (США), RIJNDAEL (Бельгия), SERPENT (Великобритания, Израиль, Норвегия). По своей структуре TWOFISH – классический шифр Фейстеля; MARS и RC6 можно отнести к модифицированным шифрам Фейстеля, в них используется новая малоизученная операция циклического «прокручивания» бит слова на число позиций, изменяющихся в зависимости от шифруемых данных и секретного ключа; RIJNDAEL и SERPENT являются классическими SP-сетями. MARS и TWOFISH имеют самую сложную конструкцию, RIJNDAEL и RC6 – самую простую.

Финалисты будут описаны по единой схеме, данной в документе NIST. Сначала описываются обнаруженные «слабости» алгоритма (если таковые имеются), затем преимущества и, наконец, недостатки.

MARS выставлен на конкурс фирмой IBM, одним из авторов шифра является Д. Копперсмит, участник разработки DES. В алгоритме не обнаружено слабостей в защите.

Преимущества:

высокий уровень защищенности;

высокая эффективность на 32-разрядных платформах, особенно поддерживающих операции умножения и циклического сдвига;

потенциально поддерживает размер ключа больше 256 бит.

Недостатки:

сложность алгоритма, затрудняющая анализ его надежности;

снижение эффективности на платформах без необходимых операций;

сложность защиты от временного анализа и анализа мощности.

RC6 предложен фирмой RSA Lab, один из авторов – Р. Ривест. В алгоритме не обнаружено слабостей в защите.

Преимущества:

высокая эффективность на 32-разрядных платформах, особенно поддерживающих операции умножения и циклических сдвигов;

простая структура алгоритма, упрощающая анализ его надежности;

наличие хорошо изученного предшественника – RC5;

быстрая процедура формирования ключа;

потенциально поддерживает размер ключа больше 256 бит;

длина ключа и число раундов могут быть переменными.

Недостатки:

относительно низкий уровень защищенности;

снижение эффективности на платформах, не имеющих необходимых операций;

сложность защиты от временного анализа и анализа мощности;

невозможность генерации раундовых ключей «на лету».

TWOFISH основан на широко используемом шифре BLOWFISH; один из авторов разработки – Б. Шнайер. Главная особенность шифра – изменяющиеся в зависимости от секретного ключа таблицы замен. В алгоритме не обнаружено слабостей в защите.

Преимущества:

высокий уровень защищенности;

хорошо подходит для реализации в Smart-картах из-за низких требований к памяти;

высокая эффективность на любых платформах, в том числе на ожидаемых в будущем 64-разрядных архитектурах фирм *Intel* и *Motorola*;

поддерживает вычисление раудовых ключей «на лету»;

поддерживает распараллеливание на уровне инструкций;

допускает произвольную длину ключа до 256 бит.

Недостатки:

особенности алгоритма затрудняют его анализ;

высокая сложность алгоритма;

применение операции сложения делает алгоритм уязвимым к анализу мощности и временному анализу.

RIJNDAEL, большинством участников конкурса названный лучшим выбором (если будет отвергнут их собственный шифр), основан на шифре SQUARE тех же авторов. В алгоритме не обнаружено слабостей в защите.

Преимущества:

высокая эффективность на любых платформах;

высокий уровень защищенности;

хорошо подходит для реализации в Smart-картах из-за низких требований к памяти;

быстрая процедура формирования ключа;

хорошая поддержка параллелизма на уровне инструкций;

поддержка разных длин ключа с шагом 32 бита.

Недостаток:

уязвимость к анализу мощности.

SERPENT – разработка профессиональных криптоаналитиков Р. Андерсона, Э. Бихэма, Л. Кнудсена. Создав шифр, успешно противостоящий всем известным на сегодня атакам, разработчики затем удвоили количество его раундов. В алгоритме не обнаружено слабостей в защите.

Преимущества:

высокий уровень защищенности;

хорошо подходит для реализации в Smart-картах из-за низких требований к памяти.

Недостатки:

самый медленный алгоритм среди финалистов;

уязвимость к анализу мощности.

В октябре 2000 г. конкурс завершился – победителем был признан бельгийский шифр RIJNDAEL, имеющий наилучшее сочетание стойкости, производительности, эффективности реализации, гибкости. Его низкие требования к объему памяти делают его идеально подходящим для встроенных систем. Авторы

шифра – Joan Daemen и Vincent Rijmen, начальные буквы фамилий которых и образуют название алгоритма – **RIJ**N**DAE**L.

AES-128. RIJNDAEL – это итеративный блочный шифр, имеющий переменную длину блоков и различные длины ключей. Длина ключа и длина блока могут быть равны независимо друг от друга 128, 192 или 256 битам.

Рассмотрим версию криптоалгоритма AES-128. Промежуточные результаты преобразований, выполняемых в рамках криптоалгоритма, называются *состояниями* (state). Все входные блоки данных, все промежуточные результаты преобразований, все выходные блоки данных, а также ключ шифрования можно представить в виде квадратного массива байтов (рис. 2.12). Этот массив имеет четыре строки и четыре столбца.

a_{00}	a_{01}	a_{02}	a_{03}
a_{10}	a_{11}	a_{12}	a_{13}
a_{20}	a_{21}	a_{22}	a_{23}
a_{30}	a_{31}	a_{32}	a_{33}

k_{00}	k_{01}	k_{02}	k_{03}
k_{10}	k_{11}	k_{12}	k_{13}
k_{20}	k_{21}	k_{22}	k_{23}
a_{30}	a_{31}	a_{32}	a_{33}

Рис. 2.12. Пример представления состояния и ключа шифрования для AES-128

В некоторых случаях ключ шифрования рассматривается как линейный массив 4-байтовых слов. Слова состоят из четырех байтов, которые находятся в одном столбце (при представлении в виде прямоугольного массива).

Входные данные для шифра обозначаются как байты состояния в порядке a_{00}, a_{10}, a_{20}, a_{30}, a_{01}, a_{11}, a_{21}, a_{31}, ... После завершения действия шифра выходные данные получаются из байтов состояния в том же порядке.

В состав раунда AES-128 входят следующие четыре преобразования (рис. 2.13):

побайтовая замена байтов состояния с использованием фиксированной таблицы замен размером 8×256 (SubBytes);

побайтовый циклический сдвиг строк результата – i-я строка сдвигается на i байтов влево, $i = \overline{0,3}$ (ShiftRows);

перемешивание столбцов результата (MixColumns);

поразрядное сложение по модулю 2 (XOR) результата с раундовым ключом (AddRoundKey).

Десятый раунд отличается от остальных – в нем отсутствует предпоследняя операция для более эффективной реализации обратного преобразования.

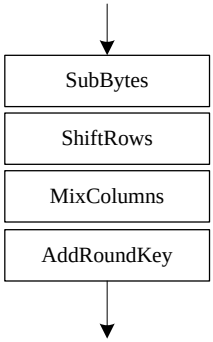


Рис. 2.13. Раундовые преобразования AES-128

Замена байтов (SubBytes). Преобразование представляет собой нелинейную замену байтов, выполняемую независимо с каждым байтом состояния. Таблица замены *S*-блока является инвертируемой. Рис. 2.14 иллюстрирует применение преобразования SubBytes к состоянию.

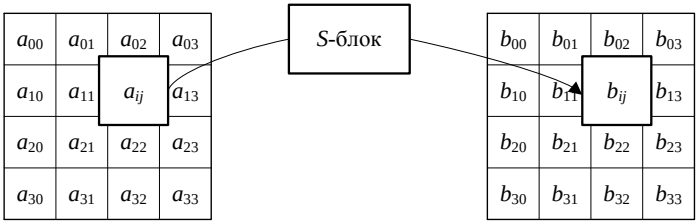


Рис. 2.14. SubBytes действует на каждый байт состояния

Преобразование сдвига строк (ShiftRows). Последние три строки состояния циклически сдвигаются на различное число байт. Строка 1 сдвигается на один байт, строка 2 – на два байта

и строка 3 – на три байта. Рис. 2.15 показывает влияние преобразования на состояние.

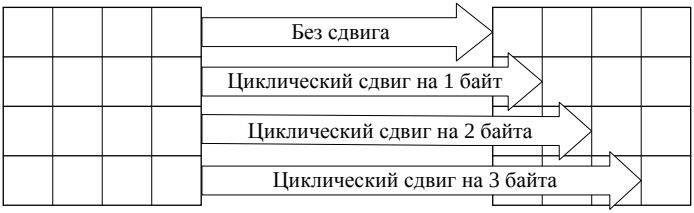


Рис. 2.15. ShiftRows действует на строки состояния

Преобразование перемешивания столбцов (MixColumns). Рис. 2.16 демонстрирует применение преобразования MixColumns к столбцу состояния.

Добавление раундового ключа (AddRoundKey). В данной операции раундовый ключ добавляется к состоянию посредством простого поразрядного XOR (рис. 2.17). Раундовый ключ вырабатывается из ключа шифрования посредством *алгоритма выработки ключей* (key schedule).

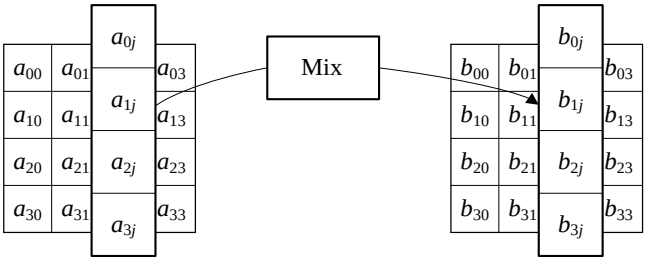


Рис. 2.16. MixColumns действует на столбцы состояния

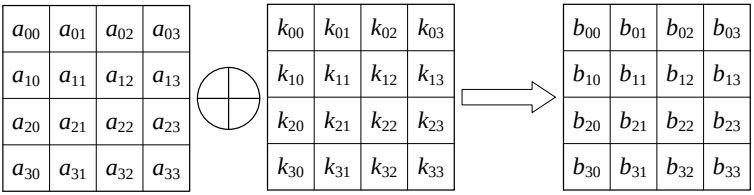


Рис. 2.17. При добавлении ключа раундовый ключ складывается посредством операции XOR с состоянием

Алгоритм выработки ключей (Key Schedule). Раундовые ключи получаются из ключа шифрования посредством алгоритма выработки ключей. Он содержит два компонента: *расширение ключа* (Key Expansion) и *выбор раундового ключа* (Round Key Selection). Основополагающие принципы алгоритма выглядят следующим образом:

общее число бит раундовых ключей равно длине блока, умноженной на число раундов плюс 1 (например, для длины блока 128 бит и 10 циклов требуется 1408 бит циклового ключа);

ключ шифрования расширяется в *расширенный ключ* (Expanded Key);

раундовые ключи берутся из расширенного ключа следующим образом: первый раундовый ключ содержит первые четыре слова, второй – следующие четыре слова и т.д.

Расширение ключа (Key Expansion). Расширенный ключ представляет собой линейный массив 4-байтовых слов. Первые четыре слова содержат ключ шифрования. Все остальные слова определяются рекурсивно из слов с меньшими индексами.

Криптоалгоритм. Шифр AES-128 состоит из:

начального добавления раундового ключа;

девяти раундов;

заключительного раунда.

Новая архитектура (рис. 2.18,*а*), с использованием которой построена функция E_k, получила название «Квадрат». Свойства алгоритма иллюстрирует рис. 2.18,*б*, из которого видно, что два раунда обеспечивают полное рассеивание и перемешивание информации. Видно, что даже незначительные изменения на входе нелинейной функции приводят к существенным изменениям на выходе (в среднем изменяется половина бит преобразованного блока).

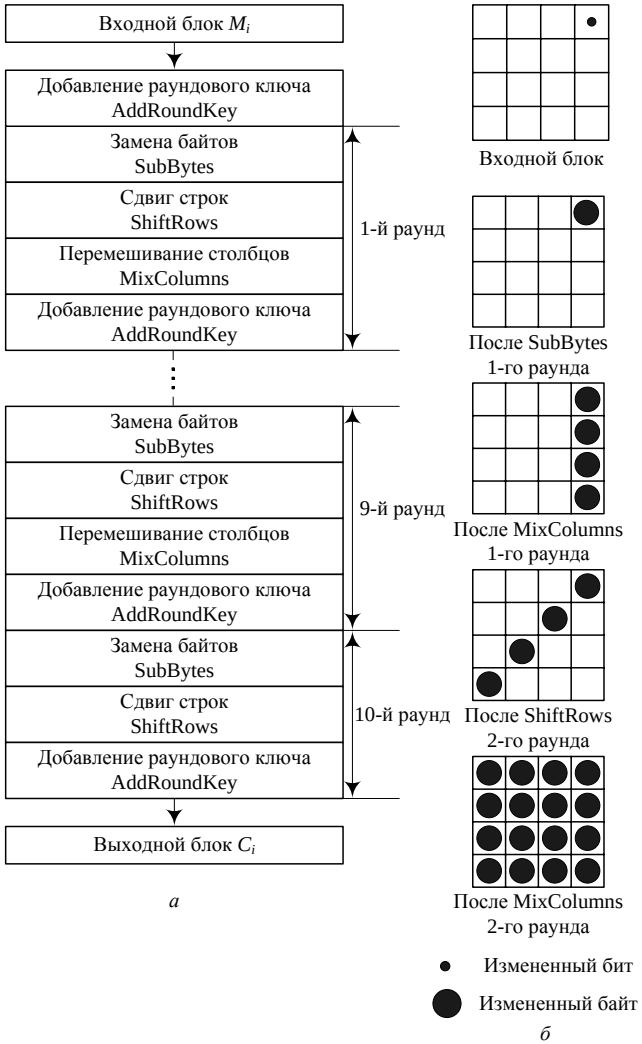


Рис. 2.18. Шифр AES-128: *а* – итеративное блочное преобразование (нелинейная функция зашифрования); *б* – рассеивание и перемешивание информации в процессе двухраундового преобразования

Обоснование числа раундов выглядит следующим образом. Авторы криптоалгоритма продемонстрировали эффективнейшую атаку на 4-раундовую версию RIJNDAEL [14]. После добавления

5-го раунда атака по-прежнему возможна, но становится сложнее. После добавления 6-го раунда атака по-прежнему возможна, но ее сложность уже сопоставима с атакой полного перебора. К получившейся 6-раундовой функции шифрования, обладающей минимальной стойкостью к рассматриваемой атаке, добавляются два начальных раунда, обеспечивающих полное рассеивание и перемешивание, и два конечных раунда, обладающих тем же свойством. Таким образом, суммарное число раундов становится равным 10.

Основные достоинства AES-128:

новая архитектура «Квадрат», обеспечивающая быстрое рассеивание и перемешивание информации, при этом за один раунд преобразованию подвергается весь входной блок;

байт-ориентированная структура, удобная для реализации на 8-разрядных микроконтроллерах;

все раундовые преобразования суть операции в конечных полях, допускающие эффективную аппаратную и программную реализацию на различных платформах.

Возможные направления совершенствования архитектуры «Квадрат». Можно выделить следующие пути совершенствования архитектуры «Квадрат»:

замена раундовой операции сдвига строк ShiftRows на операцию перемешивания строк MixRows позволит обеспечить полное рассеивание и перемешивание за один раунд;

совместное использование архитектур «Петля Фейстеля» и «Квадрат», например построение раундового преобразования f на основе операций добавления раундового ключа, замены байтов, перемешивания строк и перемешивания столбцов;

переход от архитектуры «Квадрат» к архитектуре «Куб» (приложение 3).

2.7. Вероятностное блочное шифрование

Одной из функций генераторов ПСЧ в системах криптографической защиты информации может быть внесение неопределенности в работу средств защиты, например, выбор элементов вероятностного пространства R при *вероятностном шифровании*

$$E_k : M \times R \to C,$$

где E_k, k, M, C – функция зашифрования, секретный ключ, пространство открытых текстов и пространство шифротекстов соответственно. Главная особенность вероятностного шифрования – один и тот же исходный текст, зашифрованный на одном и том же ключе, может привести к появлению огромного числа различных шифротекстов.

Во избежание путаницы следует отметить: термин вероятностное шифрование (Probabilistic Encryption) был впервые введен Ш. Гольдвассер и С. Микали. Ими же предложена первая схема такого шифрования, основанная на использовании BBS-генератора ПСЧ в качестве источника ключевой последовательности. В настоящем разделе термин «вероятностное шифрование» используется, чтобы подчеркнуть факт внесения неопределенности в работу криптоалгоритма. Представляется, что такое применение термина – более обосновано.

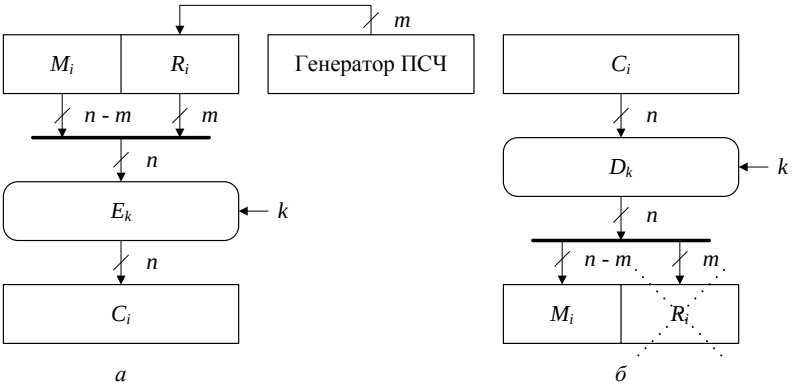
Схема одного из возможных вариантов вероятностного блочного шифрования в режиме ECB показана на рис. 2.19, где на вход функции зашифрования E_k поступает «расширенный» n-разрядный блок M'_i, полученный в результате конкатенации m-разрядного блока открытого текста M_i и двоичного набора R_i разрядности $(n - m)$ с выхода генератора ПСЧ. В результате зашифрования получается блок c_i закрытого текста, разрядность которого больше разрядности блока M_i. В результате при зашифровании одинаковых блоков на одном и том ключе получаются различные блоки шифротекста. При расшифровании часть R_i блока, полученного на выходе функции D_k, просто отбрасывается.

Можно выделить следующие достоинства вероятностного шифра:

появляется принципиальная возможность увеличения времени жизни сеансовых ключей;

использование качественного генератора ПСЧ позволяет при использовании симметричных блочных шифров уменьшить число раундов шифрования, а значит, увеличить быстродействие криптоалгоритма;

отношение длин блока открытого текста M'_i и соответствующего ему элемента r_i вероятностного пространства может выступать в качестве параметра безопасности.



а *б*

Рис. 2.19. Пример вероятностного шифрования в режиме ECB:
а – зашифрование; *б* – расшифрование

Недостаток у рассматриваемой схемы лишь один – шифротекст всегда длиннее соответствующего ему открытого текста.

Следует отметить, что подобная схема может использоваться и в асимметричных криптосистемах (системах с открытым ключом) при этом противник лишается возможности вычислять значение функции шифрования интересующих его текстов и сравнивать их с перехваченным шифротекстом.

2.8. Синхронные поточные шифры

В синхронных поточных шифрах гаммирующая последовательность формируется независимо от потока открытого текста при зашифровании и шифротекста при расшифровании. Функционирование генератора гаммы, разворачивающего короткий случайный ключ k в длинную псевдослучайную последовательность (ПСП), в общем случае можно описать следующим образом:

$$s_{t+1} = F(s_t, k),$$
$$\gamma_t = f(s_t, k),$$

где s_t – состояние элементов памяти генератора гаммы в момент времени t; F – функция обратной связи (перехода); f – функция выхода. Начальное заполнение s_0 может быть функцией от ключа и синхропосылки.

Существуют два основных режима функционирования синхронных поточных шифров. В режиме обратной связи по выходу OFB функция выхода f генератора ПСЧ не зависит от ключа и уравнения работы генератора гаммы имеют вид

$$s_{t+1} = F(s_t, k),$$
$$\gamma_t = f(s_t).$$

В режиме счетчика функция обратной связи F генератора ПСЧ не зависит от ключа, но гарантированно обеспечивает прохождение генератора через все пространство состояний (или большую его часть). Уравнения генератора гаммы в этой ситуации имеют вид

$$s_{t+1} = F(s_t),$$
$$\gamma_t = f(s_t, k).$$

Главное свойство синхронного поточного шифра – отсутствие эффекта размножения ошибок. При расшифровании неправильного бита искажается только соответствующий бит открытого текста. Данное свойство ограничивает возможность обнаружения ошибки при расшифровании. Кроме того, противник имеет возможность производить управляемые изменения шифротекста, совершенно точно зная, какие изменения в результате произойдут в соответствующем открытом тексте. При использовании синхронных поточных шифров необходимо согласование работы устройств на передающей и принимающей стороне, так как в случае потери или вставки элемента зашифрованной последовательности на принимающей стороне будет потеряна вся информация, начиная с того места, где сбилась синхронизация. Обычно синхронизация достигается либо вставкой специальных маркеров, либо переинициализацией передающего и принимающего

устройства при некотором заранее согласованном условии [21, 24, 30].

Описание наиболее известных поточных шифров приведено в [24].

Шифр A5 – поточный шифр, используемый в системах GSM (Group Special Mobile) для закрытия связи между абонентом и базовой станцией. Он является европейским стандартом для цифровых сотовых мобильных телефонов. A5 использует три LFSR длиной 19, 22 и 23 с прореженными многочленами обратной связи, т.е. с многочленами, имеющими небольшое число ненулевых коэффициентов. Выходом генератора гаммы является выход элемента сложения по модулю два M2, на входы которого поступают последовательности с выходов трех LFSR (их начальное заполнение – секретный сеансовый ключ). Используется управление синхронизацией LFSR (рис. 2.20).

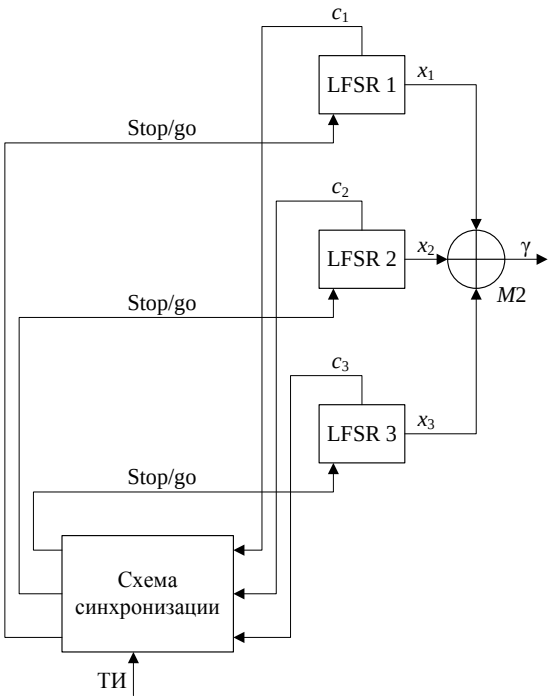


Рис. 2.20. Генератор ПСЧ поточного шифра A5

Для управления синхронизацией используются биты c_1, c_2, c_3 с выходов LFSR. В каждом такте сдвигаются как минимум два LFSR. Если $c_1 = c_2 = c_3$, сдвигаются все три регистра, в противном случае – те два регистра *i* и *j*, для которых выполняется равенство $c_i = c_j$.

Криптоанализ алгоритма показал, что для определения начального заполнения LFSR при известных 64 бит гаммы требуется перебор 2^{40} вариантов. Кроме того, около 40 % ключей приводят к циклу, длина которого наименьшая из всех возможных и равна $\left(2^{23}-1\right)\cdot 4/3$ бит. Алгоритм разрабатывался с нарушением правила Кирхгофа, поэтому имеет большое число слабостей, подробное описание которых приведено в [24].

Шифр RC4 – поточный шифр переменным размером ключа, разработанный Р. Ривестом. Алгоритм работает в режиме OFB, т.е. поток ключевой информации не зависит от открытого текста. Используется 8-разрядный *S*-блок, таблица замен имеет размерность 8×256 и является перестановкой (зависящей от ключа) двоичных чисел от 0 до 255. Применяются два счетчика Q_1 и Q_2 с нулевым начальным состоянием (рис. 2.21).

Рассмотрим процедуру генерации очередного байта гаммы. Пусть S_i и γ – содержимое ячейки с адресом *i* таблицы замен *S*-блока и очередной байт гаммы.

Алгоритм RC4

1. Такт работы первого счетчика:
$$Q_1 = \left(Q_1 + 1\right) \bmod 2^8.$$
2. Такт работы второго счетчика:
$$Q_2 = \left(Q_2 + S_{Q_1}\right) \bmod 2^8.$$
3. Ячейки таблицы замен *S*-блока с адресами Q_1 и Q_2 обмениваются своим содержимым:
$$S_{Q_1} \leftrightarrow S_{Q_2}.$$

4. Вычисление суммы содержимого ячеек таблицы замен *S*-блока с адресами Q_1 и Q_2:

$$T = \left(S_{Q_1} + S_{Q_2}\right) \bmod 2^8.$$

5. Считывание содержимого ячейки таблицы замен *S*-блока с адресом *T*:

$$\gamma = S_T.$$

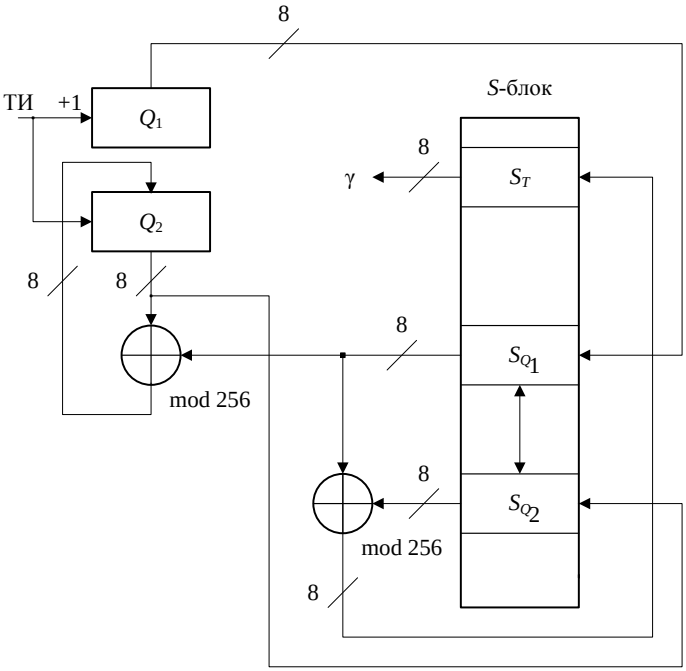


Рис. 2.21. Схема генератора ПСЧ RC4

Алгоритм инициализации таблицы замен S-блока

1. Запись в каждую ячейку таблицы замен *S*-блока ее собственного адреса:

$$\forall i = \overline{0,\ 255},\ S_i = i.$$

2. Заполнение байтами ключа другой 256-байтовой таблицы:

$$k = \{k_i\},\ i = \overline{0,\ 255}.$$

3. Инициализация индекса j: $j = 0$.
4. Перемешивание таблицы замен S-блока:

$$\forall i = \overline{0, 255},\ j = (j + S_i + k_i) \bmod 2^8,\ S_i \leftrightarrow S_j.$$

Число состояний RC4 равно приблизительно

$$2^{1700}\left(256! \times 256^2\right).$$

Таблица замен S-блока медленно изменяется при использовании, при этом счетчик Q_1 обеспечивает изменение каждого элемента таблицы, а Q_2 гарантирует, что элементы таблицы изменяются случайным образом.

Основные достоинства алгоритма:

простая и понятная структура шифра;

доступность для исчерпывающего анализа, в том числе за счет возможности проведения полного исследования версий шифра меньшей разрядности;

универсальный алгоритм перемешивания ключевой таблицы, имеющий самостоятельное значение, пригодный для создания как S-, так и R-блоков.

Наиболее известный пример неудачной реализации шифра – стандарт безопасности Wep, криптоанализ которого приведен в [24].

Шифр PIKE – это модификация взломанного шифра FISH, предложенная Р. Андерсоном. Алгоритм использует три аддитивных генератора (рис. 2.22):

$$Q_i = (Q_{i-55} + Q_{i-24}) \bmod 2^{32},$$

$$Q_i = (Q_{i-57} + Q_{i-7}) \bmod 2^{32},$$

$$Q_i = (Q_{i-58} + Q_{i-19}) \bmod 2^{32}.$$

Управление синхронизацией осуществляется на основе анализа битов переноса cro_1, cro_2, cro_3 на выходах сумматоров Sm. Если все три одинаковы $cro_1 = cro_2 = cro_3$ (все нули или все единицы), то тактируются все три генератора; в противном случае тактируются те два генератора i и j, для которых выполняется равенство $cro_i = cro_j$. Выходом генератора является результат

поразрядной операции M2 значений на выходах всех трех генераторов.

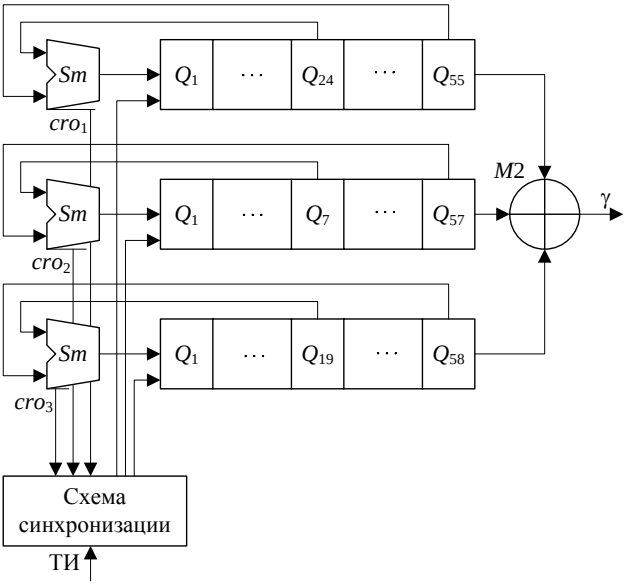


Рис. 2.22. Генератор ПСЧ PIKE

Шифр Trivium – синхронный поточный шифр, разработанный в 2005 году C. De Canniere и P. Preneel для европейского проекта eSTREAM.

Trivium, изначально проектировавшийся для аппаратной реализации, также может успешно применяться и в программном исполнении, где одним из его преимуществ является быстродействие.

Для генерации ключевого потока длиной 2^{64} бит требуется 80 бит секретного ключа и 80 бит инициализирующего значения (*IV* – initial value). Параметры шифра представлены в табл. 2.1.

Как и в большинстве поточных шифров, процесс генерации ключевого потока состоит из двух фаз: 1 – внутреннее состояние шифра инициализируется ключом и инициализирующим значением (*IV*), 2 – состояние неоднократно обновляется и генерируется ключевой поток. Сначала рассмотрим вторую.

Таблица 2.1

Параметры Trivium

Параметр	Значение, бит
Размер ключа	80
Размер IV	80
Внутреннее состояние	288

Генерация ключевого потока. Шифр состоит из 288 бит внутреннего состояния обозначаемых $(s_1, s_2, ..., s_{288})$. Генерация ключевого потока представляет собой повторяющийся процесс: из внутреннего состояния шифра извлекаются значения 15 определенных битов, с помощью которых обновляются 3 бита внутреннего состояния и рассчитывается один бит ключевого потока z_i, затем внутреннее состояние сдвигается. Процесс повторяется до тех пор, пока требуемое количество битов ($N \le 2^{64}$) ключевого потока не сгенерируется. Описание этого процесса на псевдокоде имеет следующий вид:

for $i = 1$ to N **do**

$t_1 \leftarrow s_{66} + s_{93}$

$t_2 \leftarrow s_{162} + s_{177}$

$t_3 \leftarrow s_{243} + s_{288}$

$z_i \leftarrow t_1 + t_2 + t_3$

$t_1 \leftarrow t_1 + s_{91} \cdot s_{92} + s_{171}$

$t_2 \leftarrow t_2 + s_{175} \cdot s_{176} + s_{264}$

$t_3 \leftarrow t_3 + s_{286} \cdot s_{287} + s_{69}$

$(s_1, s_2, ..., s_{93}) \leftarrow (t_3, s_1, s_2, ..., s_{92})$

$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (t_1, s_{94}, s_{95}, ..., s_{176})$

$(s_{178}, s_{179}, ..., s_{288}) \leftarrow (t_2, s_{178}, s_{179}, ..., s_{288})$

end for

Операторы «+» и «·» – операции XOR и AND соответственно.

Схема генератора ПСЧ Trivium представлена на рис. 2.23.

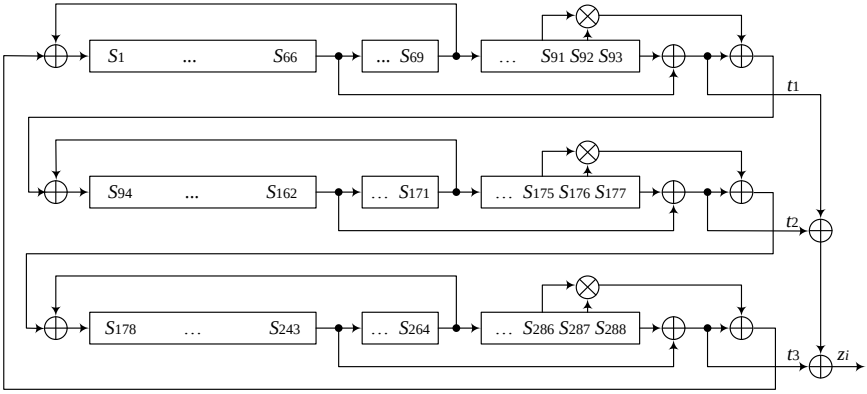


Рис. 2.23. Генератор ключевого потока Trivium

Инициализация начального состояния ключом и IV. Алгоритм инициализируется загрузкой 80-битного ключа K и инициализирующего значения IV такой же разрядности в 288 бит внутреннего состояния, все остальные биты внутреннего состояния, исключая $s_{286}, s_{287}, s_{288}$, обнуляются. Затем полученное внутреннее состояние сдвигается вправо четыре полных цикла похожим способом, как описано выше, но без генерации ключевого потока. Этот процесс на псевдокоде можно описать следующим образом:

$$(s_1, s_2, ..., s_{93}) \leftarrow (K_1, K_2, ..., K_{80}, 0, ..., 0)$$
$$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (IV_1, IV_2, ..., IV_{80}, 0, ..., 0)$$
$$(s_{178}, s_{179}, ..., s_{288}) \leftarrow (0, ..., 0, 1, 1, 1)$$

for $i = 1$ to $4 \cdot 288$ **do**

$t_1 \leftarrow s_{66} + s_{91} \cdot s_{92} + s_{93} + s_{171}$

$t_2 \leftarrow s_{162} + s_{175} \cdot s_{176} + s_{177} + s_{264}$

$t_3 \leftarrow s_{243} + s_{286} \cdot s_{287} + s_{288} + s_{69}$

$(s_1, s_2, ..., s_{93}) \leftarrow (t_3, s_1, s_2, ..., s_{92})$

$(s_{94}, s_{95}, ..., s_{177}) \leftarrow (t_1, s_{94}, s_{95}, ..., s_{176})$

$(s_{178}, s_{179}, ..., s_{288}) \leftarrow (t_2, s_{178}, s_{179}, ..., s_{288})$

end for

Несмотря на тот факт, что Trivium проектировался не для программной реализации, шифр довольно успешно может применяться на стандартных ПК.

2.9. Самосинхронизирующиеся поточные шифры

Самосинхронизирующиеся поточные шифры благодаря свойству восстанавливать информацию после нарушения синхронизации – это наиболее распространенный метод шифрования в дипломатических, военных и промышленных системах связи. Наиболее распространенный режим функционирования самосинхронизирующихся поточных шифров – режим обратной связи по шифротексту.

В шифрах рассматриваемого типа каждый зашифрованный символ (бит при последовательной передаче) зависит от ключа k и только от последних N_m символов шифротекста. Память разрядности N_m поточного шифра может быть реализована различными способами, простейший из них – использование N_m-разрядного регистра сдвига (рис. 2.24).

Пусть F_c – функция шифрования, G – функция, преобразующая на стадии инициализации секретный ключ K и синхропосылку S во внутренний ключ k и вектор инициализации IV регистра сдвига. Тогда уравнение работы криптосхемы имеет вид

$$(k, IV) = G(K, S);$$
$$(c_{-N_m+1} \ldots c_0) = IV;$$
$$z_t = F_c(k, c_{t-N_m} \ldots c_{t-1});$$
$$c_t = M_t \oplus z_t.$$

Расшифрование символа закончится успешно, если последние N_m символов шифротекста были приняты правильно. Пусть N_s – разрядность символа, тогда одиночная ошибка в канале связи вызовет серию из $N_m N_s$ потенциально неправильно расшифрованных бит на принимающей стороне. Самосинхронизирующееся шифрование имеет смысл применять лишь тогда, когда вероятность битовых ошибок в двоичном канале без памяти значительно меньше величины $(N_m N_s)^{-1}$. Дополнительное достоинство рассматриваемой криптосхемы – возможность реализации поверх любой цифровой коммуникационной системы.

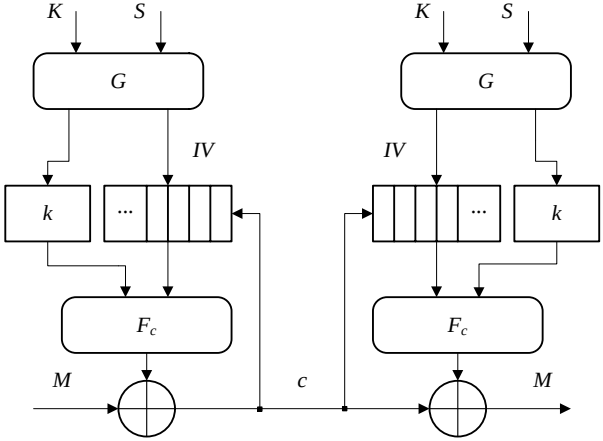


Рис. 2.24. Общая схема самосинхронизирующегося поточного шифра

Один из существующих подходов – использование для построения самосинхронизирующихся поточных шифров последовательной и параллельной композиции простейших функций f шифрования. Параллельная композиция суть побитовый XOR последовательностей с выходов двух блоков f, в последователь-

ной композиции выход одного блока f является входом другого. На рис. 2.25 показана схема поточного шифра У. Маурера.

Базовый компонент схемы – 3-битовый регистр сдвига с зависящей от ключа функцией f выхода. Четыре таких компонента образуют блок следующего уровня – пару параллельно соединенных цепочек из двух базовых компонентов. На следующем уровне четыре таких блока собраны в последовательную композицию. На двух следующих уровнях эта конструкция повторяется. Результирующая последовательностная машина (ПМ) имеет входную память объемом 192 бит и четыре бита-компонента на каждый бит памяти. Внутреннее состояние ПМ служит входом для зависящей от ключа функции шифрования F_c. Для задания конструкций булевых функций f ключ шифрования «разворачивается» в 256 байтов, каждый из которых определяет таблицу истинности соответствующей функции.

Основная идея данной конструкции – построение ПМ, количество элементов памяти которой превышает разрядность входной памяти.

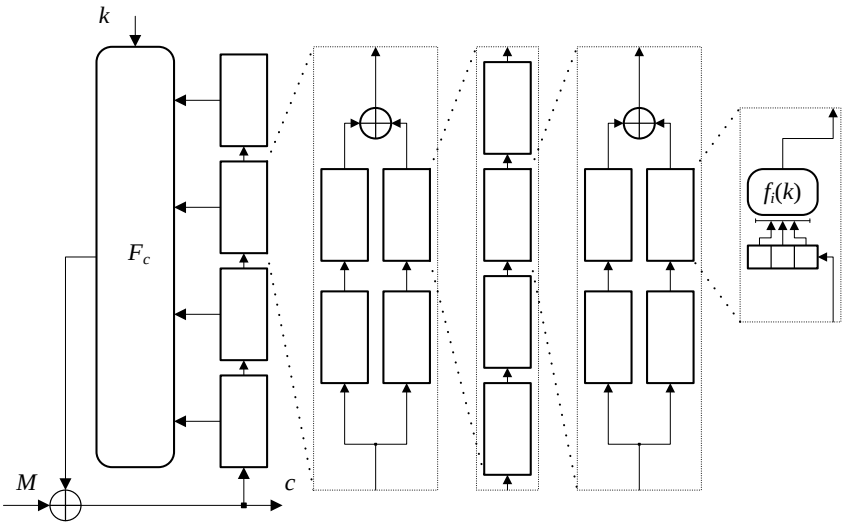


Рис. 2.25. Схема Маурера

Контрольные вопросы

1. Сформулируйте требования к качественному симметричному шифру.
2. Перечислите недостатки режима шифрования ECB.
3. Укажите области предпочтительного использования режима шифрования ECB.
4. Какие режимы шифрования могут использоваться для формирования контрольного кода целостности информации?
5. Какие режимы шифрования требуют наличия только функции зашифрования E_{AB}?
6. Сравните генераторы ПСЧ со структурами OFB и Counter.
7. Перечислите основные достоинства Российского стандарта криптозащиты.
8. Сколько раундов AES-128 обеспечивают полное рассеивание и перемешивание информации? Обоснуйте свой ответ.
9. Сравните архитектуры блочных шифров «Сеть Фейстеля» и «Квадрат».
10. Укажите основные различия блочных и поточных шифров (не менее пяти).
11. Укажите способы заполнения таблиц замен *S*-блоков (не менее трех).
12. Кто является инициатором создания симметричной криптосистемы: отправитель или получатель?
13. Укажите два принципиальных недостатка симметричной криптосистемы.
14. Нарисуйте схему вероятностного блочного шифрования: а) в режиме ECB; б) в режиме OFB, в) в режиме CFB.
15. Сформулируйте требования к качественному *S*-блоку.
16. Перечислите достоинства вероятностного шифрования.

ГЛАВА 3. ХЕШ-ФУНКЦИИ

3.1. Требование к качественной хеш-функции

Хеширование – вид криптографического преобразования, не менее важный, чем шифрование. При этом существует мнение, что задача проектирования качественной хеш-функции более сложная, чем задача проектирования качественного симметричного шифра.

Хеш-функцией называется преобразование h, превращающее информационную последовательность (строку) M произвольной длины в информационную последовательность (строку) фиксированной длины $h(M)$. На рис. 3.1 показана упрощенная схема хеш-функции.

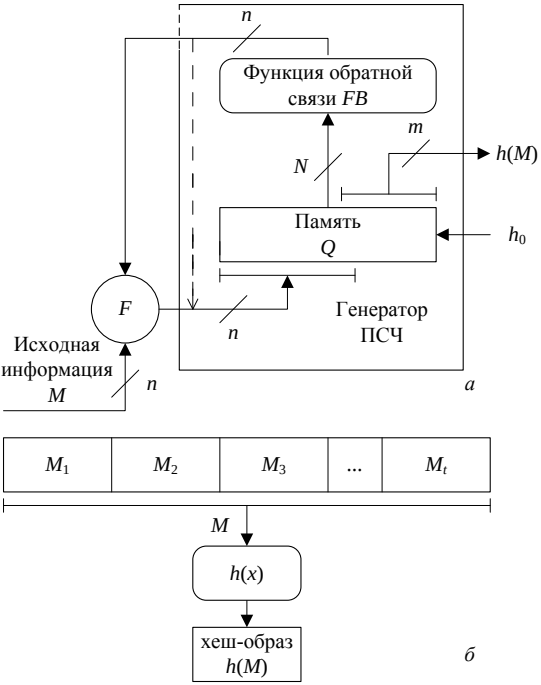


Рис. 3.1. Хеш-функция: *а* – наложение ПСП на входную информационную последовательность; *б* – упрощенный принцип действия хеш-функции

Процесс получения хеш-функции можно рассматривать как наложение псевдослучайной последовательности (ПСП) на входную

преобразуемую последовательность. По этой причине часто спецификация синхронного поточного шифра описывает и родственную хеш-функцию.

К криптографической функции $h(x)$ предъявляются следующие основные требования:

результат действия хеш-функции должен зависеть от всех двоичных символов исходного сообщения, а также от их взаимного расположения;

$h(x)$ должна быть чувствительна к любым изменениям входной информационной последовательности, при любых изменениях на входе результат действия хеш-функции должен быть непредсказуем – в среднем должна измениться половина бит хеш-образа.

Рассмотрим четыре *проблемы дней рождения*, которые могут быть применены для анализа безопасности хеш-функций.

Имеем равномерно распределенную случайную переменную, принимающую N возможных значений (между 0 и $N-1$).

1) Определить минимальное число реализаций k, при котором с вероятностью $P \geq 1/2$ хотя бы одна выборка оказалась равной предопределенной величине.

2) Определить минимальное число реализаций k, при котором с вероятностью $P \geq 1/2$ хотя бы одна выборка оказалась равной выбранной величине.

3) Определить минимальное число реализаций k, при котором с вероятностью $P \geq 1/2$ хотя бы две выборки оказались равными.

Формируем два набора случайных значений по k выборок в каждом.

4) Определить минимальное число реализаций k, при котором с вероятностью $P \geq 1/2$ хотя бы одна выборка из первого набора оказалась равной одной выборке из второго набора.

Решения проблем дней рождения приведены в табл. 3.1.

Для качественной криптографической хеш-функции три следующие задачи – вычислительно неразрешимы:

1) нахождение *прообраза* – задача нахождения последовательности M по заданному хеш-образу $h(M)$ (рис. 3.2,*а*);

2) нахождение *коллизии* – задача нахождения последовательностей M и M', причем $M' \neq M$, таких, что $h(M') = h(M)$;

3) нахождение *второго прообраза* – задача нахождения для заданной последовательности M другой последовательности M', $M' \neq M$, такой, что $h(M') = h(M)$ (рис. 3.2,*б*).

Таблица 3.1

Решения проблем дней рождения

Проблема	Вероятность	Значение k	Значение k при $P = 1/2$	Значение k при $P = 1/2$ и $N = 365$
1	$P \approx -e^{-k/N}$	$k \approx \ln[1/(1-P)] \times N$	$k \approx 0{,}69 \times N$	253
2	$P \approx -e^{-(k-1)/N}$	$k \approx n[1/(1-P)] \times N + 1$	$k \approx 0{,}69 \times N + 1$	254
3	$P \approx -e^{k(k-1)/2N}$	$k \approx \{2\ln[1/(1-P)]\}^{1/2} \times N^{1/2}$	$k \approx 1{,}18 \times N^{1/2}$	23
4	$P \approx -e^{-k^2/2N}$	$k \approx \{\ln[1/(1-P)]\}^{1/2} \times N^{1/2}$	$k \approx 0{,}83 \times N^{1/2}$	16

Примечание. Последняя ячейка в строке 3 представляет собой классический *парадокс дней рождения*.

Если n – разрядность хеш-образа, сложность первой и третьей атаки (рис. 3.2) на идеальную хеш-функцию пропорциональна 2^n. Сложность задачи нахождении коллизии пропорциональна $2^{n/2}$. В табл. 3.2 приведены оценки сложности атак на идеальную хеш-функцию, где k – размер списков сообщений, создаваемых атакующим. Рассмотрим отличия поиска коллизии в случаях 1 и 2. В первом случае речь идет о поиске двух произвольных сообщений, имеющих одинаковое значение хеш-образа. Есть точка зрения, что эта атака бесполезна для атакующего. Однако существуют атаки на конкретные протоколы с использованием известных коллизий MD5. Во втором случае предполагается, что одно сообщение реальное, а другое – фальсифицированное, при этом оба сообщения должны быть осмысленными. Решение состоит в создании двух списков осмысленных сообщений путем внесения избыточности или модификации содержимого (например, добавление пробелов,

перестановка слов сообщения, добавление дополнительных избыточных слов и т.п.) без изменения смысла сообщений.

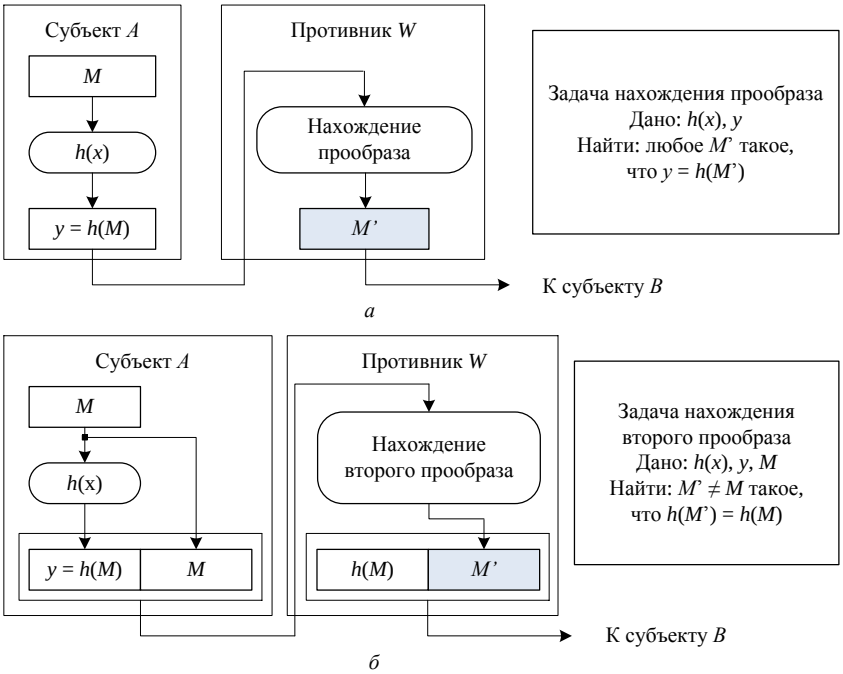


Рис. 3.2. Атаки на хеш-функцию:
а – нахождение прообраза; *б* – нахождение второго прообраза

Наиболее известные алгоритмы хеширования – MD5, SHA, Tiger, Whirlpool.

MD5 – представитель семейства хеш-функций MD (Message Digest Algorithm), предложенного Р. Ривестом; разработан в 1991 г.; преобразует информационную последовательность произвольной длины в хеш-образ разрядностью 128 бит.

Tiger разработан Р. Андерсоном и Э. Бихэмом; предназначен для реализации на 64-разрядных компьютерах; преобразует информационную последовательность произвольной длины в хеш-образ разрядностью 192 бит.

Таблица 3.2

Оценка сложности атак на хеш-функцию

Атака	Значение k при $P=1/2$	Сложность
Нахождение прообраза	$k \approx 0{,}69\times 2^n$	2^n
Нахождение второго прообраза	$k \approx 69\times 2^n + 1$	2^n
Нахождение коллизии 1	$k \approx 1{,}18\times 2^{n/2}$	$2^{n/2}$
Нахождение коллизии 2	$k \approx 0{,}83\times 2^{n/2}$	$2^{n/2}$

3.2. Итеративная хеш-функция

На рис. 3.3 показана схема итеративной хеш-функции, которая является стойкой в смысле нахождения коллизий, если аналогичным свойством обладает используемая функция сжатия. Обозначения: $M_1, M_2, ..., M_t$ – блоки дополненного сообщения; t – число блоков дополненного сообщение; h_0 – фиксированное значение, называемое стартовым вектором или вектором инициализации *IV*; $h_1, h_2, ..., h_t$ – промежуточные результаты вычисления итераций, число которых равно числу блоков t. Оригинальное сообщение дополняется до длины, кратной n, где n – разрядность блока данных, обрабатываемого функцией сжатия. На i-м итерационном шаге функция сжатия f принимает результат предыдущего шага $h_{i\text{-}1}$ и i-й блок данных M_i, а затем формирует результат $h_i = f\left(h_{i\text{-}1}, M_i\right)$. На шаге t полученное значение h_t объявляется хеш-образом исходного сообщения, т.е. $h_t = h(M)$. Типичная структура дополнения показана на рис. 3.4.

3.3. Secure Hash Algorithm (SHA)

Алгоритм SHA является частью стандарта SHS (Secure Hash Standard), разработанного в 1993 г. Национальным институтом стандартов и технологий США (FIPS 180) и Агенством национальной безопасности США. SHA использует принципы, предложенные ранее Р. Ривестом при разработке своих алгоритмов семейства MD. В 1995 г. стандарт был пересмотрен (FIPS 180-1) в пользу версии

SHA-1. Позднее стандарт пересмотрен вновь, и были определены четыре новые версии: SHA-224, SHA-256, SHA-384, SHA-512. Все версии имеют одинаковую структуру, поэтому часто их называют общим именем SHA-2. В табл. 3.3 приведены характеристики этих версий.

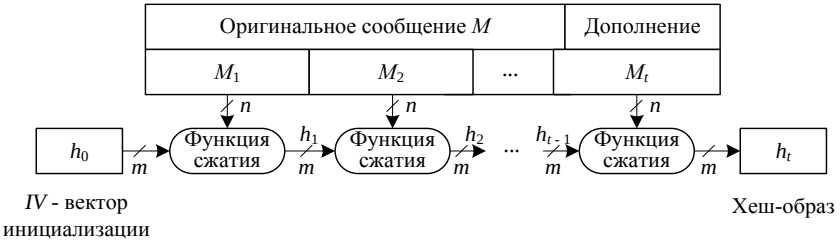


Рис. 3.3. Схема итеративной хеш-функции

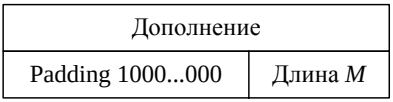


Рис. 3.4. Структура дополнения при итеративном хешировании

Таблица 3.3

Характеристики SHA

Характеристика	SHA-1	SHA-224	SHA-256	SHA-384	SHA-512
Максимальная длина сообщения	$2^{64}-1$	$2^{64}-1$	$2^{64}-1$	$2^{128}-1$	$2^{128}-1$
Длина блока	512	512	512	1024	1024
Длина хеш-образа	160	224	256	384	512
Число раундов	80	64	64	80	80
Длина слова	32	32	32	64	64

Рассмотрим версию SHA-1, в которой осуществляется преобразование информационной последовательности произвольной длины в хеш-образ разрядностью 160 бит. На первом этапе информационная последовательность дополняется до длины, кратной 512 бит. Сначала информационная последовательность дополняется до длины на 64 двоичных разряда меньшей числа, кратного 512: к концу последовательности (сообщения) приписывается 1, а затем необходимое количество нулей. После этого

приписывается 64-разрядный код длины сообщения. Если длина исходного сообщения больше 2^{64}, используются только 64 младших разряда этого кода. Пусть после дополнения получена информационная последовательность

$$M = M_1 M_2 \ldots M_i \ldots M_t,\ i = \overline{1,\ t},\ |M_i| = 512.$$

На вход i-го основного цикла SHA_i преобразования поступает i-й блок информационной последовательности и результат работы предыдущего цикла SHA_{i-1}, т.е.

$$SHA_i = h(M_i, SHA_{i-1}).$$

Перед началом каждого цикла соответствующий блок расширяется до 80 слов по 32 разряда в каждом. Расширение происходит следующим образом. Пусть

$w_1 w_2 \ldots w_{16}$ – исходный блок,

$\tilde{w}_1 \tilde{w}_2 \ldots \tilde{w}_{80}$ – расширенный блок,

при этом

$$\tilde{w}_j = w_j \text{ для } j = \overline{1,16},$$

$$\tilde{w}_j = Rol\left(w_{j-3} \oplus w_{j-8} \oplus w_{j-14} \oplus w_{j-16}\right)$$

для $j = \overline{17,80}$,

где Rol – операция циклического сдвига на один разряд влево.

Перед началом первого цикла инициализируются пять 32-разрядных переменных:

A = 67452301h, B = EFCDAB89h, C = 98BADCFEh,

D = 10325476h, E = C3D2E1F0h,

при этом стартовый вектор хеширования (синхропосылка) есть результат конкатенации этих переменных, т.е.

$$SHA_0 = (A, B, C, D, E).$$

Конкатенация новых значений этих переменных, полученных в конце i-го цикла, объявляется результатом работы цикла SHA_i.

В начале каждого цикла создаются копии входных переменных

$$AA = A,\ BB = B,\ CC = C,\ DD = D,\ EE = E.$$

Затем выполняются 80 шагов алгоритма, на каждом из которых происходит выполнение следующих операций:

$$Temp = Rol^5 A + f_j(B, C, D) + E + M_{ij} + c_j;$$

$$E = D;$$

$$D = C;$$

$$C = Rol^{30} B;$$

$$A = Temp,$$

где Rol^n – операция циклического сдвига на n разрядов влево; f_j – шаговая функция; M_{ij} – j-е слово i-го блока M_i; c_j – шаговая константа; $j = \overline{1, 80}$, $i = \overline{1, t}$.

В первом раунде (при $j = \overline{1, 20}$) используются функция

$$f_j(X, Y, Z) = XY \vee \overline{X}Z$$

и константа

$$c_j = 5A827999h;$$

во втором раунде (при $j = \overline{21, 40}$) – функция

$$f_j(X, Y, Z) = X \oplus Y \oplus Z$$

и константа

$$c_j = 6ED9EBA1h;$$

в третьем раунде (при $j = \overline{41, 60}$) – функция

$$f_j(X, Y, Z) = XZ \oplus XY \oplus ZY$$

и константа

$$c_j = 8F1BBCDCh;$$

в четвертом раунде (при $j = \overline{61, 80}$) – функция

$$f_j(X, Y, Z) = X \oplus Y \oplus Z$$

и константа

$$c_j = CA62C1D6h.$$

Цикл завершается сложением по модулю 2^{32} полученных значений *A*, *B*, *C*, *D* и *E* соответственно с *AA*, *BB*, *CC*, *DD* и *EE*:

$$A = A + AA,$$
$$B = B + BB,$$
$$C = C + CC,$$
$$D = D + DD,$$
$$E = E + EE,$$

конкатенация полученных значений *A, B, C, D* и *E* является результатом работы основного цикла.

Алгоритмы семейства SHA-2 значительно отличаются от более ранних версий SHA. Опишем алгоритм SHA-256.

Перед хешированием сообщению дополняется до длины, кратной 512 бит, аналогично SHA-1. После этого полученная последовательность разделяется на блоки по 512 бит (16 32-разрядных слов), каждый из которых поступает на вход функции сжатия SHA-256. В этом смысле мы имеем обычную итеративную хеш-функцию.

Функция сжатия обладает похожей итеративной структурой. Функция «расширения» блока на основе 16 слов исходного сообщения формирует *расширенное* сообщение – 64 слова, поступающие на вход 64 раундов функции сжатия, как показано на рис. 3.5, где РФ – раундовая функция.

Функция расширения блока *MS* описывается следующим образом. Первые 16 слов расширенного сообщения соответствуют исходным 16 словам блока, дальнейшие слова формируются по рекуррентной формуле:

$$\sigma_0^{\{256\}}(x) = ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x);$$
$$\sigma_1^{\{256\}}(x) = ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x);$$
$$w_t = \sigma_1^{\{256\}}(w_{t-2}) + w_{t-7} + \sigma_0^{\{256\}}(w_{t-15}) + w_{t-16},$$

где w_i – i-е слово расширенного сообщения. Здесь и далее $ROTR^a(x)$ обозначает слово x, циклически сдвинутое вправо на a разрядов.

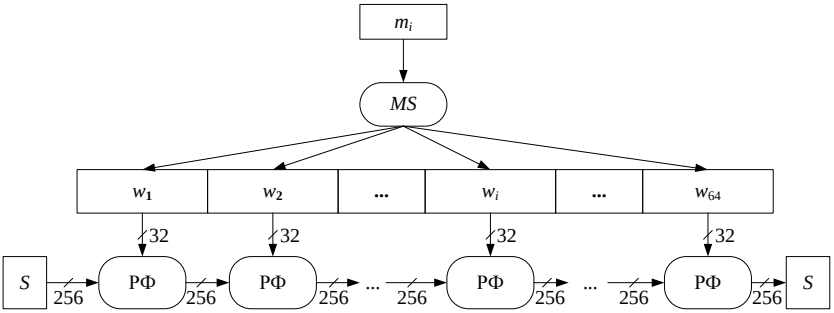


Рис. 3.5. Функция сжатия SHA-256:
m_i – блок сообщения; w_j – слова расширенного блока сообщения;
S – состояние определяемое регистрами a, b, c, d, e, f, g, h

Структура раунда изображена на рис. 3.6.

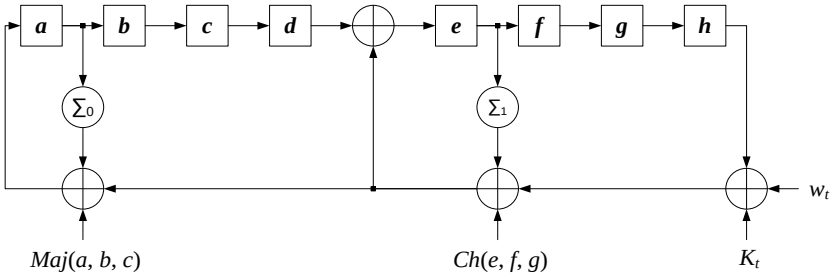


Рис. 3.6. Структура раунда SHA-2

На рисунке приняты следующие обозначения:

$$Ch\ (X, Y, Z) = XY \vee \overline{X}Z;$$

$$Maj\ (X, Y, Z) = XY \vee XZ \vee YZ;$$

$$\sum_{0}^{\{256\}}(x) = ROTR^{2}(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x);$$

$$\sum_{1}^{\{256\}}(x) = ROTR^{6}(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x).$$

Используется фиксированную последовательность из 64 32-разрядных констант (по количеству раундов). На каждом раунде применяется одна константа из этого массива, обозначенная на рис. 3.6 как K_t.

SHA-224 представляет собой SHA-256 с «обрезанным» до 224 бит хеш-образом и изменённым начальным состоянием.

SHA-384 получается аналогичным образом из SHA-512. Интересно, что по стандарту разрешается сокращать хеш-образ до любой длины, если это по каким-либо причинам необходимо.

3.4. Хеш-функции на основе симметричных блочных криптоалгоритмов

При использовании для построения $h(x)$ симметричных блочных криптоалгоритмов стойкость хеш-функции гарантируется стойкостью применяемого блочного шифра. Пусть

$$M = M_1 M_2 \ldots M_i \ldots M_t \ (i = \overline{1, t}) \ -$$

последовательность, состоящая из блоков, размер которых равен размеру ключа блочного шифра. Блоки M_i суть результат расширения блоков исходного сообщения меньшей длины (см, например, схему получения кода MDC в гл. 4). Наиболее надежные схемы получаются при использовании для вычисления текущего хеш-значения h_i функции зашифрования E_k, где ключ k – предыдущее хеш-значение h_{i-1}, хотя известны схемы, в которых в качестве k используется либо очередной блок сообщения M_i, либо $h_{i-1} \oplus M_i$. Наиболее известны следующие схемы формирования хеш-образа $h(M) = h_t$:

$$h_i = E_{h_{i-1}}(M_i) \oplus M_i;$$
$$h_i = E_{h_{i-1}}(M_i \oplus h_{i-1}) \oplus M_i \oplus h_{i-1};$$
$$h_i = E_{h_{i-1}}(M_i) \oplus M_i \oplus h_{i-1};$$
$$h_i = E_{h_{i-1}}(M_i \oplus h_{i-1}) \oplus M_i.$$

На рис. 3.7 показаны три варианта построения хеш-функции на основе функции зашифрования $E_k(M)$.

Структура, показанная на рис. 3.7,*в*, использована при проектировании хеш-функции Whirlpool, представленной в рамках европейского конкурса New European Schemes for Signatures, Integrity and Encryption (NESSIE). В качестве симметричного блочного шифра для реализации функции сжатия авторами (V. Rijmen, P. Barreto) использован модифицированный алгоритм AES.

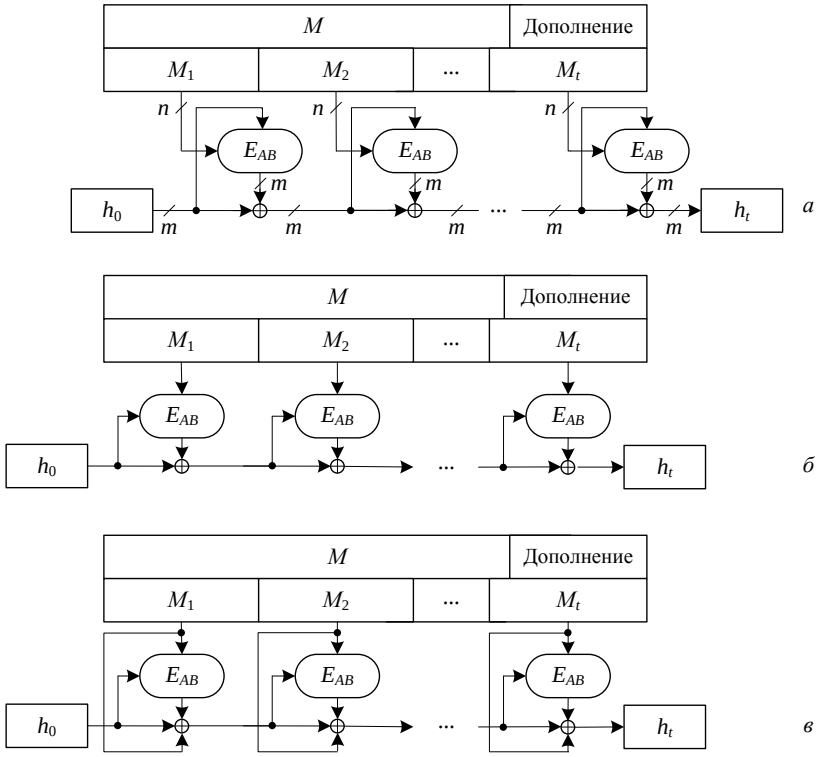


Рис. 3.7. Варианты построения хеш-функций на основе преобразования $E_k(M)$

Контрольные вопросы

1. Что такое парадокс дней рождения?
2. Какие три задачи являются вычислительно неразрешимыми для качественной криптографической хеш-функции?
3. Перечислите необходимые свойства качественной хеш-функции (не менее пяти).
4. Нарисуйте схему итеративной хеш-функции.

ГЛАВА 4. МЕТОДЫ АУТЕНТИФИКАЦИИ ИНФОРМАЦИИ

4.1. Аутентичность. Задача аутентификации информации

В компьютерных системах (КС), помимо обеспечения секретности информации, необходимо также решать не менее важную задачу *аутентификации* обрабатываемых массивов данных и пересылаемых сообщений. Информация может считаться *аутентичной*, когда потребитель имеет гарантии, во-первых, *целостности* информации и, во-вторых, ее *авторства*.

Иногда ошибочно считается, что задача аутентификации решается простым шифрованием. На первый взгляд кажется, что это действительно так. В зашифрованный массив данных трудно внести какие-то осмысленные изменения, потому что с вероятностью, близкой к единице, факты искажения становятся очевидными после расшифрования: например, вместо текста на русском языке появляется бессмысленный набор символов. Наконец, для классических криптосистем только пользователи, обладающие секретным ключом, могут зашифровать сообщение, и, значит, если получатель принял сообщение, зашифрованное на его секретном ключе, он может быть уверен в его авторстве.

На самом деле приведенные рассуждения ошибочны. Искажения, внесенные в зашифрованные данные, становятся очевидными после расшифрования только в случае большой *избыточности исходных данных*. Эта избыточность имеет место лишь в некоторых частных случаях (например, когда исходная информация является текстом на естественном или искусственном языке). В общем случае требование избыточности данных может не выполняться, а это означает, что после расшифрования модифицированных данных они по-прежнему могут поддаваться интерпретации. Кроме того, при использовании симметричной криптосистемы факт успешного расшифрования данных, зашифрованных на секретном ключе, может подтвердить их авторство лишь для самого получателя. Третий участник информационного обмена (арбитр) при возникновении споров не сможет сделать однозначного вывода об авторстве информационного массива, так как его автором может быть каждый из обладателей секрет-

ного ключа, а их как минимум двое. Таким образом, задачи обеспечения секретности и аутентичности должны решаться различными методами [8].

4.2. Имитозащита информации. Контроль целостности потока сообщений

На всех этапах своего жизненного цикла информация может подвергаться случайным и умышленным деструктивным воздействиям. Для обнаружения случайных искажений информации применяются корректирующие коды, которые в некоторых случаях позволяют не только зафиксировать факт наличия искажений информации, но и локализовать и исправить эти искажения. Умышленные деструктивные воздействия чаще всего имеют место при хранении информации в памяти компьютера и ее передаче по каналам связи. В таком случае полностью исключить возможность несанкционированных изменений в массивах данных нельзя. Поэтому крайне важно оперативно обнаружить данные изменения, поскольку в этом случае ущерб, нанесенный законным пользователям, будет минимальным. Целью противника, навязывающего ложную информацию, является выдача ее за подлинную, поэтому своевременная фиксация факта наличия искажений в массиве данных сводит на нет все усилия злоумышленника. Таким образом, *под имитозащитой понимают не исключение возможности несанкционированных изменений информации, а совокупность методов, позволяющих достоверно зафиксировать факты изменений, если они имели место* [8, 25, 29].

Для обнаружения искажений в распоряжении законного пользователя (например, получателя информации при ее передаче) должна быть некая процедура проверки $T(M)$, дающая на выходе 1, если в массиве данных M отсутствуют искажения, или 0, если такие искажения имеют место. Для ограничения возможностей противника по подбору информационной последовательности M' $(M' \neq M)$, где M – правильная последовательность (без искажений), такой, что $T(M')=1$, идеальная процедура этой проверки должна обладать следующими свойствами [8]:

невозможно найти такое сообщение M' способом, более эффективным, чем полный перебор по множеству допустимых значений M (такая возможность в распоряжении противника имеется всегда);

вероятность успешно пройти проверку у случайно выбранного сообщения M' не должна превышать заранее установленного значения.

Учитывая, что в общем случае все возможные значения M могут являться допустимыми, второе свойство требует *внесения избыточности* в защищаемый массив данных. При этом чем больше разница N между размером преобразованного избыточного M^* и размером исходного M массивов, тем меньше вероятность принятия искаженных данных за подлинные. Эта вероятность равна 2^{-N}.

На рис. 4.1 показаны некоторые возможные варианты внесения такой избыточности. В роли неповторяющегося блока данных *nrb* могут выступать метка времени, порядковый номер сообщения и т. п., а в роли контрольного кода – имитоприставка или электронная подпись. Имитоприставкой принято называть контрольный код, который формируется и проверяется с помощью одного и того же секретного ключа.

Использование блока *nrb* позволяет контролировать целостность потока сообщений, защищая от *повтора*, *задержки*, *переупорядочивания* или их *утраты*. При использовании в качестве *nrb* порядкового номера получатель, приняв (i + 1)-е сообщение, проверяет равенство $nrb_{i+1} = nrb_i + 1$, т.е. что его номер на единицу больше номера предыдущего i-го сообщения. При использовании в качестве *nrb* метки времени получатель контролирует, чтобы времена отправки и приема сообщений соответствовали друг другу с учетом задержки в канале связи и разности показаний часов отправителя и получателя. Целостность потока сообщений можно также контролировать, используя зашифрование со сцеплением сообщений (рис. 4.2).

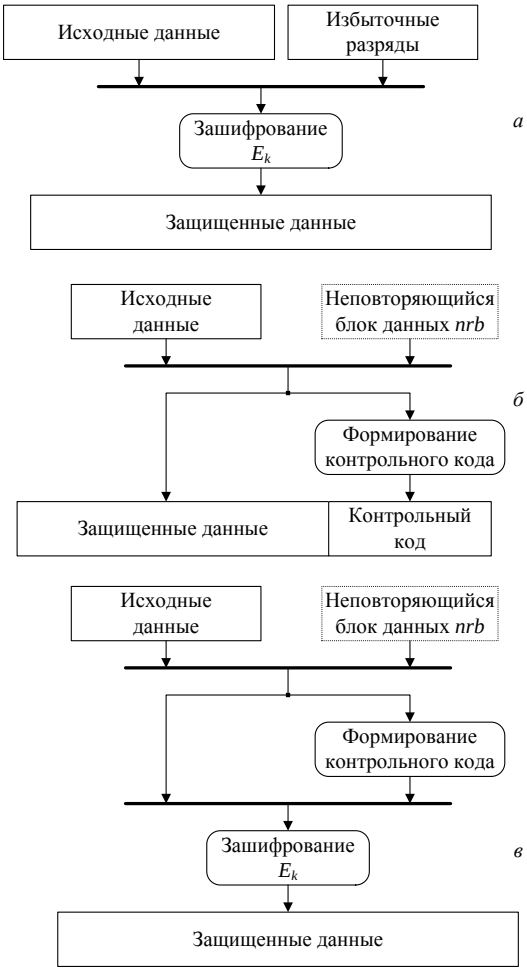


Рис. 4.1. Схемы контроля целостности:
а – с использованием зашифрования;
б – с формированием контрольного кода;
в – с формированием контрольного кода и зашифрованием

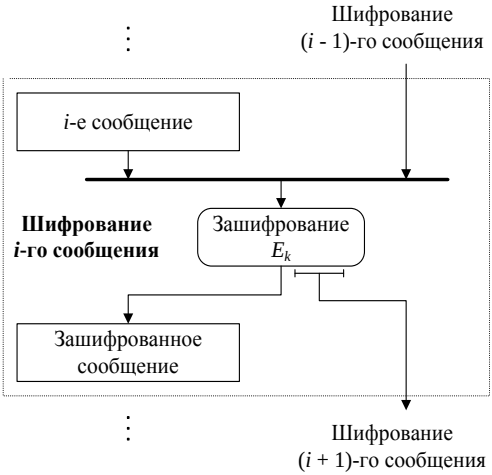


Рис. 4.2. Контроль целостности потока сообщений

Рассмотрим схему на рис. 4.1,*б*. Самый естественный способ преобразования информации с внесением избыточности – добавление к исходным данным контрольного кода s фиксированной разрядности N, вычисляемого как некоторая функция от этих данных:

$$M^* = (M,\ s) = (M,\ f(M)),\ |s| = N.$$

В такой ситуации выделение исходных данных из преобразованного массива M^* суть простое отбрасывание контрольного кода s. Проверка же целостности заключается в вычислении для содержательной части M' полученного массива данных контрольного кода $s' = f(M')$ и сравнении его с переданным значением s. Если они совпадают, сообщение считается подлинным, в противном случае ложным:

$$T(M') = \begin{cases} 1, \text{ если } s = f(M'); \\ 0, \text{ если } s \neq f(M'). \end{cases}$$

Функция f формирования контрольного кода обязана удовлетворять следующим требованиям:

она должна быть вычислительно необратимой, т.е. подобрать массив данных под заданный контрольный код можно

только путем полного перебора по пространству возможных значений M;

у противника отсутствует возможность сформировать ложный массив данных (или ложное сообщение) M' и снабдить его корректно вычисленным контрольным кодом $s = f(M')$.

Второе свойство можно обеспечить двумя способами: либо сделать функцию f зависимой от некоторого секретного параметра (ключа), либо пересылать контрольный код отдельно от защищаемых данных [8].

4.3. Криптографические методы контроля целостности

Можно выделить два основных криптографических подхода к решению задачи защиты информации от несанкционированных изменений данных, которые предполагают формирование:

1) *кода аутентификации сообщений MAC* (Message Authentication Code);
2) *кода обнаружения манипуляций с данными MDC* (Manipulation Detection Code).

В табл. 4.1 приведены сравнительные характеристики указанных двух подходов [8, 24]. Главное различие между кодами MAC и MDC заключается в том, что в первом случае для формирования контрольного кода требуется секретная информация, а во втором – нет.

4.4. Код аутентификации сообщений

Формирование кода MAC с использованием функции зашифрования блочного шифра официально или полуофициально закреплено во многих государственных стандартах шифрования. Имитоприставка ГОСТ 28147-89 – классический пример кода MAC. Код аутентификации сообщений может формироваться в режимах CBC или CFB, обеспечивающих зависимость последнего блока шифротекста от всех блоков открытого текста. В случае использования преобразования E_k для выработки контрольного кода требования к нему несколько отличаются от традиционных, используемых при зашифровании: во-первых, не требуется свой-

ство обратимости; во-вторых, его криптостойкость может быть снижена (например, за счет уменьшения числа раундов шифрования, как в ГОСТ 28147-89). Действительно, в случае выработки кода MAC преобразование всегда выполняется в одну сторону, при этом в распоряжении противника есть только зависящий от всех блоков открытого текста контрольный код, в то время как при зашифровании у него имеется набор блоков шифротекста, полученных с использованием одного секретного ключа.

Таблица 4.1

Сравнительная характеристика MAC и MDC

Параметр	MAC	MDC
Используемое преобразование	Функция зашифрования блочного шифра	Хеш-функция
Секретная информация	Секретный ключ	Нет
Возможность для противника вычислить контрольный код	Отсутствует	Присутствует
Хранение и передача контрольного кода	Вместе с защищаемыми Данными	Отдельно от защищаемых данных
Дополнительные условия	Требует предварительного распределения ключей	Необходим аутентичный канал для передачи контрольного кода
Области использования	Защита при передаче данных	Защита при разовой передаче данных, контроль целостности хранимой информации

4.5. Код обнаружения манипуляций с данными

MDC есть результат действия хеш-функции. Иначе говоря, MDC – это *хеш-образ сообщения M*, к которому применили хеш-функцию, т.е. $s = h(M)$. Основное требование к хеш-функции: не должно существовать способа определения массива данных *M*, имеющего заданное значение хеш-образа $h(M)$, отличного от перебора по всему множеству возможных значений *M*. Наиболее простой способ построения хеш-функции основан на использовании вычислительной необратимости относительно ключа *k*

функции зашифрования E_k любого блочного шифра. Даже при известных блоках открытого M и закрытого $c = E_k(M)$ текстов ключ k не может быть определен иначе как перебором по множеству всех возможных значений. Итак, схема формирования хеш-образа сообщения M, обладающая гарантированной стойкостью, равной стойкости используемого шифра, может быть следующей:

1) массив данных M разбивается на блоки фиксированного размера, равного размеру ключа $|k|$ используемого блочного шифра, т. е.

$$M = M_1 M_2 M_3 \ldots M_t,$$

$$|M_1| = |M_2| = |M_3| = \ldots = |M_{m\text{-}1}| = |k|,\ 0 < |M_t| \le |k|;$$

2) если последний блок M_t неполный, он дополняется каким-либо образом до нужного размера $|k|$;

3) хеш-образ сообщения вычисляется следующим образом:

$$s = h(M) = E_{M_t}(\ldots E_{M_3}(E_{M_2}(E_{M_1}(s_0)))),$$

где s_0 – синхропосылка, обычно выбирают $s_0 = 0$.

Задача подбора массива данных

$$M' = M'_1 M'_2 M'_3 \ldots M'_t$$

под заданный контрольный код s эквивалентна системе уравнений, которую необходимо решить для определения ключа для заданных блоков открытого и закрытого (в режиме простой замены) сообщений. Однако в рассматриваемой ситуации нет необходимости решать всю систему

$$E_{M'_1}(s_0) = s_1;$$

$$E_{M'_2}(s_1) = s_2;$$

$$E_{M'_3}(s_2) = s_3;$$

$$\ldots$$

$$E_{M'_t}(s_{M'\text{-}1}) = s,$$

достаточно решить уравнение

$$E_{M'_i}(s_{i\text{-}1}) = s_i$$

относительно M'_i, остальные блоки массива M могут быть произвольными. Но и эта задача в случае использования надежной функции E_k вычислительно неразрешима.

К сожалению, приведенная схема формирования MDC не учитывает наличия так называемых *побочных* ключей шифра. Если для $k' \neq k$ справедливо

$$E_{k'}(M_i) = E_k(M_i),$$

где M_i – некоторый блок открытого текста, то такой код k' и является побочным ключом, т.е. ключом, дающим при зашифровании блока M_i точно такой же результат, что и истинный ключ k. Обнаружение противником побочного ключа при дешифровании сообщения не является особым успехом, так как с вероятностью, близкой к 1, на этом найденном побочном ключе он не сможет правильно расшифровать другие блоки закрытого текста, учитывая, что для различных блоков побочные ключи в общем случае также различны. В случае выработки кода MDC ситуация прямо противоположная: обнаружение побочного ключа означает, что противник нашел такой ложный блок данных, использование которого не изменяет контрольного кода [8].

Для уменьшения вероятности навязывания ложных данных в результате нахождения побочных ключей, при преобразовании применяются не сами блоки исходного сообщения, а результат их *расширения* по некоторому алгоритму. Под расширением понимается процедура получения блока данных большего размера из блока данных меньшего размера. Например, для криптоалгоритма, в котором размер ключа равен 256 бит, возможна следующая схема расширения 128-битового блока в 256-битовый:

исходный блок:

$M_i = (b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15} b_{16})$;

расширенный блок:

$Ext(M_i) = (b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15} b_{16} b_1 b_4 b_7 b_{10} b_{13} b_{16} b_3 b_6 b_9 b_{12} b_{15}$
$b_2 b_5 b_8 b_{11} b_{14})$,

где b_i – байты блока данных [8].

4.6. Код HMAC

Существует вариант построения кода MAC на основе использования секретного ключа и функции хеширования, при котором хешированию подвергается результат конкатенации секретного ключа и исходного сообщения, поэтому, как и в классическом случае, у противника, не знающего ключа, отсутствует возможность вычислить контрольный код.

Для повышения безопасности подобного алгоритма получения MAC, создана схема вложенного (nested) MAC, в которой хеширование выполняется дважды (рис. 4.3). В стандарте NIST FIPS 198 вложенный MAC назван HMAC (hashed MAC). Схема формирования HMAC (рис. 4.4) более сложная, чем показанная на рис. 4.3. Алгоритм формирования HMAC имеет следующий вид:

1. Сообщение делится на N блоков, по b бит в каждом.
2. Секретный ключ дополняется слева нулями до получения b-разрядного ключа. Рекомендуется, чтобы секретный ключ до операции дополнения имел длину, большую n, где n – разрядность HMAC.
3. Выполняется операция XOR результат шага 2 с константой *ipad* (input pud). Величина *ipad* суть $b/8$ повторений последовательности 36h.
4. Полученный блок добавляется слева к N-блоковому сообщению.
5. Результат шага 4 хешируется и создается n-разрядный хеш-образ, называемый Intermediate HMAC.
6. Результат шага 5 дополняется слева нулями до получения b-разрядного блока.
7. Шаги 2 и 3 повторяются с константой *opad* (output pud). Величина *opad* суть $b/8$ повторений последовательности 5Ch.
8. Результат шага 7 добавляется слева к блоку, полученному на шаге 6.
9. Результат шага 8 хешируется с использованием той же хеш-функции и создается финальный HMAC.

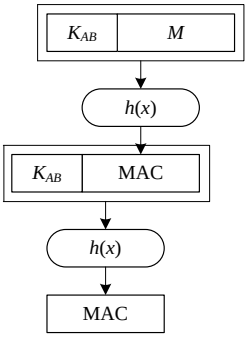


Рис. 4.3. Схема формирования кода nested MAC

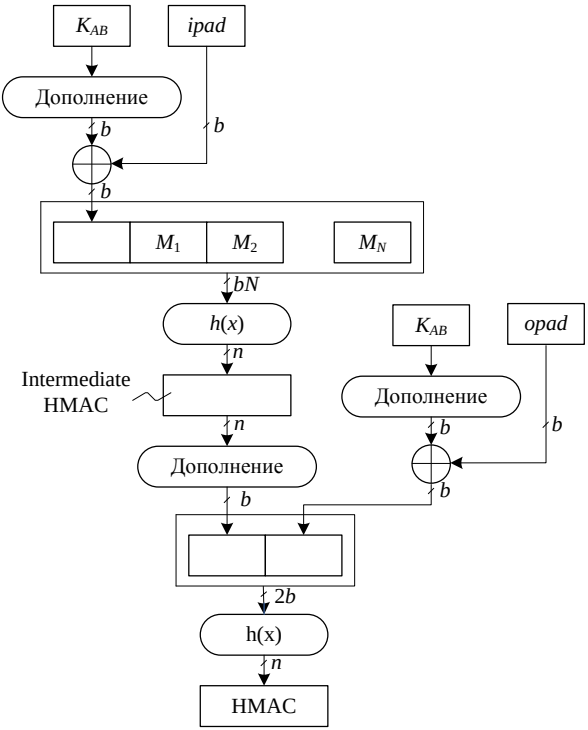


Рис. 4.4. Схема формирования кода HMAC

4.7. Код CMAC

В стандарте NIST FIPS 113 определена схема формирования кода аутентификации сообщений, названная CMAC (или CBCMAC). Схема формирования CMAC показана на рис. 4.5 и 4.6.

Оригинальное сообщение делится на N блоков по m разрядов в каждом. Если последний блок имеет длину, меньшую m, он дополняется единичным битом и последующими нулевыми битами до требуемой длины. Пусть n – разрядность CMAC. Первый блок сообщения M_1 шифруется на симметричном ключе K_{AB}, в результате получается m-разрядный блок шифротекста. Этот блок поразрядно суммируется по модулю 2 со следующим блоком сообщения M_2, в результате вновь получается m-разрядный блок шифротекста. Процесс повторяется до тех пор, пока не будет зашифрован последний блок сообщения M_N, после чего n левых разрядов результата объявляются кодом CMAC. В дополнение к симметричному ключу K_{AB} для получения CMAC на последнем шаге шифрования используется другой ключ k. Он формируется в результате шифрования на ключе K_{AB} блока, состоящего из m нулевых бит, после чего результат умножается на x или x^2 в поле $GF(2^m)$, если операции дополнения последнего блока соответственно не проводилась или проводилась.

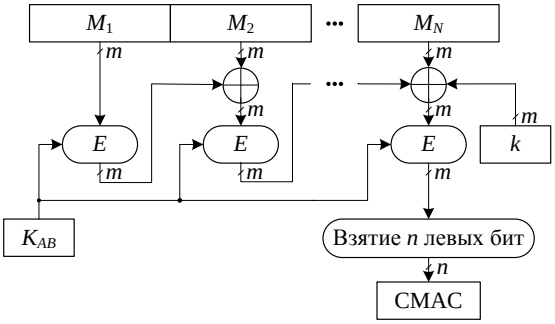


Рис. 4.5. Схема формирования кода CMAC

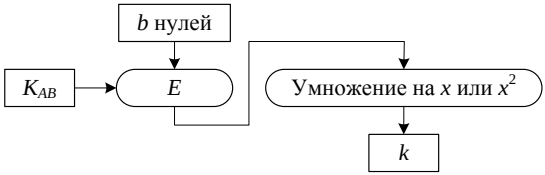


Рис. 4.6. Схема генерации ключа *k* для последнего шага формирования CMAC

4.8. Идентификация, аутентификация и авторизация

Чтобы исключить работу с системой незаконных пользователей, необходима процедура распознавания системой каждого законного пользователя (или групп пользователей). Для этого в защищенном месте система обязана хранить информацию, по которой можно опознать пользователя, а пользователь (при входе в систему, выполнении определенных действий, доступе к ресурсам) обязан себя идентифицировать, т.е. указать *идентификатор*, присвоенный ему в данной системе. Получив идентификатор, система проводит его *аутентификацию*, т.е. проверяет его содержательность (подлинность) – принадлежность множеству идентификаторов. Если бы идентификация не дополнялась аутентификацией, то сама идентификация теряла бы всякий смысл. Обычно устанавливается ограничение на число попыток предъявления некорректного идентификатора. Аутентификация пользователя может быть основана на следующих принципах:

предъявлении пользователем пароля (рис. 4.7);

предъявлении пользователем доказательств, что он обладает секретной ключевой информацией;

ответах на некоторые тестовые вопросы;

предъявлении пользователем некоторых неизменных признаков, неразрывно связанных с ним;

предоставлении доказательств того, что он находится в определенном месте в определенное время;

установлении подлинности пользователя некоторой третьей доверенной стороной.

Процедуры аутентификации должны быть устойчивы к *подлогу*, *подбору* и *подделке*.

После распознавания пользователя система должна выяснить, какие права предоставлены этому пользователю, какую информацию и каким образом (читать, записывать, модифицировать или удалять) он может использовать, какие программы может выполнять, какими ресурсы ему доступны, а также другие вопросы подобного рода. Этот процесс называется *авторизацией*. Таким образом, вход пользователя в систему состоит из идентификации, аутентификации и авторизации. В процессе дальнейшей работы иногда может появиться необходимость в дополнительной авторизации по отношению к каким-либо действиям.

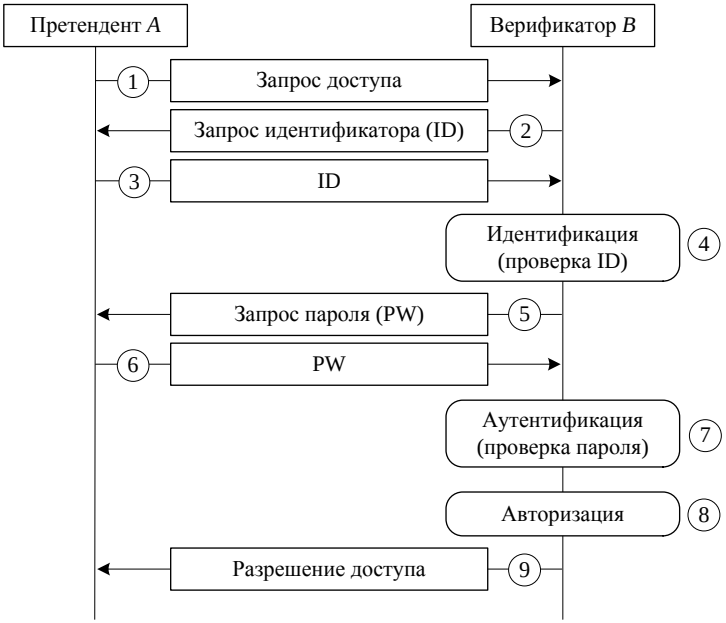


Рис. 4.7. Процессы идентификации и аутентификации

Существуют различные механизмы реализации разграничения доступа. Например, каждому ресурсу (или компоненту) системы может быть сопоставлен *список управления доступом*, в котором указаны идентификаторы всех пользователей, кото-

рым разрешен доступ к данному ресурсу, а также определено, какой именно доступ разрешен. При обращении пользователя к конкретному ресурсу система проверяет наличие у данного ресурса списка управления доступом и, если он существует, проверяет, разрешено ли этому пользователю работать с данным ресурсом в запрошенном режиме. Другим примером реализации механизма авторизации пользователя является *профиль пользователя* – список, ставящий в соответствие всем идентификаторам пользователей перечень объектов, к которым разрешен доступ данному пользователю с указанием типа доступа. Может быть организована системная структура данных, так называемая *матрица доступа*, которая представляет собой таблицу, столбцы которой соответствуют идентификаторам всех системных ресурсов, а строки – идентификаторам всех зарегистрированных пользователей. На пересечении *i*-го столбца и *j*-й строки таблицы администратор системы указывает разрешенный тип доступа владельца *i*-го идентификатора к *j*-му ресурсу.

Доступ к механизмам авторизации должны иметь только специальные системные программы ОБИ, а также строго ограниченный круг пользователей, отвечающих за безопасность системы. Рассматриваемые механизмы должны быть тщательно защищены от случайного или преднамеренного доступа неавторизованных пользователей. Многие атаки на информационные системы нацелены именно на вывод из строя или обход средств разграничения доступа.

Аналогичные действия осуществляются в системе и при аутентификации других субъектов взаимодействия (*претендентов*), например прикладных процессов или программ, с системой (*верификатором*).

В отличие от *аутентификации субъекта* взаимодействия процедура *аутентификации объекта*, устанавливая подлинность электронной почты, банковского счета и т.п., проверяет факт принадлежности данного объекта владельцу указанного идентификатора.

4.9. Аутентификация субъекта

Цель субъекта взаимодействия при аутентификации – доказать верификатору подлинности предъявленного идентификатора.

Классическим средством аутентификации субъекта являются парольные схемы. При этом для устранения последствий несанкционированного доступа противника к информации, хранящейся в памяти компьютера верификатора, передаваться может не сам пароль *pw* (password), а его хеш-образ $q = h(pw)$ (рис. 4.8). Функция $h(pw)$ в этой ситуации может быть определена как

$$h(pw) = E_{pw}(ID),$$

если длина пароля и длина ключа k функции зашифрования E_k одинаковы, или как

$$h(pw) = E_{pw \oplus k}(ID),$$

если длина пароля меньше длины ключа. Верификатор заранее вычисляет значения $h(pw)$ и для каждого идентификатора *ID* хранит значения q. Претендент, прошедший идентификацию, вводит пароль pw'; верификатор, получив $q' = h(pw')$, проверяет равенство $q = q'$ и при его выполнении заключает, что был введен правильный пароль, и разрешает доступ. Противник, даже узнав значение q, не сможет определить pw в силу вычислительной необратимости функции $h(pw)$.

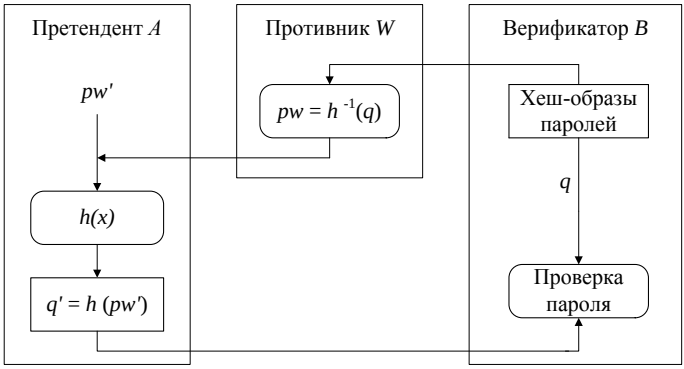


Рис. 4.8. Простейшая парольная схема аутентификации (вариант 1)

Однако рассмотренная схема не защищает от противника, который может передавать информацию, подключаясь непосредственно к линии связи. В этой ситуации может помочь, схема, показанная на рис. 4.9, в которой процедура вычисления $q' = h(pw')$ возложена на верификатора.

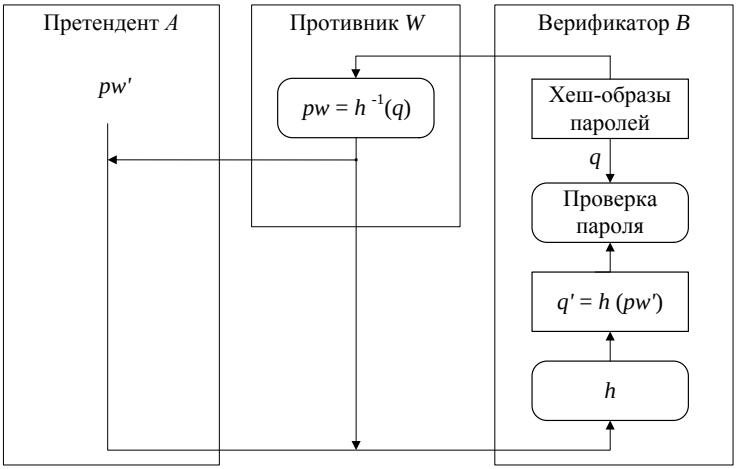


Рис. 4.9. Простейшая парольная схема аутентификации (вариант 2)

Обе рассмотренные схемы не защищают от *атаки перехвата и повтора*, когда противник записывает информацию, передаваемую претендентом, и организует ее повторение для входа в систему. Для устранения последствий перехвата информации, передаваемой претендентом, или несанкционированного доступа противника к информации, хранящейся в памяти компьютера верификатора, может быть рекомендована схема (рис. 4.10), предполагающая использование двух хеш-функций h_1 и h_2, причем результат работы второй из них зависит от неповторяющегося блока данных *nrb*.

На рис. 4.11 рассмотрена модификация схемы, показанной на рис. 4.9. Особенностью данного варианта является использование при создании пароля случайной строки, называемой *salt*, после выполнения конкатенации введенного пароля *pw'* и

salt, хранимой на стороне верификатора, результат хешируется и сравнивается с хранимым значением $h(pw \| salt)$.

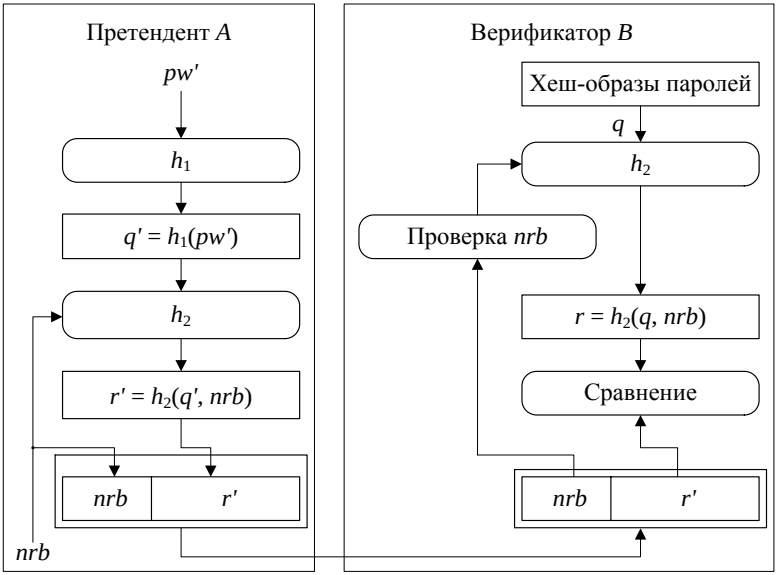


Рис. 4.10. Парольная схема аутентификации, защищенная от повтора

На рис. 4.12 показана схема с использованием одноразового пароля. Пользователь системы и верификатор договариваются об используемых хеш-функции, пароле P_0 и состоянии счетчика n. Система (верификатор) вычисляет $h^n(P_0)$, где h^n означает n-кратное применение хеш-функции. Иначе говоря, справедлива цепочка равенств $h^n(x) = h(h^{n-1}(x))$, $h^{n-1}(x) = h(h^{n-2}(x))$, …, $h^2(x) = h(h(x))$, $h^1(x) = h(x)$. Вначале, при первом доступе система хранит $h^n(P_0)$ и значение n. На рис. 4.12 рассмотрен принцип осуществления первого доступа в систему.

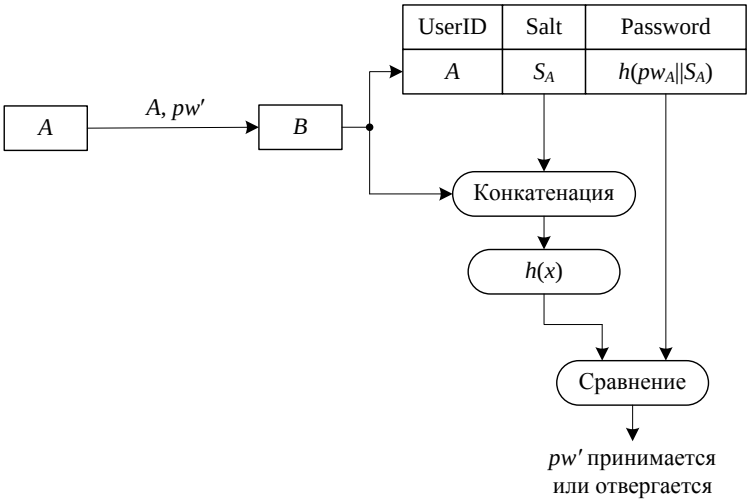


Рис. 4.11. Использование случайной строки *salt* при проверке пароля

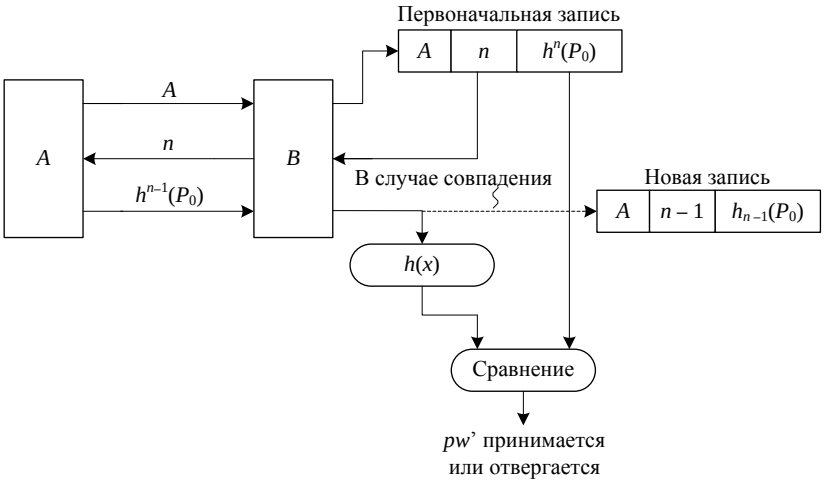


Рис. 4.12. Использование одноразового пароля при аутентификации субъекта

Аутентификация субъекта может быть как *односторонней*, так и *взаимной*. В первом случае процедуру аутентификации проходит один субъект, во втором – аутентифицируют друг друга два взаимодействующих субъекта, например связывающиеся между собой по линиям связи. Взаимная аутентификация

не есть простое объединение двух сеансов односторонней аутентификации, так как в последнем случае противник легко может осуществить атаку перехвата и повтора, выдавая себя за верификатора перед претендентом и за претендента перед верификатором.

Проверка подлинности предполагает применение неповторяющихся блоков данных, в качестве которых используются временные метки, механизмы *запрос-ответ* и процедуры *рукопожатия* (handshake) (рис. 4.13). Метки времени позволяют регистрировать время отправки конкретного сообщения, что дает возможность получателю определить, насколько «устарело» пришедшее сообщение, а значит, защититься от повтора. При использовании меток времени возникает проблема *допустимого времени задержки*, связанная, во-первых, с невозможностью мгновенной передачи сообщения, а во-вторых, невозможностью абсолютной синхронизации хода часов получателя и отправителя. Механизм «запрос–ответ» предполагает включение пользователем *A* в сообщение для пользователя *B* запроса x_A – некоторого случайного числа. Перед ответом пользователь *B* обязан выполнить над числом x_A некоторую операцию, например вычислить хеш-образ $h(x_A)$. Получив ответ с правильным результатом вычислений, пользователь *A* может быть уверен, что *B* – подлинный. Процедура рукопожатия заключается во взаимной проверке ключей, используемых субъектами взаимодействия. Последние признают друг друга законными партнерами, если докажут друг другу, что обладают правильными ключами [25].

4.10. Аутентификация объекта

В процессе аутентификации объекта, иногда называемой *аутентификацией источника данных,* проверяется подлинность идентификатора, представленного с некоторыми данными. В отличие от аутентификации субъекта в этой ситуации претенденту не нужно быть активным участником процесса ау-

тентификации. Данный тип аутентификации (рис. 4.14), по сути, ничем не отличается от процедуры контроля целостности. Для аутентификации объекта применяется шифрование симметричным алгоритмом, выработка имитоприставки или электронной подписи. Первые два варианта применяются в том случае, когда претендент и верификатор доверяют друг другу. Если необходимо иметь возможность доказательства подлинности идентификатора третьей стороне (при условии, что верификатор не имеет возможности изменить массив данных M), например необходима юридическая значимость пересылаемых электронных документов, требуется электронная подпись.

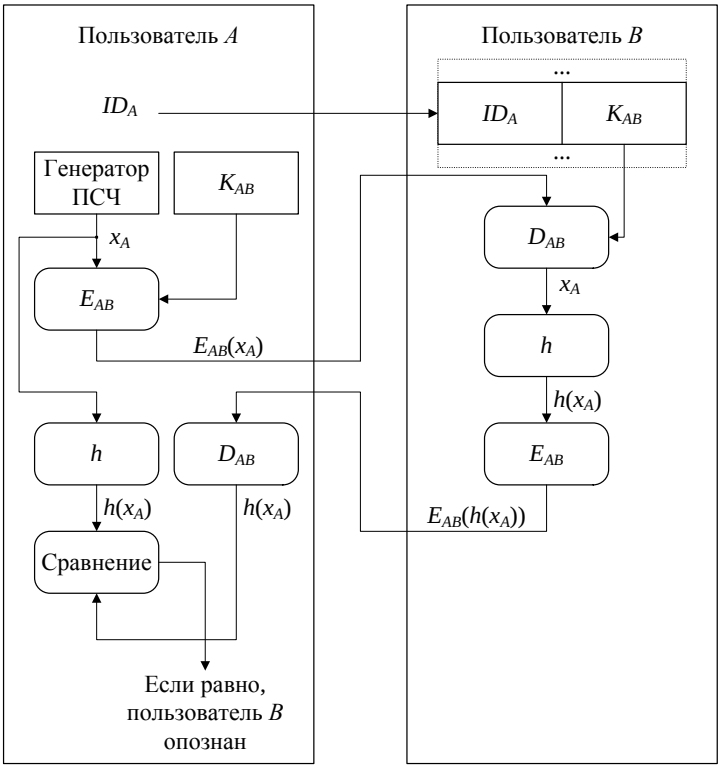


Рис. 4.13. Механизм «запрос–ответ» и процедура handshake

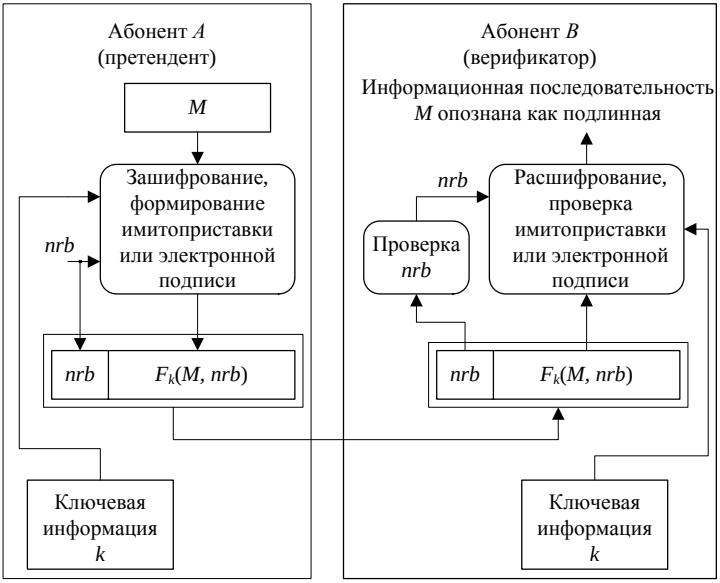


Рис. 4.14. Схема аутентификации объекта

Контрольные вопросы

1. Сформулируйте требования к качественному контрольному коду целостности информации.
2. Что такое имитозащита информации?
3. Перечислите методы аутентификации объектов информационного взаимодействия.
4. Назовите методы аутентификации субъектов информационного взаимодействия.
5. В чем заключается принцип вычисления кода аутентификации сообщений?
6. Сформулируйте принцип вычисления кода обнаружений манипуляций с данными.
7. Дайте сравнительную характеристику кодам MAC и MDC.
8. В чем заключается принцип вычисления кода HMAC.
9. Как осуществляется аутентификация пользователей.

10. В каком виде надежнее хранить пароли пользователей: в шифрованном или хешированном?
11. Как осуществляется контроль целостности потока сообщений?
12. Что такое одноразовый пароль?

ГЛАВА 5. КРИПТОСИСТЕМЫ С ОТКРЫТЫМ КЛЮЧОМ

Основная проблема, возникающая при использовании криптосистем с секретным ключом, рассмотренных в гл. 2, – проблема распределения ключей. Учитывая, что процедуры зашифрования и расшифрования выполняются с помощью одного и того же ключа, в системе с n абонентами потребуется $n(n-1)/2$ ключей, которые не только должны быть сгенерированы, но и надежным образом распределены среди всех участников информационного обмена. Необходимость наличия недоступных для противника секретных каналов для обмена ключами делает такой подход к построению системы секретной связи чрезвычайно дорогостоящим мероприятием, практически не применимым в коммерческих сетях связи, в региональных и глобальных информационных системах и, естественно, абсолютно недоступен частным лицам. Другой принципиальный недостаток криптосистем с секретным ключом – невозможность разрешения конфликтов. При их возникновении арбитр оказывается в безвыходной ситуации, так как секретную ключевую информацию k_{AB} знают как минимум двое участников информационного обмена. А это означает, что в рамках системы, показанной на рис. 2.1, невозможно обеспечить юридическую значимость пересылаемых электронных документов.

Еще в сороковых годах прошлого столетия К. Шеннон предложил строить шифр таким образом, чтобы задача его вскрытия была эквивалентна решению некоей математической задачи, требующего объема вычислений, недоступного для современных компьютеров. Реализация этой идеи стала возможной, когда в 1976 г. Диффи и Хеллман предложили принципиально новый способ организации секретной связи без предварительного обмена ключами, так называемое шифрование с открытым ключом. При этом для зашифрования и расшифрования используются разные ключи и знание одного из них не дает практической возможности определить второй. В результате ключ зашифрования может быть сделан открытым без потери стойкости шифра, и лишь ключ расшифрования держится получателем в секрете.

5.1. Односторонние функции

Базовым в новом направлении криптографии является понятие *односторонней функции* [4]. По заданному аргументу x легко вычислить значение этой функции $F(x)$, в то же время определение x из $F(x)$ трудновычислимо, иначе говоря, нет алгоритма для решения этой задачи с полиномиальным временем работы. Теоретически x по известному значению $F(x)$ можно найти всегда, используя метод полного перебора, т.е. проверяя по очереди все возможные значения x до тех пор, пока соответствующее значение $F(x)$ не совпадет с заданным. Однако практически при значительной размерности множества X такой подход неосуществим.

Итак, односторонней функцией называется функция $F: X \to Y$, обладающая двумя свойствами:

1) существует полиномиальный алгоритм вычисления значений $F(x)$, иначе говоря, задача нахождения значений $y = F(x)$ при известных значениях x вычислительно разрешима;
2) не существует полиномиального алгоритма инвертирования функции F, т.е. задача нахождения значений x по известным значениям $y = F(x)$ вычислительно неразрешима.

До сих пор ни для одной функции кандидата на звание односторонней не доказано, что она действительно является односторонней.

Пример кандидата на звание односторонней функции – *модульное возведение в степень*, т.е. функция

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

где p – большое простое число, ω – примитивный элемент поля $GF(p)$. То, что эта функция может быть эффективно вычислена даже при разрядности параметров в несколько сотен знаков, можно показать на примере: ω^{25} можно вычислить с помощью всего лишь шести операций умножения (умножением считается и возведение в квадрат), так как

$$\omega^{25} = \left(\left(\left(\omega^2 \cdot \omega\right)^2\right)^2\right)^2 \cdot \omega .$$

При вычислении $\omega^x \bmod p$ взятие модуля во избежания получения очень больших чисел следует делать после каждого умножения. Пусть l – разрядность экспоненты x, тогда при вычислении $\omega^x \bmod p$ по приведенному в [3] рекурсивному алгоритму потребуется выполнить от l до $2l$ модульных операций умножения:

function expmod (ω, *x*, *p*)
  if $x = 0$ *then return* 1
  if $x = 0 \bmod 2$ *then return*

$$\left(\text{expmod}\left(\omega, \frac{x}{2}, p\right)\right)^2 \bmod p$$

  else return $(\omega \cdot \text{expmod}\ (\omega, x - 1, p)) \bmod p$

Задача вычисления функции, обратной модульному возведению в степень, называется *задачей дискретного логарифмирования*. На сегодняшний день неизвестно ни одного эффективного алгоритма вычисления дискретных логарифмов больших чисел.

Односторонняя функция в качестве функции зашифрования неприменима, так как, хотя $F(x)$ – надежно зашифрованное сообщение x, никто, в том числе и законный получатель, не сможет восстановить x. Обойти эту проблему можно с помощью *односторонней функции с секретом* (one-way trapdoor function). Такова, например, функция $E_k : X \to Y$, имеющая обратную $D_k : Y \to X$, однако узнать обратную функцию только по E_k без знания секрета k невозможно.

Таким образом, односторонней функцией с секретом k называется функция $E_k : X \to Y$, зависящая от параметра k и обладающая тремя свойствами:

1) при любом k существует полиномиальный алгоритм вычисления значений $E_k(x)$;
2) при неизвестном k не существует полиномиального алгоритма инвертирования E_k;

3) при известном k существует полиномиальный алгоритма инвертирования E_k.

Функцию E_k можно использовать для зашифрования информации, а обратную ей функцию D_k – для расшифрования, так как при всех $x \in X$ справедливо

$$D_k(E_k(x)) = x.$$

При этом подразумевается, что тот, кто знает, как зашифровывать информацию, вовсе не обязательно должен знать, как расшифровывать. Так же, как и в случае с односторонней функцией, вопрос о существовании односторонних функций с секретом открыт. Для практической криптографии найдено несколько функций, кандидатов на звание односторонней функции с секретом. Для них второе свойство не доказано, однако известно, что задача инвертирования эквивалентна некоторой хорошо изученной и давно известной трудной математической задаче. Это означает, что второе требование к односторонней функции с секретом заменяется более слабым условием: при неизвестном k *вероятно* не существует полиномиального алгоритма инвертирования E_k.

Применение рассмотренных функций для шифрования информации позволяет [3, 4, 29]:

избавится от необходимости надежных (а точнее, аутентичных) каналов связи для предварительного обмена ключами;

свести проблему взлома шифра к решению трудной математической задачи, т.е., в конечном счете, принципиально подругому подойти к обоснованию стойкости криптосистемы;

решать средствами криптографии задачи, отличные от шифрования, например задачу обеспечения юридической значимости электронных документов.

5.2. Модель криптосистемы с открытым ключом

Криптосистема с открытым ключом (public key cryptosystem) (рис. 5.1), основанную на применении односторонних функций с секретом, можно реализовать следующим образом. Пользователь *B*, который хочет получать конфиденциальные сообщения,

должен выбрать одностороннюю функцию E_B с секретом (ключом) k_B. Он сообщает всем заинтересованным описание своей функции зашифрования E_B, например, публикует его в справочнике открытых ключей. При этом ключ k_B, а, следовательно, и алгоритм расшифрования D_B держатся в секрете. Если теперь пользователь *A* хочет послать *B* секретное сообщение *m*, то он находит в справочнике функцию E_B, а затем вычисляет $c = E_B(m)$ и посылает шифротекст *c* по открытому каналу пользователю *B*. Последний, зная секрет k_B, т.е. умея инвертировать E_B, определяет *m* по полученному *c*, вычисляя $m = D_B(c)$. Противник, не зная ключа k_B, в силу второго свойства односторонней функции с секретом не сможет прочитать секретную информацию. В отличие от криптосистемы с секретным ключом, если абонент *A*, зашифровав некоторое сообщение *m* для абонента *B*, сохранит шифротекст *c*, но потеряет *m* или забудет его содержание, он не будет иметь никаких преимуществ перед противником в раскрытии исходного текста *m* по известному *c*.

Атаки на криптосистему с открытым ключом аналогичны атакам на криптосистемы с секретным ключом, однако следует помнить, что в первом случае противник всегда знает открытый ключ, а значит, всегда возможна атака на основе выбранного открытого текста. Кроме того, возможна так называемая атака с проверкой текста (verifiable text attack): если число возможных открытых текстов невелико, противник, зная открытый ключ, может заранее заготовить соответствующие им шифротексты, а затем, сравнивая с ними перехваченные шифровки, получать соответствующие последним открытые тексты. Исключить такой вид атак на криптосистему позволяет *вероятностное шифрование*.

Шифрование с открытым ключом может использоваться в тех из рассмотренных в гл. 2 режимах, которые для зашифрования и расшифрования используют разные функции соответ-

ственно E_k и D_k (например, ECB и CBC, при этом, очевидно, что второй вариант предпочтительней).

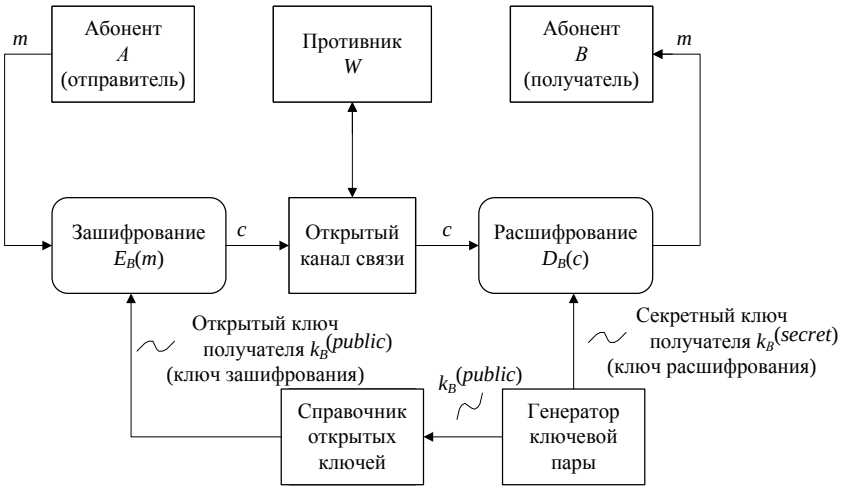


Рис. 5.1. Модель криптосистемы с открытым ключом

Наиболее известные системы с открытым ключом:

рюкзачная криптосистема (Knapsack Cryptosystem);

криптосистема RSA;

криптосистема Эль-Гамаля (ElGamal Cryptosystem);

криптосистема, основанная на свойствах эллиптических кривых.(Elliptic Curve Cryptosystem (ECCS)).

Криптосистема Т. Эль-Гамаля, основанная на сложности решения задачи дискретного логарифмирования, послужила основой для создания старых российского и американского стандартов электронной подписи. Остальные криптосистемы будут рассмотрены ниже.

5.3. Открытое распределение ключей

Помимо принципа построения асимметричных криптосистем Диффи и Хеллман предложили *протокол выработки общего секретного ключа*, т.е. процедуру взаимодействия по открыто-

му каналу связи удаленных абонентов A и B, обладающую следующими свойствами:

в конце процедуры у A и B появляется общая секретная информация (общий секретный ключ k_{AB}), которой до начала действия протокола не было;

противник, способный перехватывать все сообщения и знающий цель протокола, не может восстановить сформированный секретный ключ.

Этот протокол, на первый взгляд невозможный, основывается на задаче дискретного логарифмирования, рассмотренной выше. Противник, который хочет узнать формируемый секретный ключ, вынужден решать именно эту трудную математическую задачу. Пусть p – большое простое число, ω – примитивный элемент поля $GF(p)$ (основы теории конечных полей даны в гл. 8). Числа p и ω считаются общедоступными. Тогда процедура получения общей секретной информации имеет следующий вид.

Протокол выработки общего секретного ключа Диффи–Хеллмана

1. Абоненты A и B независимо друг от друга вырабатывают два случайных числа соответственно x_A и x_B, которые держат в секрете.
2. Абоненты A и B вычисляют значения $y_A = \omega^{x_A} \bmod p$ и $y_B = \omega^{x_B} \bmod p$.
3. Абоненты A и B обмениваются сообщениями y_A и y_B.
4. A, получив сообщение y_B, вычисляет значение
$$(y_B)^{x_A} \bmod p = \left(\omega^{x_B}\right)^{x_A} \bmod p;$$
5. B, получив сообщение y_A, вычисляет значение
$$(y_A)^{x_B} \bmod p = \left(\omega^{x_A}\right)^{x_B} \bmod p.$$
6. Элемент $GF(p)$, равный $\omega^{x_A x_B} \bmod p$, объявляется общим секретным ключом.

5.4. Электронная цифровая подпись

Диффи и Хеллман предложили использовать односторонюю функцию с секретом для *цифровой подписи* сообщений, которая позволила бы, например, устанавливать истину при возникновении споров относительно авторства пересылаемых важных документов. Традиционная криптосистема с секретным ключом (см. рис. 2.1) не может быть использована для разрешения противоречий, возникающих между не доверяющими друг другу абонентами, так как секретная информация (ключ), используемая для зашифрования сообщений, известна как минимум двум лицам. Поэтому пересылка по линиям связи платежных документов или информации, требующей нотариального заверения, в этом случае чревата большими неприятностями.

Если E_k – односторонняя функция с секретом, то подписанное сообщение можно представить в виде пары

$$(m,\ D_k(m)),$$

где m – подписываемый документ, а D_k – функция, обратная E_k. Из определения односторонней функции с секретом следует, что для такой подписи характерны следующие особенности:

подписать сообщение m может только обладатель секрета k, т.е. подделать подпись практически невозможно;

проверить подпись может любой абонент, знающий функцию E_k, так как проверка подписи заключается в проверке равенства

$$m = E_k(D_k(m));$$

при возникновении споров относительно авторства сообщений отказаться от подписи нельзя из-за невозможности ее подделки;

подпись позволяет контролировать *целостность* подписанных документов, а значит, их можно безо всякого ущерба пересылать по открытым каналам связи.

Если необходимо обеспечить секретность, можно применить следующий *протокол цифровой подписи*. Пусть у каждого из двух абонентов *A* и *B* имеется пара взаимно-обратных функций –

открытая функция зашифрования E и секретная функция расшифрования D. Если абонент A хочет послать подписанное сообщение абоненту B, то он, используя сначала свой секретный ключ, а затем открытый ключ получателя, вычисляет

$$c = E_B\left(D_A(m)\right)$$

и передает шифротекст c абоненту B. Получатель сообщения, используя сначала свой секретный ключ, а затем открытый ключ абонента A, восстанавливает исходный текст, вычисляя

$$m = D_B\left(E_A(c)\right).$$

Теперь при возникновении спорной ситуации, когда A отказывается от посланного сообщения, он обязан предъявить арбитру свой секретный ключ. Арбитру необходимо лишь проверить равенство

$$D_A(m) = D_B(c).$$

В случае его выполнения он убеждается, что никто, кроме A, отправить сообщение не мог, и принимает решение в пользу B. Приведенный протокол заставляет быть честным и абонента B. Для того чтобы исказить принятое сообщение m на m', ему необходимо изменить и подпись $s = D_A(m)$ на $s' = D_A(m')$. Но для этой операции необходим секретный ключ абонента A, которого у B нет.

Приведенная последовательность применения функций D_A и E_B единственна возможна. Если абонент A попытается послать B «подписанное» сообщение вида

$$c = D_A\left(E_B(m)\right),$$

противник W, перехватив это сообщение, используя открытый ключ абонента A, сможет вычислить $E_A(c) = E_B(m)$, а затем, используя свой закрытый ключ, послать якобы подписанное им сообщение

$$c' = D_W\left(E_B(m)\right).$$

Тогда B, вычислив

$$D_B\left(E_W(c')\right) = m,$$

был бы убежден, что m подписано именно W. Таким образом, в рассматриваемой ситуации противник всегда может подписывать сообщения, содержимое которых он сам прочитать не в состоянии.

Общий недостаток всех криптосистем с открытым ключом – низкое быстродействие, поэтому на практике применяются модифицированные схемы цифровой подписи, в значительной степени решающие данную проблему. Эти схемы будут рассмотрены далее.

5.5. Криптосистема, основанная на задаче об укладке рюкзака

Создание данной системы шифрования и ее модификаций чаще всего связывают с именами Р. Меркля и М. Хеллмана. Прототипом трудной математической задачи, лежащей в ее основе, является следующая ситуация.

Имеются рюкзак и множество предметов, которые хотелось бы захватить с собой, отправляясь в поход. Требуется найти подмножество предметов, которые заполнят рюкзак до отказа. Рассмотрим пример простой числовой задачи такого типа.

Простая задача об укладке рюкзака

Представить число 55 *в виде суммы некоторых из чисел* {3, 8, 12, 2, 32, 59}.

Ответ: 55 = 3 + 8 + 12 + 32.

Итак, число 55 равно сумме первого, второго, третьего и пятого чисел из представленного списка. Данный факт можно записать следующим образом: $55 \leftrightarrow (111010)$. При этом можно решить обратную задачу: если известна последовательность (111010), можно восстановить число 55, суммируя первое, второе, третье и пятое числа из заданного списка. Таким образом, на основе списка чисел можно построить примитивную систему шифрования, как показано на рис. 5.2.

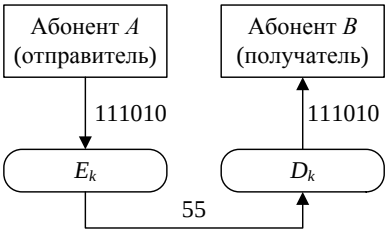


Рис. 5.2. Примитивная система шифрования, в которой отправителю и получателю известен список чисел {3, 8, 12, 2, 32, 59}

Еще более простая задача об укладке рюкзака получается в том случае, если каждое число в списке больше суммы всех предыдущих, стоящих справа, например, являются степенями двойки.

Очень простая задача об укладке рюкзака

Представить число 26 *в виде суммы различных чисел из списка* {32, 16, 8, 4, 2, 1}.

Ответ: 26 = 16 + 8 + 2 или $26 \leftrightarrow (011010)$.

По сути, решение задачи сводится к переводу заданного десятичного числа в двоичную систему счисления, обратная же задача суть перевод заданного двоичного числа в десятичную систему счисления.

Теперь можно сформулировать основные принципы получения односторонней функции с секретом и построения на ее основе криптосистемы с открытым ключом на основе задачи об укладке рюкзака:

1) *составляется трудная задача T, не решаемая за полиномиальное время*;
2) *из T выделяется легкая подзадача* T_{easy}, *имеющая полиномиальный или даже более простой алгоритм решения*;
3) *путем «взбивания» легкая задача* T_{easy} *превращается в труднорешаемую задачу* $T_{shuffle}$, *не имеющую никакого сходства с* T_{easy};

4) *на основе $T_{shuffle}$ определяется открытая функция зашифрования; процедура получения T_{easy} из $T_{shuffle}$ держится в секрете*;
5) *криптосистема конструируется таким образом, чтобы для противника процедура дешифрования заключалась в решении $T_{shuffle}$, имеющей вид трудной задачи T, а законный получатель, знающий секрет, решал бы легкую задачу T_{easy}.*

Предположим, что список $\{a_{n-1}, ..., a_2, a_1\}$состоит из ста 40-значных чисел, т.е. $t = 100$, и c – 42-значное число.

Трудная задача T об укладке рюкзака

Дан список чисел $\{a_{n-1}, ..., a_i, ..., a_1, a_0\}$.

Представить число c в виде суммы некоторых чисел a_i из списка.

Если числа в списке выбраны случайным образом, для ответа на вопрос, какие из них в сумме дают c, необходимо перебрать 2^{100} различных вариантов. Если использовать этот список в системе шифрования, аналогичной показанной на рис. 5.2, она была бы надежной, но законный получатель не смог бы восстановить переданное ему сообщение.

Необходима задача об упаковке рюкзака с секретом, которую мог бы легко решить законный получатель информации, а для противника она бы выглядела по-прежнему трудноразрешимой. Сначала составим легкую задачу T_{easy} об укладке рюкзака, для чего выбираем числа

$$\{a_{n-1}, ..., a_i, ..., a_1, a_0\},$$

содержащие в десятичной записи степени двойки.

Схема соответствующей системы шифрования на основе списка

$a_0 = 020105,$

$a_1 = 220209,$

$a_2 = 260405,$

$a_3 = 120802,$

$a_4 = 031608$

показана на рис. 5.3. Второй и третий разряды каждого числа a_i из списка содержит степень двойки 2^i, первые разряды всех чисел в данном примере содержат нули. В общем случае нулевые разряды должны стоять не только правее, но и левее степеней двойки, чтобы после сложения любого количества чисел из списка в результате переносов не исказилась сумма этих степеней.

«Взбивание» задачи T_{easy} и получение $T_{shuffle}$ можно выполнить с помощью операции модульного умножения. Выберем два больших взаимно простых числа r и s и найдем такое число t, чтобы выполнялось условие $st \equiv 1 (\mathrm{mod} r)$. Уничтожим список

$$\{a_{n-1}, \ldots, a_i, \ldots, a_1, a_0\}$$

и число s, предварительно получив новый список

$$\{b_{n-1}, \ldots, b_i, \ldots, b_1, b_0\},$$

каждый элемент которого суть результат умножения соответствующего число из первоначального списка на s по модулю r:

$$b_i = sa_i (\mathrm{mod} r),\ i = \overline{0, (n-1)}.$$

Числа из нового списка будут казаться случайными числами из интервала $[0; (r-1)]$, и все, кто не знает принцип их получения, примут соответствующую задачу $T_{shuffle}$ за трудную задачу об укладке рюкзака. Именно она будет применяться при зашифровании информации, поэтому список

$$\{b_{n-1}, \ldots, b_i, \ldots, b_1, b_0\},$$

сообщается всем заинтересованным лицам. Числа t и r держатся в секрете. Таким образом, получим следующую систему шифрования.

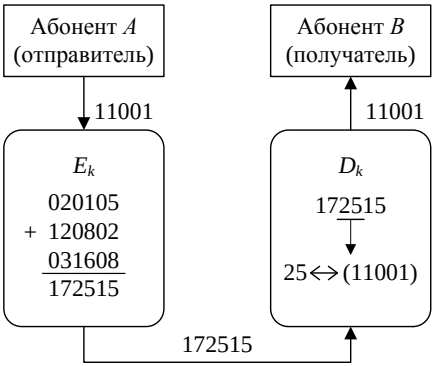


Рис. 5.3. Примитивная система шифрования на основе легкой задачи об укладке рюкзака

Криптосистема на основе задачи об укладке рюкзака

Открытый ключ зашифрования: список

$$\{b_{n-1},\ ...,\ b_i,\ ...,\ b_1,\ b_0\}.$$

Закрытый ключ расшифрования: пара чисел $(t;\ r)$.

Алгоритм зашифрования двоичного сообщения

$$m=(m_{t-1},\ ...,\ m_i,\ ...,\ m_1,\ m_0):$$

$$E_k(m)=\sum_{i=0}^{n-1} b_i m_i = c.$$

Алгоритм расшифрования закрытого сообщения c:

1) составляем произведение

$$tc=\sum_{i=0}^{n-1} tb_i m_i (\mathrm{mod} r)=\sum_{i=0}^{n-1} a_i m_i (\mathrm{mod} r);$$

2) решая легкую задачу об укладке рюкзака, находим

$$m=(m_{n-1},\ ...,\ m_i,\ ...,\ m_1,\ m_0).$$

Пример шифрования по приведенной схеме показан на рис. 5.4.

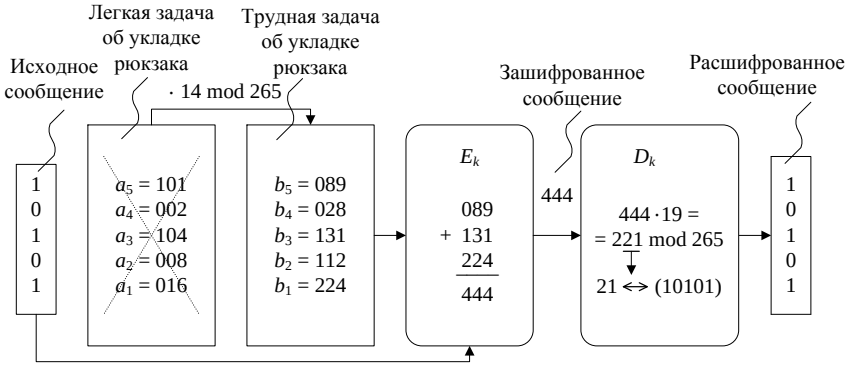


Рис. 5.4. Пример рюкзачной криптосистемы

К сожалению, некоторые варианты этой оригинальной и красивой криптосистемы были раскрыты вскоре после своего создания.

5.6. Криптосистема RSA

Система RSA, название которой происходит от первых букв фамилий авторов – Р. Ривеста, А. Шамира и Л. Адлемана (R. Rivest, A. Shamir, L. Adleman), является наиболее распространенной и изученной криптосистемой с открытым ключом. Она основана на использовании того факта, что легко перемножить два больших простых числа, в то же время крайне трудно разложить на множители их произведение. В результате произведение может быть использовано в качестве элемента открытого ключа зашифрования. Исходные простые числа необходимы для расшифрования, при этом задача их восстановления до сих пор сопротивляется всем атакам криптоаналитиков. Таким образом, имеются хорошие предпосылки для построения односторонней функции с секретом.

Рассмотрим некоторые математические факты, на которых основан алгоритм. Согласно малой теореме Ферма, если p – простое число, то для любого x, взаимно простого с p, справедливо

$$x^{p-1} \equiv 1 (\mathrm{mod}\, p);$$

для любого x, справедливо

$$x^p \equiv x(\mathrm{mod}\, p).$$

Функция $\varphi(n)$ натурального аргумента n, равная количеству положительных целых чисел, меньших n и взаимно простых с n, называется функцией Эйлера. Для нее справедливы следующие соотношения:

$$\varphi(1)=1;$$
$$\varphi(p^r)=p^{r-1}(p-1);$$
$$\varphi(ab)=\varphi(a)\,\varphi(b),$$

где p – простое, r, a и b – натуральные, $(a,b)=1$. Эти свойства позволяют легко вычислять $\varphi(n)$, когда известно разложение n на простые множители. Если натуральное число e удовлетворяет условию $(e,\varphi(n))=1$, то существует единственное натуральное число $d<\varphi(n)$, для которого справедливо

$$de\equiv 1(\mathrm{mod}\varphi(n)).$$

Пусть $n=pq$, где p и q – два больших различных простых числа, и $(x,\varphi(n))=1$. Тогда согласно теореме Эйлера для любого натурального x выполняется сравнение

$$x^{ed} \equiv x(\mathrm{mod}n)$$

и, следовательно,

$$x^{ed}(\mathrm{mod}n)=x$$

при условии $x<n$.

Итак, выберем два больших различных простых числа p и q и число e, взаимно-простое с числом $(p-1)(q-1)$. Вычислим $n=pq$ и найдем такое d, что

$$de\equiv 1(\mathrm{mod}(p-1)(q-1)).$$

Тогда рассматриваемая криптосистема выглядит следующим образом.

Криптосистема RSA

Открытый ключ зашифрования: числа n и e.

Закрытый ключ расшифрования: числа p, q и d.

Алгоритм зашифрования сообщения m:

$$E_k(m) = m^e (\mathrm{mod} n) = c.$$

Алгоритм расшифрования закрытого сообщения c:

$$D_k(c) = c^d (\mathrm{mod} n) = m.$$

Примечание. Необходимым условием является уничтожение чисел p и q после вычисления ключей e и d.

Единственный способ найти алгоритм расшифрования D_k при известных e и n состоит в том, чтобы разложить на простые множители число n, найти p и q, а следовательно, и d. Правильный выбор разрядности чисел p и q (первоначально авторы RSA предлагали в качестве p и q использовать не менее чем 40-разрядные десятичные числа) делает задачу факторизации n практически неосуществимой.

Пример шифрования по приведенной схеме показан на рис. 5.5.

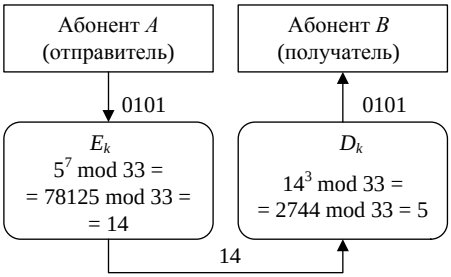


Рис. 5.5. Пример шифрования по схеме RSA

Авторы RSA при описании принципов функционирования своей системы выбрали в качестве исходного текста фразу

«ITS ALL GREEC TO ME»

(«Для меня все это совершенно непонятно»).

Для того чтобы преобразовать этот текст в одно большое число, закодировали пробел между словами 0, букву *A* – 1, бук-

ву *B* – 2, букву *C* – 3, ... , букву *Z* – 26. На представление каждого символа выделили пять двоичных разрядов. В результате приведенной фразе стало соответствовать число

$$m = 09201900011212000718050511002015001305.$$

Для шифрования авторы выбрали $e = 9007$ и $n =$

$$114381625757888867669235779976146612010218296721242362562651842935706935245733897830597123563958705058989075147599290026879543541.$$

После зашифрования получили число

$$c = m^e \pmod{n} =$$

$$199935131497805100452317122740260647423204017058391463103703717406259716089489275043992096267258267501289355446135382376974802 6.$$

Число *n* было произведением 64-значного и 65-значного простых чисел *p* и *q*, выбранных случайным образом.

Если исходный текст слишком велик, чтобы с ним можно было обращаться как с одним числом, то его следует разбить на блоки и каждый блок рассматривать как отдельное число, при этом необходимо использовать шифрование со сцеплением блоков (режим CBC).

При разработке криптосистемы сначала выбираются два различных простых числа *p* и *q*. Для этого произвольно выбирается нечетное число *r* подходящего размера (например, сторазрядное) и проверяется на простоту. В случае, если тест дает отрицательный результат, проверяется число *r* + 2 и т.д. Существует приблизительно $\frac{10^{100}}{\ln 10^{100}} - \frac{10^{99}}{\ln 10^{99}}$ 100-разрядных простых чисел. Учитывая, что всего существует $\left(10^{100} - 10^{99}\right)/2$ 100-разрядных нечетных чисел, вероятность успеха одного конкретного теста равна приблизительно 0,00868. После того как *p* и *q* выбраны, кандидаты на *e* проверяются с помощью алгоритма Евклида. Когда *e* удовлетворяет условию $(e, \varphi(n)) = 1$, цепочка равенств, получаемых в результате применения алгоритма Евклида, дает нам и значение *d*. Процедура шифрования суть модульное возведение в степень, данную операцию следует осуществлять методом

последовательного возведения в квадрат, т.е. после каждого возведения в квадрат результат берется по модулю r. При этом никогда не возникают числа большие n^2 [3].

5.7. Криптосистема Эль-Гамаля

Криптосистема Эль-Гамаля – асимметричная криптографическая схема шифрования, предложенная Тахиром Эль-Гамалем в 1984 г. (Taher ElGamal, «A Public-Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms», IEEE Transactions on Information Theory, v. IT-31, n. 4, 1985, pp. 469–472).

Криптосистема может быть определена над любой циклической группой G. Защищенность схемы основана на сложности решения задачи дискретного логарифмирования.

Алгоритм генерации ключей:

1) породить мультипликативную циклическую группу G степени q с образующей g:
   выбрать простое число q – степень криптосистемы;
   сформировать случайное g, $g < q$;
2) выбрать случайное x, $x < q$;
3) вычислить $h = g^x \bmod q$.

Открытый ключ = (h, g, q).

Закрытый ключ = x.

Алгоритм зашифрования:

1) подготовить шифруемое сообщение m, $m < q$;
2) выбрать случайное y, $y < q$;
3) вычислить $c_1 = g^x \bmod q$, $c_2 = m\, h^y \bmod q$.

Криптограмма – (c_1, c_2).

Алгоритм расшифрования:

1) вычислить $m = \dfrac{c_2}{c_1^x} \bmod q$.

Расшифрование будет верным, поскольку

$$m = \frac{c_2}{c_1^x} = \frac{mh^y}{g^{xy}} = \frac{mg^{xy}}{g^{xy}} = m.$$

Криптостойкость системы Эль-Гамаля зависит от свойств группы G и особенностей конкретной реализации. Схема уязвима к атакам с подобранным шифротекстом. К примеру, имея криптограмму (c_1, c_2) некоторого (возможно, неизвестного) сообщения m, легко создать криптограмму $(c_1, 2c_2)$ сообщения $2m$.

Шифрование по схеме Эль-Гамаля является вероятностным, другими словами, каждому исходному сообщению может соответствовать множество достоверных шифротекстов за счет произвольности выбора y.

5.8. Гибридные криптосистемы

Несмотря на все преимущества криптосистем с открытым ключом, ни одна из известных на сегодняшний день их реализаций не может конкурировать по быстродействию с криптосистемами с секретным ключом. Так, например, быстродействие системы RSA в тысячи раз ниже быстродействия DES. В результате при шифровании длинных информационных последовательностей может случиться так, что применение асимметричного алгоритма недопустимо снижает скорость информационного обмена, а применение симметричного невозможно из-за отсутствия общего секретного ключа у участников этого обмена или по каким-то другим причинам. Выходом из этой ситуации является использование гибридной криптосистемы (рис. 5.6, 5.7).

В схеме, показанной на рис. 5.6, на начальном этапе участники информационного обмена (абоненты A и B), используя протокол выработки общего секретного ключа, формируют общую секретную информацию (сеансовый ключ k_{AB}). На следующем этапе для обмена зашифрованными сообщения используется криптосистема с секретным ключом.

Схема, показанная на рис. 5.7, предполагает наличие у каждого участника информационного обмена двух ключей: открытого $k^{(public)}$ и закрытого $k^{(secret)}$. Рассмотрим процесс пересылки некоего документа m. Отправитель (абонент A) вырабатывает секретный ключ – случайное число, используемое только один раз, и поэтому называемое одноразовым или сеансовым ключом. Се-

ансовый ключ используется для зашифрования документа m при помощи симметричного криптоалгоритма, после этого он зашифровывается на открытом ключе получателя (абонент B) и присоединяется к ранее зашифрованному документу. Сформированное таким образом сообщение отсылается получателю. Последний, получив сообщение, повторяет те же процедуры, но в обратном порядке. С помощью своего секретного ключа получатель восстанавливает сеансовый ключ, а затем с его помощью расшифровывает и сам документ.

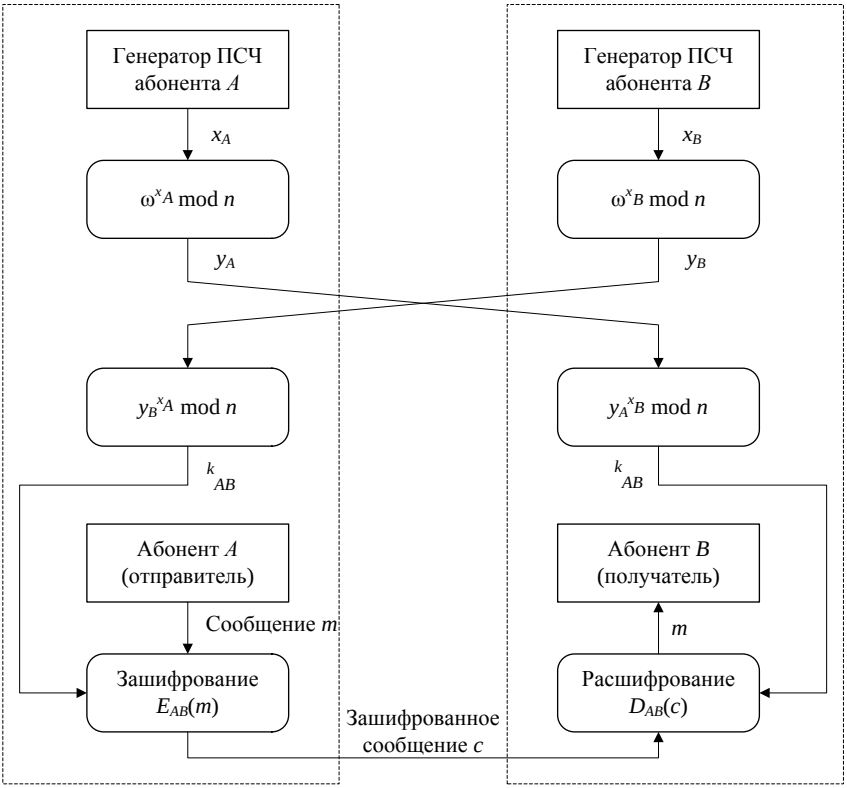


Рис. 5.6. Первый вариант схемы гибридного шифрования

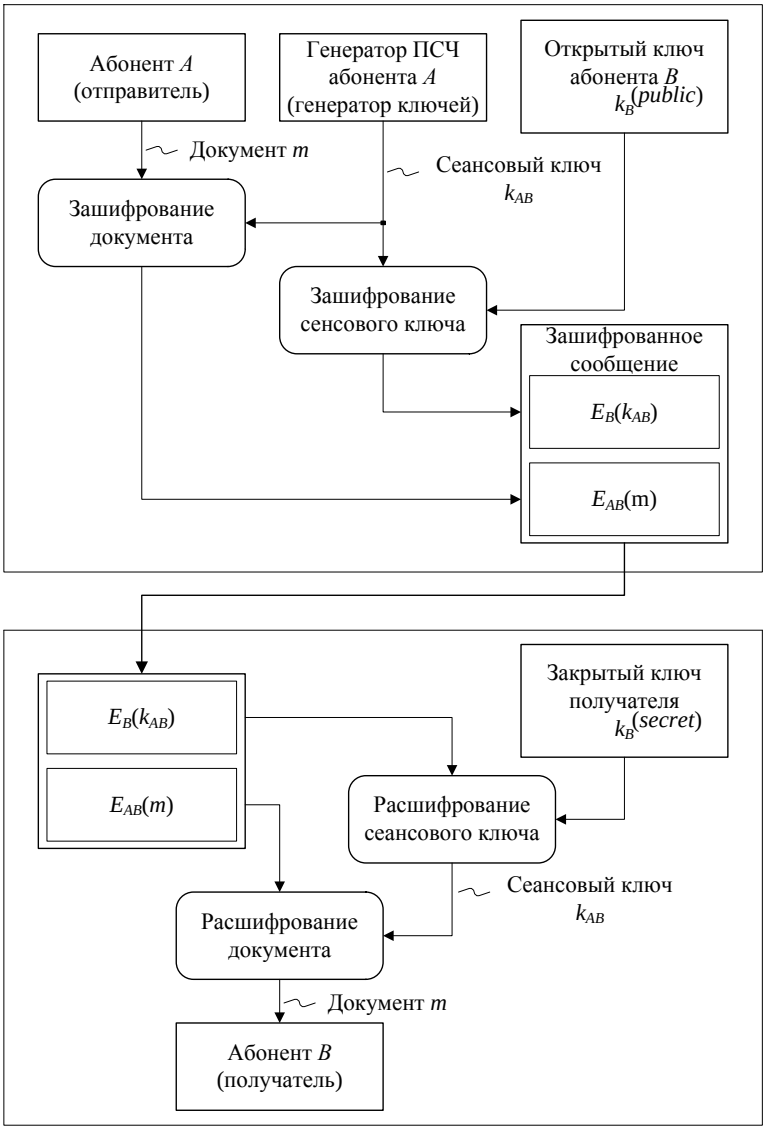


Рис. 5.7. Второй вариант схемы гибридного шифрования

5.9. Генераторы ПСЧ на основе односторонних функций с секретом

Криптографически стойкие генераторы ПСЧ могут быть построены на основе односторонних функций с секретом. М. Блюм и С. Микали для обозначения основного требования к криптостойкому генератору ПСЧ ввели понятие *непредсказуемость* (unpredictable) или *непредикативность влево* – криптоаналитик, знающий логику работы такого генератора, имеющий возможность анализировать его выходную последовательность, но не знающий используемого инициализирующего вектора (синхропосылки), для определения первого выработанного бита не может предложить лучшего способа, чем подбрасывание жребия.

Справедливы следующие утверждения [3, 4]:

непредикативный влево генератор является криптографически сильным;

криптографически сильные генераторы существуют тогда и только тогда, когда существуют односторонние функции.

Рассмотрим, например, так называемый *BBS*-генератор, получивший свое название в честь авторов Э. Блюм, М. Блюма и М. Шуба, основанный на *криптосистеме Блюма*.

Пусть p и q – два больших простых числа примерно одинакового размера, причем

$$p \equiv \bmod 4, q \equiv \bmod 4.$$

Тогда число $n = pq$ называется *целым числом Блюма*. Пусть $\tilde{Z}_n^*$ – множество целых положительных чисел, меньших n, которые не делятся ни на p, ни на q. Пусть QR_n – подмножество $\tilde{Z}_n^*$, состоящее из квадратичных вычетов по модулю n. Число элементов множества $\tilde{Z}_n^*$ равно $(p - 1)(q - 1)$, причем в точности четвертую их часть составляют элементы подмножества QR_n. Каждый элемент QR_n имеет ровно четыре различных квадратных корня в $\tilde{Z}_n^*$, из них лишь один, называемый *примитивным*, принадлежит QR_n.

Пример 5.1. Если $p = 19$, $q = 23$; имеем $n = 437$. Тогда

$$133 = 19 \cdot 7 \notin \tilde{Z}^*_{437}, 135 \in \tilde{Z}^*_{437}, 139 \in \tilde{Z}^*_{437};$$

кроме того, учитывая, что не существует такого целого числа a, что

$$a^2 \equiv 35 \bmod 437,$$

а

$$24^2 \equiv 39 \bmod 437,$$

можно записать

$$135 \notin QR_{437}, 139 \in QR_{437}.$$

Квадратными корнями из 139 по модулю 437 являются числа 24, 185, 252 и 413, при этом 24 – примитивный квадратный корень, так как

$$47^2 = 24 \bmod 437.$$

Задача определения примитивных квадратных корней по модулю числа n вычислительно эквивалентна задаче разложения этого числа на множители. Таким образом, получаем кандидата на одностороннюю функцию с секретом, поскольку функция

$$f(x) = x^2 \bmod n$$

эффективно вычисляется, а произвести обратное преобразование может лишь тот, кто знает секрет – разложение n на множители.

Пусть n – целое число Блюма.

Генератор Э. Блюм–М. Блюма–Шуба

1. Выберем в качестве инициализирующего вектора случайное число $x_0 \in QR_n$. Для этого возьмем такое случайное число x, что $(x, n) = 1$, и вычислим

   $$x_0 = x^2 \bmod n.$$

2. Искомой последовательностью бит длиной m будет последовательность

   $$BBS_{n,\,m}(x_0) = b_0 b_1 b_2 \ldots b_i \ldots b_{m-1},\ i = \overline{0, (m-1)},$$

   где b_i – младший бит числа x_i,

$$x_{i+1} = x_i^{\,2} \bmod n.$$

Важным достоинством этого генератора является то, что при знании разложения n на множители он допускает *прямое определение* отдельных бит, которые в нем вырабатываются. Имеем

$$x_i = x_0^{2^i} \bmod n.$$

По теореме Эйлера

$$x^{(p-1)(q-1)} \equiv 1 \bmod n,$$

а значит,

$$x_i = x_0^{\,2^i \bmod (p-1)(q-1)} \bmod n,$$

т.е. с помощью двух операций модульного возведения в степень, которые эффективно вычисляются, любое число x_i может быть найдено исходя лишь из начального вектора x_0 и индекса i.

Аналогичным образом можно построить генератор ПСЧ на основе другой односторонней функции с секретом, например лежащей в основе криптосистемы RSA, т.е. на основе функции

$$f(x) = x^e \bmod n.$$

Генератор RSA

1. Выберем в качестве инициализирующего вектора случайное число $x_0 \in Z_n^*$.
2. Искомой последовательностью бит длиной m будет являться последовательность

   $$RSA_{n,\,m}\,(x_0) = b_0\, b_1\, b_2\, \ldots b_i\, \ldots b_{m-1},\ i = \overline{0, (m-1)},$$

   где b_i – младший бит числа x_i,

   $$x_{i+1} = x_i^{\,e} \bmod n.$$

Ш. Гольдвассер и С. Микали предложили схему шифрования, основанную на использовании BBS-генератора в качестве источника ключевой последовательности. *Данная схема не обеспечивает секретности по отношению к атаке на основе выбранного шифротекста*.

Пусть исходное сообщение M суть m-разрядная битовая последовательность, x_0 – случайный квадратичный вычет по модулю n. Функция зашифрования по схеме Гольдвассер–Микали имеет вид

$$c = \left(x_m,\ M \oplus BBS_{n,m}(x_0)\right),$$

при этом x_m включается в шифротекст для того, чтобы законный получатель мог его расшифровать. Простейший алгоритм расшифрования в данном случае может заключаться в восстановлении ПСП в обратном направлении в соответствии с рекуррентным уравнением

$$x_i = \sqrt{x_{i+1}} \bmod n,$$

а затем получении исходного текста по формуле

$$M = c \oplus BBS_{n,m}(x_0).$$

Знание множителей p и q позволяет законному получателю сразу из x_m получить x_0, а затем формировать ПСП точно так же, как это делал отправитель.

Алгоритм вычисления x_0 из x_m

1. С помощью обобщенного алгоритма Евклида вычисляем такие целые числа a и b, что
$$ap + bq = 1.$$
2. Вычисляем
$$\alpha = \left(\frac{p+1}{4}\right)^m \bmod (p-1);$$
$$\beta = \left(\frac{q+1}{4}\right)^m \bmod (q-1);$$
$$u = (x_m \bmod p)^{\alpha} \bmod p;$$
$$v = (x_m \bmod q)^{\beta} \bmod q;$$
$$x_0 = (apv + bqu) \bmod n.$$

Таким образом, эффективность рассмотренной схемы выше эффективности схемы RSA. Более того, тщательный анализ BBS-генератора показал, что после каждой операции модуль-

ного возведения в квадрат можно использовать приблизительно $\log_2 L$ младших битов числа x_i, где L – разрядность модуля n [3].

Контрольные вопросы

1. Кто инициатор создания двухключевой криптосистемы: отправитель или получатель?
2. Назовите два принципиальных недостатка двухключевых криптосистем.
3. Что такое односторонняя функция? Приведите пример.
4. Что такое односторонняя функция с секретом? Приведите пример.
5. На чем основана стойкость криптосистемы RSA?
6. Приведите пример создания криптосистемы RSA.
7. На чем основана стойкость ранцевой криптосистемы?
8. На чем основана стойкость криптосистемы Эль-Гамаля?
9. Приведите пример создания криптосистемы Эль-Гамаля.

ГЛАВА 6. КРИПТОГРАФИЧЕСКИЕ ПРОТОКОЛЫ

Успешное решение задачи открытого распределения ключей, создание практических схем цифровой подписи способствовали возникновению нового направления криптографии – теории *криптографических протоколов*. Для современных информационных систем характерен перевод всего документооборота в электронную форму. Возникающие при этом проблемы чаще всего могут быть решены только с использованием возможностей криптографии. Типичный пример: взаимодействуют два удаленных абонента *A* (клиент банка) и *B* (банк). Абонент *A* хочет доказать абоненту *B*, что он тот, за кого себя выдает, а не злоумышленник. Для решения этой задачи требуется *протокол аутентификации абонента*. В каждом конкретном случае для решения некой задачи, связанной с обеспечением безопасности пересылаемых электронных документов, требуется использование соответствующих криптографических протоколов, которые за последние годы превратились в основной объект исследований криптографии. Материал данной главы в значительной степени основывается на работе [4].

6.1. Основные понятия

Объект изучения теории криптографических протоколов – удаленные абоненты, взаимодействующие по открытым каналам связи. Цель взаимодействия – решение какой-либо практической задачи. Модель предусматривает наличие противника, преследующего собственные цели. Противник может выдавать себя за законного субъекта взаимодействия, вмешиваться в информационный обмен между абонентами и т.п. Участники протокола в общем случае не доверяют друг другу, иначе говоря, некоторые протоколы должны быть рассчитаны на ситуацию, когда противником может оказаться даже один из абонентов или несколько абонентов, вступивших в сговор. В настоящее время известно несколько десятков различных типов протоколов. Все эти типы можно разделить на две группы:

1) *прикладные протоколы*, решающие конкретную задачу, которая может возникнуть на практике;
2) *примитивные протоколы*, используемые в качестве своеобразных строительных блоков при создании прикладных протоколов.

Протокол чаще всего интерактивен, т.е. предусматривает многораундовый обмен сообщениями между участниками, и включает в себя:

распределенный алгоритм, т.е. описание характера и последовательности действий каждого из участников;
спецификацию форматов пересылаемых сообщений;
спецификацию синхронизации действий участников;
описание действий при возникновении сбоев [4].

6.2. Доказательства с нулевым разглашением

Существенное влияние на разработку многих криптографических протоколов оказали исследования двух математических моделей – *интерактивной системы доказательств* (Interactive Proof System) и *доказательств с нулевым разглашением знаний* (Zero-Knowledge Proofs).

Интерактивная система доказательств суть протокол (P, V, S) взаимодействия двух субъектов: доказывающего (претендента) P и проверяющего (верификатора) V. Абонент P хочет доказать V, что утверждение S истинно. При этом считается, что абонент V самостоятельно проверить утверждение S не в состоянии, что абонент V не может быть противником, а абонент P может быть противником, пытающимся доказать истинность ложного утверждения S. Протокол, состоящий из некоторого числа раундов обмена сообщениями между P и V, должен удовлетворять двум условиям:

1) *полнота* – если S действительно истинно, то доказывающий убедит проверяющего признать это;
2) *корректность* – если S ложно, то доказывающий не сможет убедить проверяющего в обратном.

Если предположить, что V может быть противником, который хочет получить информацию об утверждении S, необходим протокол (P, V, S), называемый доказательством с нулевым разглашением, удовлетворяющий, помимо перечисленных, еще и следующему условию:

3) *нулевое разглашение* – в результате работы протокола абонент V не увеличит своих знаний об утверждении S.

Иными словами, в результате реализации протокола абонент P сможет доказать абоненту V, что он владеет некоторой секретной информацией, не разглашая ее сути.

Упрощенно процедуру доказательства с нулевым разглашением можно представить следующим образом. Проверяющий задает серию случайных вопросов, каждый из которых допускает ответ «да» или «нет». После первого вопроса проверяющий убеждается в том, что доказывающий заблуждается с вероятностью 1/2. После второго вопроса проверяющий убеждается в том, что доказывающий заблуждается с вероятностью 1/4 и т.д. (после каждого вопроса знаменатель удваивается). После 100 вопросов вероятность того, что доказывающий заблуждается или не располагает доказательством (не владеет секретной информацией), становится настолько близкой к нулю, что даже у самого «недоверчивого» проверяющего не должно остаться сомнений в справедливости доказываемого утверждения. После 300 вопросов знаменатель достигает величины, которая превосходит число атомов во Вселенной.

Роль доказательств с нулевым разглашением особенно велика при реализации *протоколов аутентификации*. Пусть, например, P – алгоритм, реализованный в интеллектуальной карточке клиента (абонент A) банка; V – программа, выполняемая компьютером банка (абонент B). Перед выполнением любой операции банк должен убедиться в подлинности карточки и идентифицировать ее владельца. Если для этой цели использовать протокол (P, V, S), свойство полноты позволит карточке доказать свою *аутентичность*; свойство корректности защитит интересы банка от злоумышленника, который попытается воспользоваться

фальшивой карточкой; свойство нулевого разглашения защитит клиента от злоумышленника, который попытается пройти аутентификацию под именем абонента *A*, воспользовавшись информацией, перехваченной во время предыдущих раундов аутентификационного обмена. Аналогичных результатов можно добиться, используя протоколы доказательств с нулевым разглашением для создания не поддающихся подделке удостоверений личности, как для военных так и гражданских целей [4].

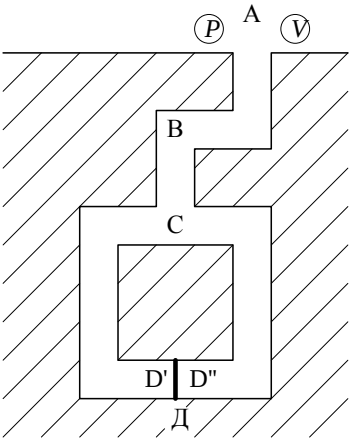
6.2.1. Пещера нулевого знания

На рис. 6.1 показана физическая реализация протокола доказательства с нулевым разглашением с помощью так называемой «пещеры нулевого знания» [32]. В пещере имеется дверь Д, от которой у участника протокола *P* (доказывающего) есть секретный ключ. *P* хочет доказать другому участнику протокола *V* (проверяющему) свою способность проходить через дверь (наличие секретного ключа), не демонстрируя сам факт прохождения через дверь. Показаны две итерации протокола.

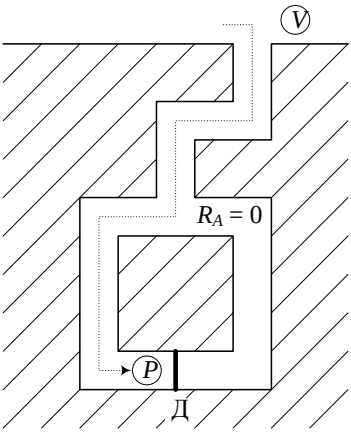
Работу генератора ПСЧ в этой реализации имитирует процедура подбрасывания монеты, которую на определенных шагах протокола выполняют обе стороны.

6.2.2. Протокол Фиата–Шамира

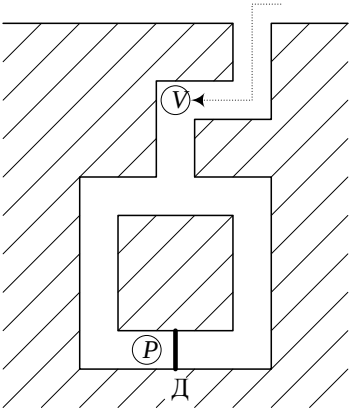
Примером протокола доказательства с нулевым разглашением может являться схема Фиата–Шамира, показанная на рис. 6.2. Доверенный третий участник протокола выбирает два больших простых числа *p* и *q*, затем вычисляет $n = pq$. Величина *n* открыто публикуется. Абонент *A* выбирает случайное число $s \in Z_n$, вычисляет $\nu = s^2 \bmod n$. *A* хранит *s* в качестве своего секретного ключа и объявляет ν своим открытым ключом.



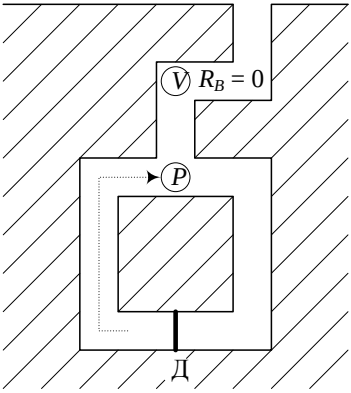
Исходная ситуация:
доказывающий *P* и проверяющий *V*
находятся в точке A



Итерация 1, шаг 1:
доказывающий *P* спускается в пещеру;
дойдя до точки C, подбрасывает монету;
исход $R_A = 0$, поэтому *P* поворачивает
налево и спускается в точку D';
V не видит, куда повернул *P*

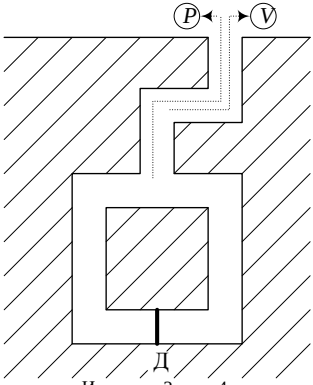


Итерация 1, шаг 2:
V спускается в точку B;
V не видит, где находится *P*

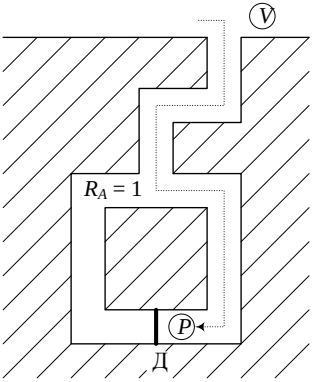


Итерация 1, шаг 3:
V подбрасывает монету; исход $R_B = 0$,
поэтому *V* просит *P* выйти из пещеры
слева, *P* из точки D' возвращается в точку
B, не пройдя через дверь Д

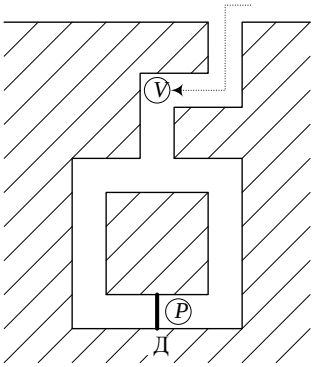
Рис. 6.1. Пещера нулевого знания (начало)



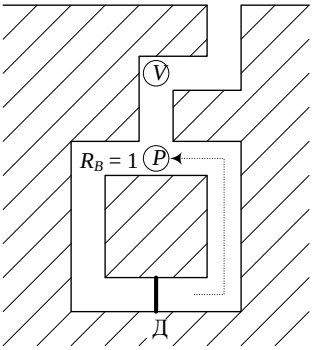
Итерация 2, шаг 4:
P и *V* возвращаются в исходную ситуацию – в точку A



Итерация 3, шаг 1:
доказывающий *P* спускается в пещеру; дойдя до точки C, подбрасывает монету; исход $R_A = 1$, поэтому *P* поворачивает направо и спускается в точку D''; *V* не видит, куда повернул *P*



Итерация 3, шаг 2:
V спускается в точку B;
V не видит, где находится *P*



Итерация 3, шаг 3:
V подбрасывает монету; исход $R_B = 1$, поэтому *V* просит *P* выйти из пещеры справа, *P* из точки D'' возвращается в точку B, не пройдя через дверь Д

Итерация 3, шаг 4:
P и *V* возвращаются в исходную ситуацию - в точку A
Результат:
С вероятностью 7/8 можно утверждать, что *P* владеет секретом прохождения через дверь Д

Рис. 6.1. Пещера нулевого знания (окончание)

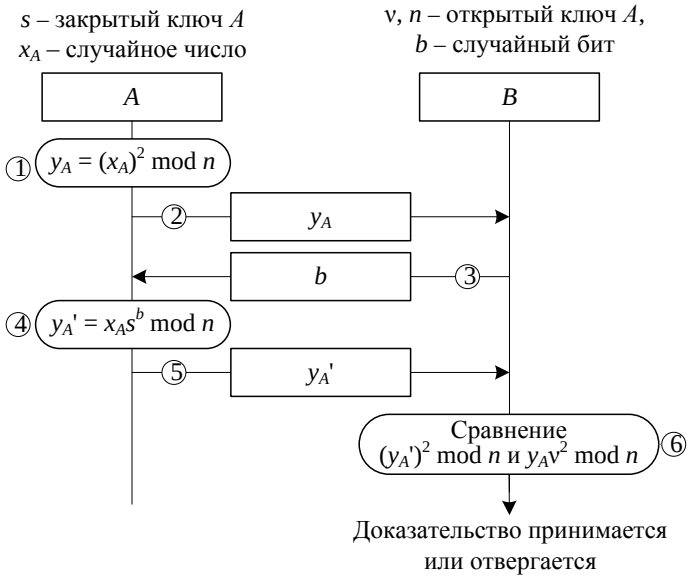


Рис. 6.2. Протокол Фиата–Шамира

Протокол включает шесть шагов:

1) абонент A выбирает случайное число x_A; A вычисляет $y_A = x_A^{\,2} \bmod n$;
2) A посылает y_A абоненту B;
3) абонент B, проверяющий, посылает A в качестве запроса случайный бит $b \in \{0,1\}$;
4) абонент A вычисляет ответ $y_A' = x_A s^b$;
5) A посылает ответ B, чтобы доказать, что он знает свой секретный ключ s;
6) абонент B вычисляет $(y_A')^2$ и $y_A v^b$. Если эти два числа сравнимы по модулю n, то либо абонент A знает число s, либо вычислил значение у каким-либо другим способом. Доказательство этого факта элементарно:

$$(y_A')^2 = (x_A s^b)^2 = (x_A)^2 (s^2)^b = y_A v^b.$$

Эти шесть шагов образуют один раунд. Проверка осуществляется несколько раз с выбираемыми случайно значениями b.

Доказывающая сторона должна успешно пройти проверку в каждом раунде, чтобы быть аутентифицированной. Если в каком-то раунде проверка завершилась отрицательным результатом, процесс аутентификации прерывается.

Если *A* знает *s*, он пройдет проверку во всех раундах. Если *A* не является тем, за кого он себя выдает, т.е. не знает *s*, он может попытаться пройти раундовую проверку, правильно предсказав величину *b*. При этом возможны две ситуации:

1) *A* предполагает, что *b* будет равно 1. *A* вычисляет $y_A = x_A^2 / \nu$ и посылает y_A абоненту *B*.

Если предположение *A* оказывается правильным, он посылает на шаге 3 $y_A' = x_A$ в качестве ответа. Видно, что *A* пройдет проверку, так как $(y_A')^2 = y_A \nu^b$.

Если *A* ошибся, значение y_A' не пройдет проверку и *B* прервет процесс аутентификации.

2) *A* предполагает, что *b* будет равно 0. *A* вычисляет $y_A = (x_A)^2$ и посылает y_A абоненту *B*.

Если предположение *A* оказывается правильным, он посылает на шаге 3 $y_A' = x_A$ в качестве ответа. Видно, что *A* пройдет проверку, так как $(y_A')^2 = y_A \nu^b$.

Если *A* ошибся, значение y_A' не пройдет проверку и *B* прервет процесс аутентификации.

Таким образом, абонент, не знающий числа *s*, успешно пройдет раундовую проверку с вероятностью 1/2. Если процесс повторяется 20 раз, вероятность успеха снижается до $(1/2)^{20} \approx 9{,}5 \times 10^{-7}$.

6.2.3. Протокол Фейга–Фиата–Шамира

Особенностью данного протокола (рис. 6.3) является использование вектора секретных ключей, вектора открытых ключей и вектора запросов. Секретные ключи выбираются случайно, но при этом они должны быть взаимно простые с *n*. Открытые клю-

чи выбираются таким образом, чтобы $v_i = \left(s_i^2\right)^{-1} \bmod n$. b_i на шаге 3 также выбираются случайно.

Протокол основывается на справедливости равенства

$$\left(y_A'\right)^2 v_1^{b_1} v_2^{b_2} \ldots v_m^{b_m} = x_A{}^2 \left(s_1^{b_1}\right)^2 \left(s_2^{b_2}\right)^2 \ldots \left(s_m^{b_m}\right)^2 v_1^{b_1} v_2^{b_2} \ldots v_m^{b_m} =$$
$$= y_A \left(s_1^2\right)^{b_1} \left(v_1^{b_1}\right) \left(s_2^2\right)^{b_2} \left(v_2^{b_2}\right) \ldots \left(s_m^2\right)^{b_m} \left(v_m^{b_m}\right) =$$
$$= y_A \left(s_1^2 v_1\right)^{b_1} \left(s_2^2 v_2\right)^{b_2} \ldots \left(s_m^2 v_m\right)^{b_m} = y_A (1)^{b_1} (1)^{b_2} \ldots (1)^{b_m} = y_A .$$

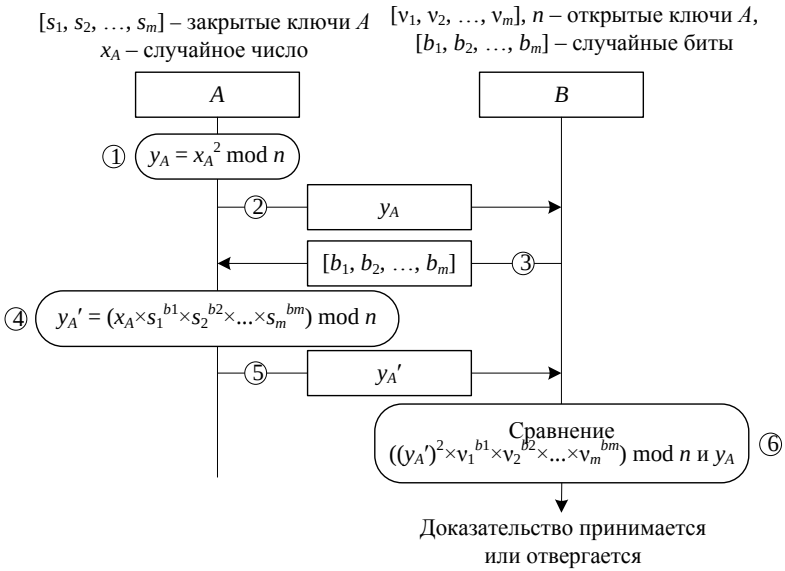


Рис. 6.3. Протокол Фейга–Фиата–Шамира

6.3. Протоколы подбрасывания монеты

Классическим примером задачи, решаемой двумя удаленными абонентами, является бросание жребия с помощью подбрасывания монеты. Это необходимо сделать так, чтобы абонент *A*, подбрасывающий монету, не мог изменить результат после получения догадки от абонента *B*, угадывающего этот результат. Протокол решения данной задачи называют *протоколом подбрасывания монеты*. Этот примитив (по сути, протокол генера-

ции случайного бита) используют в своем составе многие прикладные протоколы.

Примером такого протокола является *схема Блюма–Микали*, предполагающая наличие у двух его участников односторонней функции $F: X \to Y$, удовлетворяющая следующим требованиям:

X – конечное множество целых чисел, содержащее одинаковое количество четных и нечетных чисел;

любые числа $x_1, x_2 \in X$, такие, что $F(x_1) = F(x_2)$, имеют одинаковую четность;

по заданному значению $F(x)$ невозможно определить четность аргумента x.

Протокол подбрасывания монеты (схема Блюма–Микали)

1. Абонент A выбирает случайное число $x_A \in X$ (подбрасывает монету), вычисляет $y_A = F(x_A)$ и посылает y_A абоненту B.
2. Абонент B, получив y_A, пытается угадать четность x_A и посылает свою догадку A.
3. Абонент A, получив догадку от B, сообщает последнему, угадал ли он, посылая ему выбранное число x_A.
4. Абонент B, получив x_A, проверяет, не обманывает ли A, вычисляя значение $F(x_A)$ и сравнивая его с полученным на втором шаге значением y_A.

Суть протокола состоит в том, что абонент A связывает себя с числом x_A, так как он сообщает абоненту B значение $y_A = F(x_A)$, однако абонент B не может самостоятельно вычислить x_A, так как $F(x)$ является односторонней функцией.

Следующий протокол подбрасывания монеты (рис. 6.4) основан на предположении о вычислительной неразрешимости задачи дискретного логарифмирования. Пусть p и q – простые числа, причем q делит $p - 1$. Найдем такое $g \in Z_p$, что

$$g^q = 1 \bmod p,\ g \neq 1.$$

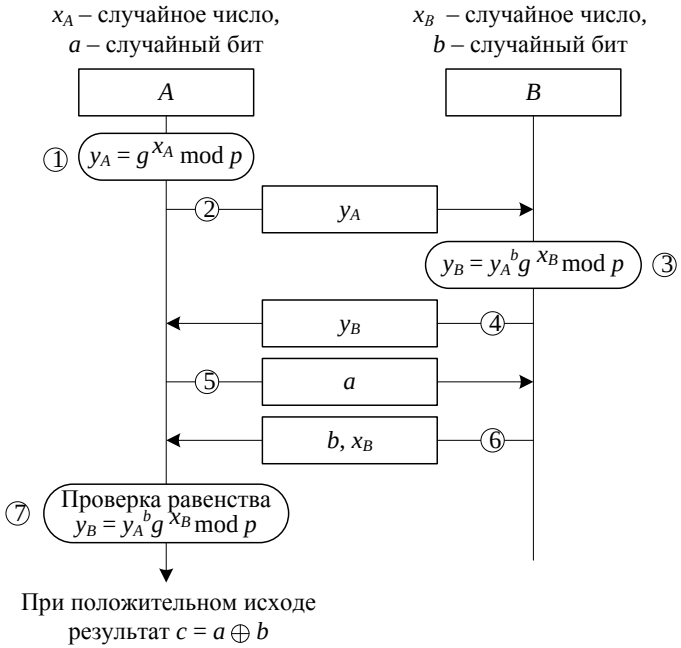


Рис. 6.4. Протокол подбрасывания монеты

Протокол подбрасывания монеты (схема Блюма)

1. Абонент A выбирает случайное число $x_A \in Z_q$, вычисляет $y_A = g^{x_A} \bmod p$.
2. Абонент A посылает y_A абоненту B.
3. Абонент B выбирает случайный бит b и случайное число $x_B \in Z_q$, вычисляет $y_B = y_A^b g^{x_B} \bmod p$.
4. Абонент B посылает y_B абоненту A.
5. Абонент A выбирает случайный бит a и посылает его абоненту B.
6. Абонент B посылает абоненту A бит b и число x_B.
7. Абонент A проверяет равенство $y_B = y_A^b g^{x_B} \bmod p$. При положительном исходе результатом выполнения протокола будет бит $c = a \oplus b$.

Суть протокола объясняет следующая его «физическая» реализация. Абонент B выбирает случайный бит b, записывает его на листе бумаги, запирает этот лист в ящик (y_B суть криптографический аналог этого ящика), оставляя ключ (x_B) у себя, и посылает ящик абоненту A. Получив запертый ящик, абонент A не может добраться до его содержимого. Он выбирает случайный бит a и посылает его абоненту B. В ответ B посылает A ключ от ящика, а затем определяет исход подбрасывания монеты $c = a \oplus b$. Абонент A, получив ключ, отпирает ящик, читает b и точно таким же образом узнает c [4].

Следующий протокол основан на предположении, о вычислительной неразрешимости задачи дискретного логарифмирования.

Протокол подбрасывания монеты

1. Абонент A выбирает случайным образом некоторое число Блюма n (см. гл. 5) число $x_A \in Z_n$, вычисляет $y_A = x_A^2 \bmod n$ и $z_A = y_A^2 \bmod n$, затем сообщает числа n и z_A абоненту B.
2. Абонент B сообщает абоненту A свое предположение о четности или нечетности y_A.
3. Абонент A сообщает абоненту B числа x_A и y_A, а также дает возможность убедиться, что n является числом Блюма.
4. Абонент B проверяет равенства $y_A = x_A^2 \bmod n$ и $z_A = y_A^2 \bmod n$.

Таким образом, абонент A передает абоненту B квадратичный вычет по модулю целого числа Блюма, а B пытается угадать, является ли его примитивный квадратный корень четным или нечетным. Выбор в качестве n целого числа Блюма – гарантия отсутствия у z_A двух квадратных корней различной четности, оба из которых являются квадратичными вычетами. Можно предложить следующую «физическую» реализацию приведенного протокола. A бросает монету в колодец. A видит исход бросания монеты, но не может его изменить. B находится далеко и не может

его увидеть. Однако *A* в конце концов может позволить *B* подойти поближе и заглянуть в колодец [4].

6.4. Протоколы битовых обязательств

Если из приведенной выше схемы Блюма вычленить шаги 1, 2 и 4, получим так называемый *протокол битового обязательства* или *привязки к биту* (bit commitment). Шаги 1 и 2 в этом протоколе называются этапом привязки, а шаг 4 – этапом открытия бита. В данной схеме для значения y_B, в которое «упаковывается» бит *b*, используется термин *блоб* (blob), абонент *A* называется получателем, а абонент *B* – отправителем. Схема привязки к биту классический пример криптографического примитива, который используется в многочисленных прикладных протоколах.

Идеальная конструкция блоба должна обеспечивать одновременное выполнение двух требований:

1) *гарантии безопасности отправителя* – после выполнения этапа привязки получатель не может самостоятельно определить, какой бит упакован в блоб;
2) *гарантии безопасности получателя* – на этапе открытия бита отправитель может открыть блоб либо только как 0, либо как 1.

Иными словами, цель привязки к биту – позволить абоненту *A* взять на себя в качестве обязательства значение некоторого бита информации таким способом, который не позволит абоненту *B* узнать это значение без помощи *A*, при этом сам *A* после привязки к биту не может его изменить.

Описанный протокол привязки к биту гарантирует безусловную безопасность отправителя. В то же время безопасность получателя основывается на недоказанном предположении о вычислительной неразрешимости задачи дискретного логарифмирования. Существует другой тип протоколов привязки к биту, в которых безопасность получателя безусловна, а безопасность отправителя основывается на недоказанном предположении. Учитывая, что такая асимметрия типична для многих типов криптографических протоколов, иногда строятся двойственные протоколы такого рода [4].

6.5. Протоколы разделения секрета

Эффективным методом защиты является *разделение доступа* (не путать с *разграничением доступа*!), разрешающее доступ к секретной информации только при одновременном предъявлении своих полномочий участниками информационного взаимодействия, не доверяющих друг другу. Протоколы или *схемы разделения секрета* (СРС) (Secret Sharing Scheme) позволяют распределить секрет между n участниками протокола таким образом, чтобы заранее заданные разрешенные множества участников могли однозначно восстановить секрет, а неразрешенные – не получали бы никакой информации о возможном значении секрета. Выделенный участник протокола, распределяющий доли (share) секрета, обычно называется *дилером*.

Пусть m – защищаемая информационная последовательность (битовая строка) длиной t. Простейшая схема разделения секрета между тремя абонентами A, B и C имеет следующий вид:

1) дилер D вырабатывает две случайные битовые строки r_A и r_B длиной t и вычисляет $r_C = m \oplus r_A \oplus r_B$;
2) D передает абоненту A информационную последовательность r_A, абоненту B – информационную последовательность r_B и абоненту C – информационную последовательность r_C;
3) чтобы прочитать информацию m, абонентам A, B и C необходимо предъявить свои доли секрета r_A, r_B и r_C и вычислить $m = r_A \oplus r_B \oplus r_C$.

Шаги 1 и 2 – это стадия распределения долей секрета. Шаг 3 – стадия восстановления секрета. Каждая доля секрета сама по себе не имеет никакого смысла, но если их сложить смысл исходной информационной последовательности полностью восстанавливается. При правильной реализации приведенный протокол полностью безопасен, так как для закрытия информа-

ции применяется абсолютно стойкий шифр, описанный в гл. 1, и поэтому никакие вычисления не смогут помочь при попытке определить секрет по одной или двум его частям. Аналогичную схему можно легко реализовать для любого числа участников.

Рассмотрим *схему разделения доступа Шамира*. Пусть n – число участников протокола, $GF(p)$ – конечное поле из p элементов, p – простое, $p > n$. Поставим в соответствие каждому i-му участнику ненулевой элемент поля α_i, $i = \overline{1, n}$, и положим $\alpha_0 = 0$.

Схема разделения доступа Шамира

Стадия распределения долей секрета s_0. Дилер СРС вырабатывает $t - 1$ независимых равномерно распределенных на $GF(p)$ случайных чисел γ_j, $j = \overline{1,(t-1)}$, и посылает каждому i-му участнику соответствующее ему значение $s_i = f(\alpha_i)$ многочлена

$$f(x) = \gamma_{t-1}x^{t-1} + \ldots + \gamma_1 x + \gamma_0, \text{ где } \gamma_0 = s_0.$$

Стадия восстановления секрета. Учитывая, что любой многочлен степени $t - 1$ однозначно восстанавливаются по его значениям в произвольных t точках, то любые t участников могут восстановить многочлен $f(x)$, т.е. найти значение секрета по формуле $s_0 = f(0)$. По этой же причине для любых $t - 1$ участников, любых значений s_i и любого секрета s_0 существует только один соответствующий им многочлен, для которого справедливо $s_i = f(\alpha_i)$ и $s_0 = f(0)$.

Схемы подобного типа находят применение при построении пороговых структур доступа и носят название (n, t)-*пороговых СРС*. Такие схемы, например, позволяет владельцу некоей секретной информации распределить эту информацию при хранении на n своеобразных ее дубликатов таким образом, что ему для восстановления секрета достаточно получить доступ к любым t из них. При этом никакие $t - 1$ дубликатов не предоставляют никакой информации об этом секрете [4].

6.6. Протоколы электронной подписи

Основные понятия. Схемы контроля целостности, рассмотренные в гл. 4, могут использоваться только при взаимодействии доверяющих друг другу сторон. Они принципиально не способны обеспечивать улаживание возникающих между ними противоречий и разногласий. Такая возможность появляется только при использовании односторонних функций с секретом для формирования электронной цифровой подписи. Протоколы или схемы электронной подписи – основное криптографическое средство обеспечения аутентичности информации:

с помощью электронной подписи получатель документа может доказать, что он принадлежит отправителю, при этом автор подписи не сможет оспорить факт отправки подписанного документа;

электронную подпись невозможно подделать, только абонент, чья подпись стоит на документе, мог подписать данный документ;

электронная подпись – неотъемлемая часть документа, перенести ее на другой документ нельзя;

ни противник, ни получатель не могут изменить документ, оставив данный факт незамеченным;

любое пользователь, имеющий образец подписи, может удостовериться в подлинности документа.

Для повышения надежности схемы электронной подписи (например, для противодействия атакам перехвата и повтора сообщений, фальсификации времени отправки сообщений) ее можно сделать зависимой от неповторяющегося блока данных *nrb*.

Схема электронной подписи включает в себя:

параметр безопасности n, в качестве которого может выступать длина подписи, длина подписываемых сообщений и т.п.;

пространство исходных сообщений;

максимальное число подписей (signature bound), которые могут быть получены в данной схеме *без замены секретной информации*;

алгоритм G генерации ключей – полиномиальный (от n) вероятностный алгоритм, формирующий по заданному параметру n пару

$$\left(k_A^{(secret)}, k_A^{(public)}\right),$$

где $k_A^{(secret)}$ – секретный ключ подписывающего (абонента A), $k_A^{(public)}$ – соответствующий открытый ключ проверяющего (абонента B);

алгоритм S формирования подписи сообщения – полиномиальный вероятностный алгоритм, вырабатывающий по заданным исходному сообщению M и секретному ключу $k_A^{(secret)}$ подпись s для сообщения M;

алгоритм V проверки (verification) *подписи* – полиномиальный вероятностный алгоритм, дающий на выходе при заданных сообщению M, открытому ключу $k_A^{(public)}$ и возможной подписи s' либо значение 1, когда подпись сообщения принимается, либо 0, когда подпись отвергается.

Подпись $s = S_{k_A^{(secret)}}(M)$ называется *допустимой* для сообщения M, если она принимается алгоритмом V с вероятностью, близкой к 1. *Подделкой подписи* сообщения M называется нахождение противником, не имеющим секретного ключа подписывающего, допустимой подписи для этого сообщения.

В общем случае неинтерактивный протокол электронной подписи имеет следующий вид.

Схема электронной подписи документа M

1. Отправитель A вычисляет
$$\left(k_A^{(secret)}, k_A^{(public)}\right) = G(n)$$
и посылает $k_A^{(public)}$ получателю B, сохраняя $k_A^{(secret)}$ в секрете.
2. Для получения подписи документа M отправитель A вычисляет
$$s = S_{k_A^{(secret)}}(M)$$
и посылает M и s получателю B.
3. Получатель B вычисляет

$$V\left(M, s, k_A^{(public)}\right)$$

и в зависимости от результата принимает или отвергает подпись s сообщения M.

В классической схеме электронной подписи предполагается, что подписывающий (абонент A) знает содержание документа M, который он подписывает; проверяющий (абонент B), зная открытый ключ проверки подписи, может проверить правильность подписи в любое время без какого-либо разрешения или участия претендента A. При создании электронной подписи по классической схеме (рис. 6.5) претендент A выполняет следующую последовательность действий:

применяет к исходному сообщению M хеш-функцию $h(x)$ и формирует хеш-образ или дайджест (message digest) сообщения $h(M)$;

при необходимости дополняет хеш-образ до требуемой длины;

вычисляет электронную подпись s с использованием несимметричного криптоалгоритма и секретного ключа создания подписи:

$$s = D_A^{(secret)}(Ext(h(M))).$$

Хеш-функция является неотъемлемой частью схем электронной подписи. Применение медленных несимметричных криптоалгоритмов для преобразования всего исходного сообщения M нерационально, поэтому для повышения быстродействия схемы перед процедурой формирования подписи используется функция необратимого сжатия информации. Последовательность действий абонента B, получившего подписанное сообщение:

верификатор B применяет к тексту полученного сообщения хеш-функцию;

дополняет в случае необходимости хеш-образ сообщения до требуемой длины (на рис. 6.5 рассматривается случай, когда расширение $Ext(h(M))$ не требуется);

применяет к полученной подписи s несимметричный криптоалгоритм с использованием открытого ключа проверки подписи и сравнивает полученное значение $E_A^{(public)}(s)$ с найденным на предыдущем шаге $Ext(h(M))$; при положи-

тельном результате сравнения подпись признается, в противном случае – отвергается [4].

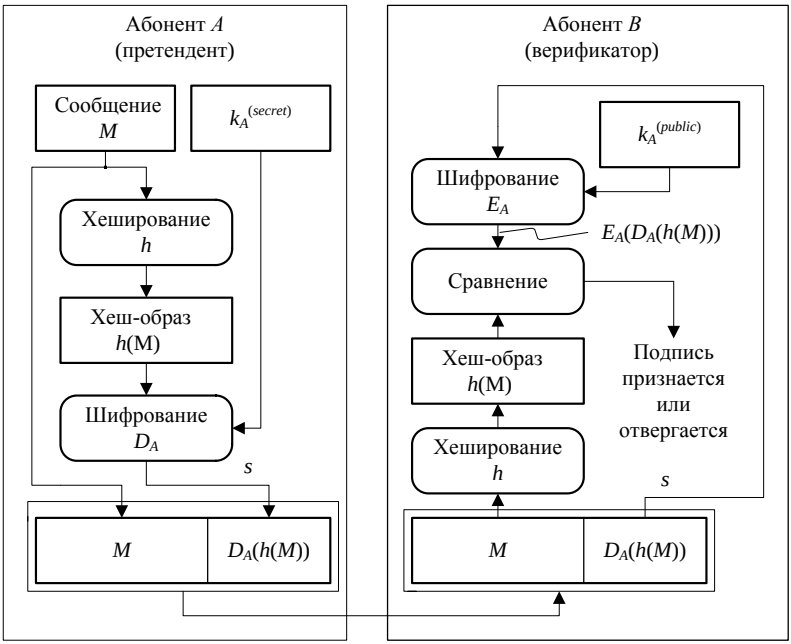


Рис. 6.5. Классическая схема создания и проверки электронной подписи

6.6.1. Протокол электронной подписи RSA

Криптосистема с открытым ключом RSA часто используется не только для шифрования, но и для построения схемы электронной подписи. Пусть $E_A(x) = x^e \bmod n$ – открытая функция зашифрования, а $D_A(x) = x^d \bmod n$ – секретная функция расшифрования.

Схема электронной подписи RSA

1. Абонент *A* (претендент) вычисляет хеш-образ $h(M)$ сообщения *M*.
2. Абонент *A* зашифровывает полученное значение $h(M)$ на своем секретном ключе *d*, вычисляя подпись

$$s = (h(M))^d \bmod n,$$

и отправляет абоненту B пару документ-подпись (M, s).

3. Абонент B (верификатор) расшифровывает s на открытом ключе e отправителя, т.е. вычисляет

$$s^e \bmod n.$$

4. Абонент B вычисляет хеш-образ $h(M)$ полученного сообщения и проверяет равенство

$$h(M) = s^e \bmod n.$$

В случае положительного результата проверки подпись принимается, в противном случае – отвергается.

В качестве хеш-функции в схеме подписи RSA используются функции семейства MD.

6.6.2. Протокол электронной подписи Шнорра

В протоколе аутентификации Шнорра интерактивность требуется только для того, чтобы получить от верификатора B случайный запрос x_B. Если бы у претендента A был бы надежный источник случайности на замену генератору ПСЧ B, пользующийся доверием верификатора B, то протокол можно было бы сделать неинтерактивным. Фиат и Шамир предложили способ преобразования схемы аутентификации в протокол электронной подписи. Если M – подписываемое сообщение, а $h(x)$ – криптографическая хеш-функция, вместо обращения к верификатору (получателю) претендент (отправитель) вычисляет $h(M)$ и использует ее вместо запроса x_B. Этот прием универсален, так как применим и к другим протоколам аутентификации.

Пусть p и q – такие простые числа, что q делит $(p-1)$. Пусть $g \in Z_p$ такое, что

$$g^q = 1 (\bmod p),\ g \neq 1.$$

Пусть хеш-функция $h(x)$ отображают пары значений (y_A, M) на множество $\{0, 1, ..., (2^t - 1)\}$. В качестве своего секретного ключа $k_A^{(secret)}$ абонент A выбирает случайное число из Z_q. Затем он вычисляет

$$g^{-k_A^{(secret)}} \bmod p$$

и публикует найденное значение в качестве своего открытого ключа $k_A^{(public)}$.

Схема электронной подписи Шнорра

1. Абонент A выбирает случайное число $x_{1A} \in Z_q$, после чего вычисляет
$$y_A = g^{x_{1A}} \bmod p.$$
2. Абонент A вычисляет
$$x_{2A} = h(y_A, M).$$
3. Абонент A вычисляет
$$s = x_{1A} + k_A^{(secret)} x_{2A} (\bmod q)$$
и посылает сообщение M с подписью (x_{2A}, s) абоненту B.
4. Абонент B вычисляет
$$y'_A = g^s \left(k_A^{(public)}\right)^{x_{2A}} \bmod p$$
и проверяет, выполняется ли равенство
$$x_{2A} = h(y'_A, M).$$
Если оно выполняется, B признает подлинность подписи, в противном случае – отвергает.

Стойкость приведенной схемы сильно зависит от свойств используемой хеш-функции. Если противник может по заданной паре (y_A, M) находить другое сообщение $M', M' \neq M$, для которого

$$h(y_A, M) = h(y_A, M'),$$

он может осуществить подделку подписи. Для этого после перехвата сообщения M и подписи (x_{2A}, s) он должен найти указан-

ное выше M', и тогда пара (x_{2A}, s) будет также подписью и для сообщения M'.

6.6.3. Классификация атак на схемы электронной подписи

Стойкость схемы электронной подписи зависит от стойкости используемых криптоалгоритмов и хеш-функций и определяется относительно пары угроза-атака. Приведем классификацию атак на схемы электронной подписи:

атака на основе известного открытого ключа (key-only attack) – самая слабая из атак, практически всегда доступная противнику;

атака на основе известных подписанных сообщений (known-message attack) – в распоряжении противника имеется некоторое (полиномиальное от k) число пар (M, s), где M – некоторое сообщение, а s – допустимая подпись для него, при этом противник не может влиять на выбор M;

простая атака с выбором подписанных сообщений (generic chosen-message attack) – противник имеет возможность выбрать некоторое количество подписанных сообщений, при этом открытый ключ он получает после этого выбора;

направленная атака с выбором сообщений (directed chosen-message attack) – выбирая подписанные сообщения, противник знает открытый ключ;

адаптивная атака с выбором сообщений (adaptive chosen-message attack) – противник знает открытый ключ, выбор каждого следующего подписанного сообщения он может делать на основе знания допустимой подписи предыдущего выбранного сообщения.

Каждая атака направлена на достижение определенной цели. Можно выделить следующие виды *угроз для схем электронной подписи* (в порядке возрастания силы):

экзистенциальная подделка (existential forgery) – создание противником подписи для какого-нибудь, возможно бессмысленного, сообщения M', отличного от перехваченного;

селективная подделка (selective forgery) – создание подписи для заранее выбранного сообщения;

универсальная подделка (universal forgery) – нахождение эффективного алгоритма формирования подписи, функционально эквивалентного *S*;

полное раскрытие (total break) – вычисление секретного ключа, возможно отличного от $k_A^{(secret)}$, соответствующего открытому ключу $k_A^{(public)}$, что дает возможность формировать подписи для любых сообщений.

Наиболее стойкими являются схемы, стойкие против самой слабой из угроз на основе самой сильной из атак, т.е. против экзистенциальной подделки на основе атаки с выбором подписанных сообщений. Справедливо следующее утверждение. *Схемы электронной подписи, стойкие против экзистенциальной подделки на основе атаки с выбором подписанных сообщений, существуют тогда и только тогда, когда существуют односторонние функции* [4].

6.6.4. Процедура разрешения споров

Для практического применения схем электронной подписи помимо алгоритмов формирования подписи и ее верификации требуется процедура арбитража, т.е. разрешения споров. Арбитраж необходим, когда один из абонентов, например *B*, предъявляет сообщение и подпись (M, s), утверждая, что эта пара сообщение-подпись была получена от абонента *A*, который отказывается признавать эту подпись своей. С юридической точки зрения основанием для разрешения подобных споров в суде является подписание (обычным образом) каждым пользователем при подключении к системе специального документа, в котором пользователь принимает все «правила игры», вплоть до судебной ответственности. При разборе дела в суде арбитр выступает в качестве эксперта, дающего заключение о подлинности электронной подписи [4].

Алгоритм арбитража

1. Абонент *B* предъявляет арбитру электронный документ и подпись.
2. Арбитр требует от абонента *A* предъявления своего секретного ключа. Если *A* отказывается, арбитр дает заключение, что подпись подлинная.
3. Арбитр выбирает из сертифицированного справочника открытый ключ абонента *A* и проверяет его соответствие секретному ключу, предъявленному *A*. Если они совпадают, арбитр переходит к шагу 5.
4. При обнаружении факта несоответствия ключей арбитр обращается в центр сертификации и требует предоставления заверенного абонентом *A* документа, содержащего его открытый ключ. Если выясняется, что открытый ключ, взятый из справочника, не совпадает с указанным в документе, арбитр признает подпись, предъявленную *B*, подлинной; при этом все издержки такого решения компенсируются за счет центра. Если открытые ключи в справочнике и документе совпадают, т.е. абонент *A* предъявил некорректный секретный ключ, арбитр признает подлинность электронной подписи.
5. Арбитр проверяет соответствие друг другу подписи и документа. При положительном результате проверки подпись признается подлинной, в противном случае – отвергается.

Задача арбитража значительно сложнее, чем кажется на первый взгляд. Арбитраж и решение споров в суде невозможны в следующих случаях:

секретный ключ сформирован не самим абонентом *A*, а специальным центром генерации ключей;

аппаратура, на которой выполняется алгоритм генерации или проверки подписи, содержит какие-либо элементы, не контролируемые пользователем («черные ящики», защищенные участки памяти и т.п.).

Наконец, возможны безвыходные ситуации, в которых арбитр не может принять никакого обоснованного решения. Например, абонент *B* предъявляет *s* и утверждает это подпись под докумен-

том M. Абонент A признает, что это его подпись, но под документом $M' \neq M$, при этом выясняется, что хеш-образы этих документов совпадают, т.е. $h(M) = h(M')$. Арбитр понимает, что кто-то из двоих нашел коллизию для применяемой в схеме электронной подписи хеш-функции, и оказывается в патовой ситуации. Выход из положения возможен только в том случае, если заранее обговорен порядок разрешения спора в такой ситуации.

6.6.5. Особые схемы электронной подписи

В некоторых ситуациях могут потребоваться схемы электронной подписи, отличные от рассмотренных классических схем. Известны следующие специальные схемы электронной подписи:

схема слепой подписи, когда абонент A подписывает документ, не зная его содержимого;

схема групповой подписи, которая позволяет верификатору убедиться в принадлежности полученного сообщения некоторой группе претендентов, но кто именно из членов группы подписал документ верификатор определить не в состоянии;

схема разделяемой подписи, которая формируется только при участии определенного количества участников протокола, иначе говоря, данная схема является объединением классической схемы подписи и *схемы разделения секрета*;

схема конфиденциальной (неотвергаемой) подписи, которая не может быть проверена без участия сформировавшего ее участника протокола;

схема неоспоримой подписи, в которой подделка подписи может быть доказана.

6.7. Протоколы аутентификации удаленных абонентов

6.7.1. Симметричные методы аутентификации субъекта

Аутентификация субъекта может осуществляться с использованием как симметричных, так и несимметричных криптоалгоритмов. В системах с большим числом пользователей примене-

ние симметричных методов требует введения в сеанс связи доверенной стороны, с которой разделяют секретные ключи все пользователи системы. На рис. 6.6 показана схема процедуры взаимной аутентификации субъектов A и B с использованием доверенной третьей стороны – субъекта C, обладающего секретными ключами k_{AC} и k_{BC} соответственно для взаимодействия с A и B. Последовательность шагов процедуры следующая:

1) субъект A, который хочет взаимодействовать с субъектом B, посылает C сообщение, содержащее идентификаторы субъектов запрашиваемого взаимодействия

$$\{ID_A, ID_B\};$$

2) C, получив сообщение, формирует сеансовый ключ k_{AB} для взаимодействия субъектов A и B и посылает A зашифрованное сообщение

$$E_{AC}(ID_B, k_{AB}, E_{BC}(ID_A, k_{AB})),$$

содержащее сеансовый ключ для работы с B и шифровку, которая по сути является разрешением для A на работу с B;

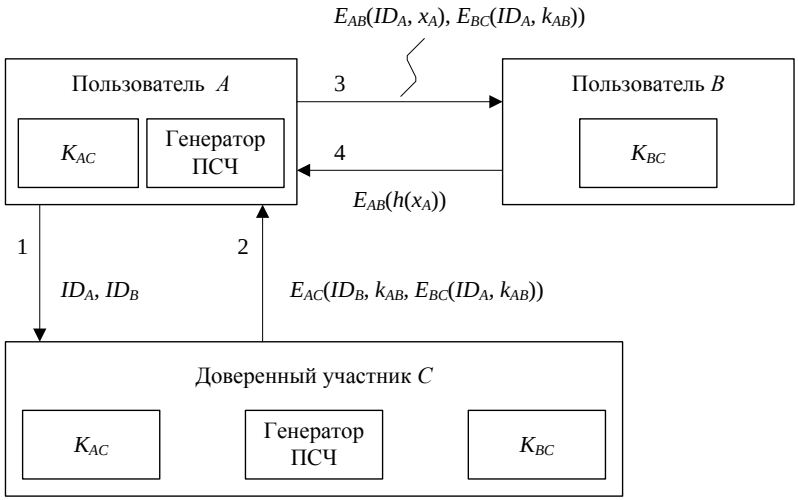


Рис. 6.6. Схема симметричной аутентификации Нидхэма-Шредера с доверенной третьей стороной

3) субъект A, расшифровав полученное сообщение, определяет ключ k_{AB} и разрешение

$$E_{BC}(ID_A, k_{AB}),$$

которое он расшифровать не может, так как не знает ключа k_{BC}; после этого A отправляет B сообщение

$$\{E_{AB}(ID_A, x_A), E_{BC}(ID_A, k_{AB})\},$$

содержащее зашифрованный запрос x_A и разрешение, полученное от C;

4) B, прочитав шифровку

$$E_{BC}(ID_A, k_{AB}),$$

узнает идентификатор субъекта взаимодействия и сеансовый ключ k_{AB} для работы с ним, читает запрос x_A; после этого B формирует ответ на запрос $h(x_A)$ и отправляет A сообщение

$$E_{AB}(ID_B, h(x_A));$$

5) субъект A, получив сообщение, расшифровывает его и проверяет ответ B; в случае положительного результата проверки, процесс аутентификации успешно завершается.

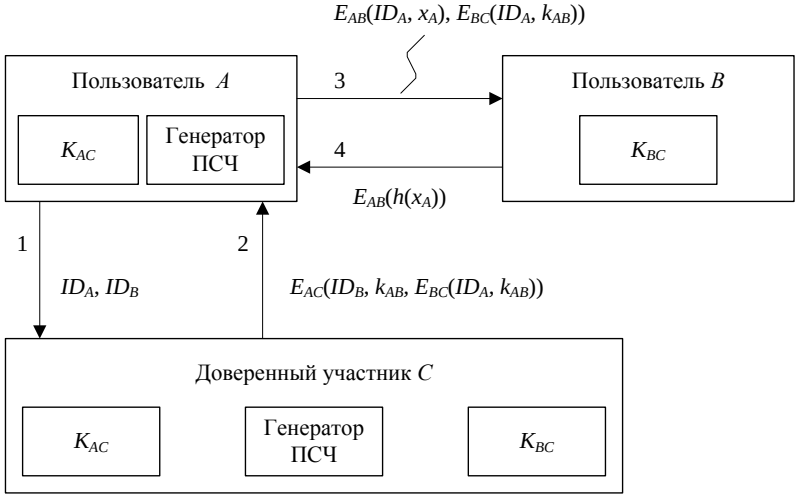


Рис. 6.6. Схема симметричной аутентификации Нидхэма–Шредера с доверенной третьей стороной

6.7.2. Схема Kerberos

Решение задачи аутентификации в современных информационных системах, представляющих собой совокупность реализованных на различных аппаратно-программных платформах территориально разнесенных компонентов, в соответствии с технологией «клиент/сервер» заключается в использовании специального сервера аутентификации. В настоящее время роль фактического стандарта сервера аутентификации выполняет *Kerberos*, продукт разработанный в Массачусетском технологическом институте (MIT) в середине 1980-х гг. и претерпевший с тех пор ряд принципиальных изменений. Широкому распространению Kerberos способствовало то, что его версия, реализованная в MIT, является свободно распространяемым продуктом. Программные средства, выполняющие аутентификацию по схеме Kerberos, разработаны для всех популярных ОС. Поддержка службы Kerberos предусмотрена в некоторых современных сетевых ОС. Схема Kerberos – типичный пример реализации симметричных методов аутентификации.

Система (рис. 6.7) предусматривает взаимодействие между тремя программными компонентами – клиентом *C*, сервером Kerberos и прикладным сервером *S*. ПО сервера Kerberos разделено по своим функциям на две части: сервер аутентификации *AS* (Authentication Server) и сервер выдачи разрешений (билетов) *TGS* (Ticket Granting Server). Клиент *C* – это компьютер, на котором установлено клиентское ПО, способное участвовать во взаимодействии по протоколу Kerberos, и на котором зарегистрирован какой-либо пользователь. В некоторых случаях прикладной сервер может являться клиентом некоторого другого сервера (например, сервер печати может пользоваться услугами файлового сервера). Сервер *S* – субъект, предоставляющий ресурсы сетевым клиентам.

Клиент *C*, который хочет обратиться к прикладному серверу для получения его услуг, должен получить разрешение от *AS*. Разрешение – это зашифрованная информация, передаваемая клиентом серверу *S* или серверу *TGS*. Разрешение позволяет серверу убедиться в подлинности клиента. Все разрешения, кроме

первого, клиент получает от сервера *TGS*. Первое разрешение, разрешение на доступ к самому *TGS*, клиент получает от сервера *AS*. Разрешение – шифровка, полученная на секретном ключе, известном только серверу *S* и серверу Kerberos, поэтому первый, получив разрешение, может быть уверен, что оно поступило именно от серверу Kerberos. Разрешения – многоразовые, имеющие определенный срок жизни (несколько часов). Когда этот срок истекает, клиент должен вновь пройти процедуру аутентификации. При установлении каждого соединения используется временная метка, поэтому в сети должна действовать служба единого времени. Необходимость в сервере *TGS* объясняется стремлением сократить число сообщений, зашифрованных с использованием секретного ключа клиента, которому требуются услуги нескольких серверов. Именно поэтому сервер Kerberos «раздваивается» на сервер *AS* (с ним клиент взаимодействует при помощи секретного ключа $k_{C,AS}$) и сервер *TGS*, с которым клиент осуществляет дальнейшее взаимодействие при помощи только сеансовых ключей $k_{C,TGS}$. Компрометация сеансового ключа, имеющего очень короткое время жизни, – вещь значительно менее опасная, чем раскрытие секретного ключа.

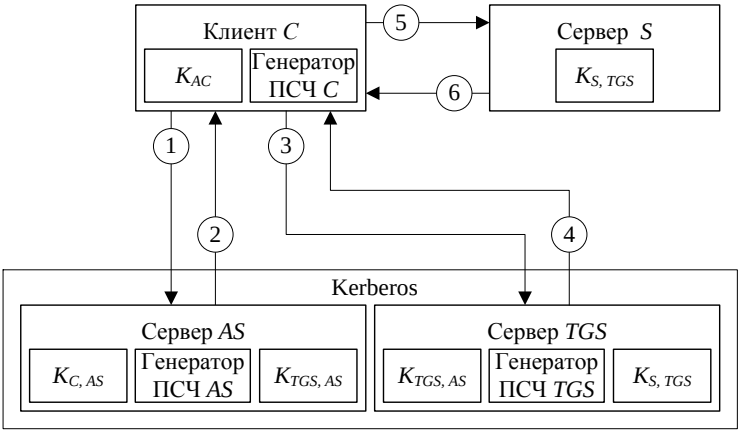


Рис. 6.7. Схема Kerberos

Процесс аутентификации состоит из пяти (односторонняя аутентификация) или шести (взаимная аутентификация) шагов.

1. Клиент *C* посылает серверу аутентификации сообщение, содержащее идентификаторы клиента и требуемого сервера выдачи разрешений, отвечающего за представление соответствующей услуги, а также информацию, предназначенную для идентификации конкретного запроса: время, свой сетевой адрес и т.п.

2. Сервер аутентификации осуществляет поиск в базе данных Kerberos по идентификатору клиента и идентификатору услуги, находит соответствующие ключи $k_{C,AS}$ и $k_{AS,TGS}$, формирует сеансовый ключ $k_{C,TGS}$ для взаимодействия клиента и сервера выдачи разрешений. После этого сервер AS посылает ответ клиенту. Этот ответ содержит две шифровки. Первая, полученная на секретном ключе клиента $k_{C,AS}$, содержит сеансовый ключ $k_{C,TGS}$ для работы с сервером выдачи разрешений, идентификатор последнего и срок жизни разрешения клиенту на работу с сервером *TGS*. Вторая шифровка, полученная на ключе $k_{AS,TGS}$, – это разрешение (ticket-granting ticket) клиенту на взаимодействие с сервером *TGS*. В состав второй шифровки, которую клиент прочесть не может, так как не знает ключа $k_{AS,TGS}$, входят идентификаторы клиента и *TGS*, сеансовый ключ $k_{C,TGS}$ и срок жизни этого разрешения.

3. Получив сообщение, клиент расшифровывает первую его половину на ключе $k_{C,AS}$ проверяет идентификатор запроса, узнает сеансовый ключ $k_{C,TGS}$ и срок жизни разрешения на работу с сервером *TGS*. Таким образом, в результате обмена сообщениями с сервером *AS* клиент получает разрешение на работу с сервером *TGS*. Затем клиент посылает запрос серверу выдачи разрешений. Сообщение для сервера *TGS* включает в себя две шифровки. Первая, полученная на сеансовом ключе $k_{C,TGS}$, включает в себя идентификаторы клиента и сервера, идентификатор запроса и временную метку. Вторая – это «запечатанное» ключом $k_{AS,TGS}$ разрешение на работу с сервером *TGS*.

4. Сервер выдачи разрешений расшифровывает разрешение, узнает сеансовый ключ $k_{C,TGS}$, с помощью которого читает первую часть пришедшего сообщения и проверяет идентификатор запроса и временную метку. Удостоверившись в подлинности клиента, сервер *TGS* вырабатывает сеансовый ключ $k_{C,S}$ для взаимодействия клиента *C* и сервера *S*. На знании этого ключа и основывается в будущем взаимная аутентификация *C* и *S*. После этого отправляет сообщение клиенту, содержащее зашифрованные на ключе $k_{C,TGS}$ сеансовый ключ и срок жизни разрешения клиенту на работу с сервером, а также само это разрешение, зашифрованное на секретном ключе $k_{S,TGS}$.

5. Клиент, получив сообщение, расшифровывает первую его часть, из которой извлекает сеансовый ключ $k_{C,S}$ для работы с сервером *S* и срок жизни разрешения на взаимодействие с сервером *S*. Само «запечатанное» ключом $k_{S,TGS}$ разрешение клиент прочесть не может. Таким образом, в результате обмена с сервером выдачи разрешений клиент получает разрешение на дальнейшее взаимодействие уже с прикладным сервером. Наконец, клиент посылает серверу *S* сообщение, содержащее зашифрованные на сеансовом ключе $k_{C,S}$ свой идентификатор, идентификатор запроса и временную метку, а также разрешение, полученное от сервера *TGS*.

6. Приняв сообщение от клиента и «распечатав» разрешение, целевой сервер узнает сеансовый ключ $k_{C,S}$ и с его помощью проводит аутентификацию клиента, проверяя идентификатор запроса и временную метку. Ответ сервера клиенту посылается в том случае, когда требуется взаимная аутентификация. Ответ прикладного сервера в этом случае содержит зашифрованный на ключе $k_{C,S}$ результат хеширования метки времени.

Сервер Kerberos имеет доступ к базе данных, содержащей идентификаторы и секретные ключи субъектов. Запись каждого пользователя и каждого прикладного сервера в базе данных Kerberos содержит следующие компоненты:

идентификатор субъекта;

секретный ключ субъекта;
дату истечения срока действия секретного ключа;
максимальной срок жизни разрешений, выдаваемых субъекту;
номер версии секретного ключа субъекта;
дату последней модификации записи;
другую служебную информацию.

Секретные ключи субъектов шифруются мастер-ключом Kerberos. Если секретный ключ изменяется нормальным образом (т.е. не в результате компрометации), то старый ключ следует сохранять до тех пор, пока остаются годными билеты, выданные с его помощью, так как возможна ситуация, когда один субъект имеет несколько активных ключей. Субъекту, обладающему несколькими активными ключами, соответствует несколько записей в базе данных. При выдаче разрешений и начальной аутентификации в сервере *AS* всегда используется самый свежий ключ.

Помимо серверов *AS* и *TGS* в системе имеется объект, отвечающий за управление базой данных, называемый диспетчером базы данных Kerberos DBM (Database Manager), а также сервер распространения ключей Kerberos KDS (Key Distribution Server).

6.7.3. Несимметричные методы аутентификации субъекта

Использование криптосистем с открытым ключом позволяет отказаться от серверов аутентификации, однако в системе должен существовать сервер, выдающий *сертификаты* на используемые субъектами взаимодействия открытые ключи. Сертификатом принято называть электронный документ, удостоверяющий принадлежность данного открытого ключа данному субъекту, иначе говоря, аутентичность ключа.

Протокол Диффи–Хеллмана. Несимметричные методы аутентификации могут быть основаны на использовании механизма электронной подписи. Классическим примером несимметричной аутентификации может служить *схема Диффи–Хеллмана*, представляющая собой совокупность процедуры выработки общего секретного ключа и взаимной аутентификации субъектов

взаимодействия. Пусть ω – примитивный элемент поля Галуа $GF(p)$, где p – простое число. Абонент A владеет парой функций: E_A – открытой функцией зашифрования и D_A – закрытой функцией расшифрования. Абонент B владеет парой функций: E_B – открытой функцией зашифрования и D_B – закрытой функцией расшифрования. Тогда последовательность взаимной аутентификации состоит из трех шагов.

Схема аутентификации Диффи-Хеллмана

1. Абонент A вырабатывает случайное число x_A и отправляет абоненту B сообщение

   $$\omega^{x_A} \bmod p.$$

2. Абонент B вырабатывает случайное число x_B, вычисляет

   $$\omega^{x_B} \bmod p,$$

   на своем секретном ключе создает подпись

   $$D_B\left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right)$$

   сообщения

   $$\left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right);$$

   затем B вычисляет сеансовый ключ

   $$k_{AB} = \omega^{x_A x_B} \bmod p,$$

   зашифровывает подпись на этом ключе и отправляет A сообщение

   $$\omega^{x_B} \bmod p, E_{AB}\left(D_B\left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right)\right).$$

3. Абонент A вычисляет сеансовый ключ

   $$k_{AB} = \omega^{x_A x_B} \bmod p,$$

   с помощью своего секретного ключа создает подпись

   $$D_A\left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right)$$

   сообщения

   $$\left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right),$$

   зашифровывает ее на ключе k_{AB} и отправляет B сообщение

$$E_{AB}\left(D_A\left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right)\right).$$

4. Если проверка подписи B абонентом A завершилась успешно, т.е. A убедился в справедливости равенства
$$E_B\left(D_{AB}\left(E_{AB}\left(D_B\left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right)\right)\right)\right) =$$
$$= \left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right),$$
он может быть уверен в подлинности абонента B.
5. Если проверка подписи A абонентом B завершилась успешно, т.е. B убедился в справедливости равенства
$$E_A\left(D_{AB}\left(E_{AB}\left(D_A\left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right)\right)\right)\right) =$$
$$= \left(\omega^{x_A} \bmod p, \omega^{x_B} \bmod p\right),$$
он в свою очередь может быть уверен в подлинности абонента A.

Протокол Шнорра – один из наиболее эффективных протоколов аутентификации. Пусть p и q – простые числа такие, что q делит $(p - 1)$. Шнорр предлагал использовать p разрядностью не менее 512 бит и q разрядностью не менее 140 бит. Пусть $g \in Z_p$ такое, что

$$g^q = 1 (\bmod p),\ g \neq 1.$$

Протокол Шнорра основан на отсутствии эффективных алгоритмов нахождения $x \in Z_q$ по заданному значению

$$y = g^x \bmod p$$

при известных p, q и g. Свойство нулевого разглашения знаний для схемы Шнорра пока не доказано.

В качестве своего секретного ключа $k_A^{(secret)}$ абонент A выбирает случайное число из Z_q. Затем он вычисляет

$$g^{\,k_A^{(secret)}} \bmod p$$

и публикует найденное значение в качестве своего открытого ключа $k_A^{(public)}$.

Схема аутентификации Шнорра

1. Абонент A выбирает случайное число $x_A \in Z_q$, вычисляет
$$y_A = g^{x_A} \bmod p$$
и посылает y_A абоненту B.
2. Абонент B выбирает случайное t-разрядное число
$$x_B \quad \left(x_B \in \left\{0,1,\ \ldots, (2^t - 1)\right\}\right)$$
и посылает его абоненту A.
3. Абонент A вычисляет
$$s = x_A + k_A^{(secret)} x_B \pmod q$$
и посылает s абоненту B.
4. Абонент B проверяет соотношение
$$y_A = g^s \left(k_A^{(public)}\right)^{x_B} \bmod p$$
и, если оно выполняется, признает подлинность абонента A, в противном случае – отвергает.

Противник, знающий только открытый ключ $k_A^{(public)}$, p, q и g, может пройти аутентификацию только с пренебрежимо малой вероятностью 2^{-t}, зависящей от параметра t. Противник может действовать следующим образом. Он выбирает случайным образом
$$x'_B \in \left\{0,1, \ldots, \left(2^t - 1\right)\right\}$$
и $s' \in Z_q$. Затем он вычисляет
$$y'_A = g^{s'} \left(k_A^{(public)}\right)^{x'_B} \bmod p$$
и посылает y'_A абоненту B. Запрос x_B, полученный от B на втором шаге, совпадет с x'_B с вероятностью 2^{-t}, и именно с такой вероятностью противник пройдет аутентификацию. Предлагая данную схему аутентификации, Шнорр рекомендовал использовать t разрядностью не менее 72 бит.

Протокол Фейга–Фиата–Шамира. Рассмотрим схему аутентификации с нулевым разглашением знаний (см. раздел 6.2). Подобные протоколы используются интеллектуальными карта-

ми. Личность владельца карты признается подлинной, если претендент доказывает свое знание секретного ключа.

Для реализации схемы необходим центр доверия, который выбирает и публикует целое число вида $n = pq$, где p и q – простые числа, которые держатся в секрете. Предположим, абонент A является доказывающим, абонент B – проверяющим. Абонент A (или центр доверия) формирует открытый и секретный ключи:

выбирает m случайных чисел $s_i \in Z_n$, $i = \overline{1, m}$, и объявляет их секретным ключом $\left(k_A^{(secret)}\right)$;

вычисляет m чисел v_i, удовлетворяющих условию

$$v_i^{-1} = s_i^2 \bmod n,$$

и объявляет их открытым ключом $\left(k_A^{(public)}\right)$.

Протокол аутентификации Фейга–Фиата–Шамира состоит в t-кратном повторении следующих шагов.

Схема Фейга–Фиата–Шамира

1. Абонент A выбирает случайное число $x_A \in Z_n$, вычисляет
$$y_A = x_A^2 \bmod n$$
и посылает его B.
2. Абонент B выбирает случайную двоичную последовательность
$$b_1 b_2 \ldots b_m,\ b_j \in \{0, 1\},\ j = \overline{1, m},$$
и посылает ее A.
3. Абонент A вычисляет
$$y'_A = x_A$$
и посылает его B.
4. Абонент B проверяет равенство
$$y_A = \left(y'_A\right)^2 \prod_{b_j=1} v_j \bmod n;$$

абонент B принимает доказательство, если проверка закончилась успешно во всех t циклах. Вероятность обмана равна $(1/2)^{mt}$.

Контрольные вопросы

1. Дайте определение криптографического протокола.
2. На чем основана стойкость протокола выработки общего секретного ключа?
3. Опишите процедуру разрешения споров при применении протокола ЭЦП.
4. Зачем в схеме ЭЦП применяется операция хеширования?
5. Опишите последовательность формирования и проверки ЭЦП.
6. Приведите пример формирования и проверки цифровой подписи RSA.
7. Перечислите функции генераторов ПСЧ в криптографических протоколах.
8. Какие криптографические протоколы являются стохастическими?
9. Перечислите функции хеш-генераторов в криптографических протоколах.
10. Опишите последовательность взаимной аутентификации удаленных абонентов в протоколе Нидхема–Шредера.
11. Разработайте протокол разделения секрета на четыре части.

ГЛАВА 7. УПРАВЛЕНИЕ КЛЮЧАМИ

Криптоалгоритмы и криптопротоколы основаны на использовании ключевой информации, под которой понимается вся совокупность действующих в КС ключей. По своему назначению они делятся на *ключи для шифрования ключей* и *ключи для шифрования данных*. По времени жизни ключи делятся на долго- и кратковременные. Примером последних являются так называемые *сеансовые ключи*, действующие в течение только одного сеанса связи. В понятие управление ключами входит совокупность методов решения следующих задач:

генерации ключей;
распределения ключей;
хранения ключей;
замены ключей;
депонирования ключей;
уничтожения ключей.

Правильное решение всех перечисленных задач имеет огромное значение, так как в большинстве случаев противнику гораздо проще провести атаку на ключевую подсистему или на конкретную реализацию криптоалгоритма, а не на сам алгоритм криптографической защиты. Использование стойкого алгоритма шифрования – необходимое, но далеко не достаточное условие построения надежной системы криптографической защиты информации. Используемые в процессе информационного обмена ключи нуждаются в не менее надежной защите на всех стадиях своего жизненного цикла.

7.1. Разрядность ключа

К ключам для симметричных и асимметричных криптосистем предъявляются различные требования. Этот факт следует учитывать при построении гибридных криптосистем. В настоящее время надежными считаются ключи разрядностью не менее 128 бит для систем с секретным ключом и не менее 2304 бит для систем с открытым ключом, стойкость которых опре-

деляется сложностью решения задачи факторизации больших чисел (например, RSA).

В распоряжении противника, атакующего криптосистему, всегда имеются две возможности: случайное угадывание ключа и полный перебор по всему ключевому пространству. Вероятность успеха и в том, и другом случае зависит от разрядности ключа. В табл. 7.1 приведены длины ключей симметричных и асимметричных систем, обеспечивающие одинаковую стойкость к атаке полного перебора и решению задачи факторизации соответственно.

Таблица 7.1

Длины ключей для криптосистем с секретным и открытым ключами

Длина ключа, бит	
Криптосистема с секретным ключом	Криптосистема с открытым ключом
56	384
64	512
80	768
112	1792
128	2304

Примечание. На практике в гибридных криптосистемах долговременный ключ для асимметричного алгоритма выбирают более стойким, чем сеансовый ключ для симметричного.

Если противник обладает неограниченными финансовыми и техническими возможностями, для того чтобы узнать ключ, ему необходимо лишь потратить достаточное количество денег. В случае противника с ограниченными возможностями при выборе разрядности ключа учитывают следующие соображения:

сложность атаки полного перебора;

требуемое быстродействие криптоалгоритма в тех случаях, когда увеличение размера ключа увеличивает время работы операций шифрования;

время жизни защищаемой информации и ее ценность;

возможности противника.

Если технические возможности противника известны, сложность атаки путем полного перебора по всему ключевому пространству оценить достаточно просто. Например, при разрядности ключа симметричной криптосистемы, равной 64 бит, объем ключевого пространства равен 2^{64}. Компьютер, который может перебирать 10^6 ключей в секунду, потратит на проверку всех возможных ключей более 5 тысяч лет.

В 1999 г. появилось сообщение [23], что международной группе исследователей удалось вскрыть шифр RSA с ключом длиной 512 бит. Интересно также отметить, что 512 двоичных разрядов – это максимальная длина ключа, которую до недавнего времени правительство США разрешало использовать в экспортируемых программных продуктах. Работа по подбору двух простых сомножителей числа, содержащего 155 десятичных цифр, велась в течение семи месяцев с привлечением ресурсов параллельно работающих 292 компьютеров, находящихся в 11 различных географических точках. В это количество входило 160 рабочих станций SGI и Sun, работающих на тактовых частотах 175–400 МГц, 8 компьютеров Origin 2000 SGI, работающих на частоте 250 МГц, 120 ПК с процессорами Pentium II (350–450 МГц) и 4 процессора, работающих на частоте 500 МГц, производства Digital/Compaq. Общие затраты вычислительных ресурсов составили около 8 тыс. MIPS-лет.

Взломанный код RSA

RSA – 155 =
1094173864157052742180970732204035761200373294544920599091384213147634998428893478471799725789126733249762575289978183379707653724402714674531593354333897 =
1026395928297411057720541965739916759007165678080380668033419335217907113077799

•

1066034883801684548209272203600128786720795857598929152227060823719306280864 3.

7.2. Генерация ключей

Согласно принципу Кирхгофа стойкость криптоалгоритма определяется секретностью ключа. Это означает, что если для генерации ключей используется криптографически слабый алгоритм, независимо от используемого шифра вся система будет нестойкой. Качественный ключ, предназначенный для использования в рамках симметричной криптосистемы представляет собой случайный двоичный набор. Если требуется ключ разрядностью n, в процессе его генерации с одинаковой вероятностью должен получаться любой из 2^n возможных кодов. Генерация ключей для асимметричных криптосистем процедура более сложная. Так, ключи, применяемые в этих системах, должны обладать определенными математическими свойствами. Например, в случае системы RSA модуль шифрования есть произведение двух больших простых чисел.

Для генерации ключевой информации, предназначенной для использования в рамках симметричной криптосистемы, используются следующие методы (в порядке возрастания качества):

программная генерация, предполагающая вычисление очередного псевдослучайного числа как функции текущего времени, последовательности символов, введенных пользователем, особенностей его клавиатурного почерка и т.п.;

программная генерация, основанная на моделировании качественного генератора ПСЧ с равномерным законом распределения;

аппаратная генерация с использованием качественного генератора ПСЧ;

аппаратная генерация с использованием генераторов случайных последовательностей, построенных на основе физических генераторов шума и качественных генераторов ПСЧ.

Невысокое качество программных методов формирования объясняется в первую очередь возможностью атаки на конкретную реализацию генератора и необходимостью защиты от разрушающих программных воздействий [29].

На рис. 7.1 показана схема генерации сеансовых ключей в стандарте ANSI X9.17, где E_k – функция зашифрования, k – ключ, V_0 – секретная синхропосылка, T_i – временная отметка, R_i – сеансовый ключ. Очевидно, что данная схема может применяться совместно с любым блочным шифром. Очередной сеансовый ключ формируется в соответствии с уравнением

$$R_i = E_k\left(E_k\left(T_i\right) \oplus V_i\right).$$

Новое значение V_{i+1} определяется следующим образом:

$$V_{i+1} = E_k\left(E_k\left(T_i\right) \oplus R_i\right).$$

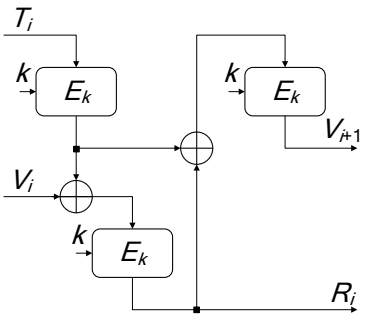


Рис. 7.1. Схема генерации ключей в стандарте ANSI X9.17

7.3. Неоднородное ключевое пространство

Для защиты от несанкционированного использования криптографического оборудования, которое противник не в состоянии вскрыть с целью исследования алгоритма, формируется *неоднородное ключевое пространство* (рис. 7.2), имеющее следующие особенности:

закрытие информации с использованием стойкого криптоалгоритма E_k имеет место только при использовании ключей специального вида;

«правильный» (стойкий) ключ k разрядности n_k $\left(\left|k\right| = n_k\right)$ состоит из двух частей: собственно ключа k^* разрядности n_{k^*} $\left(\left|k^*\right| = n_{k^*}\right)$ и некоторого кода фиксированной длины

$\Delta = n_k - n_{k^*}$, полученного в результате преобразования кода k^* с использованием некоторой криптографической функции F, т.е.

$$k = \left(k^*,\ F\left(k^*\right)\right),\ \left|F\left(k^*\right)\right| = \Delta;$$

вероятность случайно получить стойкий ключ пренебрежимо мала и равна $2^{-\Delta}$;

если поступивший на вход криптомодуля ключ не удовлетворяет указанным условиям, для шифрования информации применяется существенно менее стойкий алгоритм E'_k.

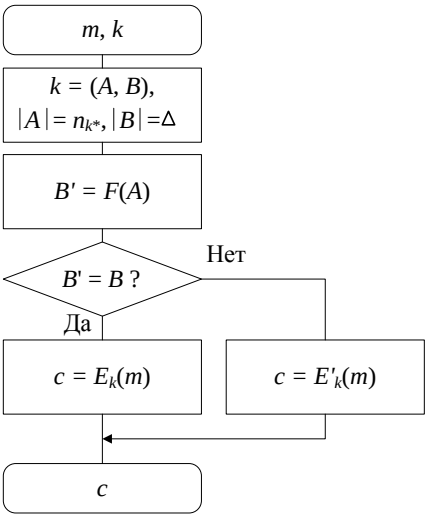


Рис. 7.2. Принцип работы криптомодуля
при использовании неоднородного ключевого пространства

7.4. Хранение ключей

Секретные ключи не должны храниться в памяти в явном виде, допускающем их считывание. Любая информация об используемых ключах должна хранится в зашифрованном виде, а значит, в защищенной системе должна иметь место иерархия ключей: либо двухуровневая (ключи шифрования ключей – ключи шифрования данных), либо трехуровневая (главный или

мастер-ключ – ключи шифрования ключей – ключи шифрования данных). Учитывая, что такое разделение функций необходимо для обеспечения максимальной безопасности, каждый из указанных типов ключей, различающихся по последствиям компрометации, времени жизни, а иногда и по способам формирования, должен использоваться только по своему прямому назначению. На самом нижнем уровне иерархии находятся ключи шифрования данных или сеансовые ключи, которые используются для шифрования пересылаемых сообщений или аутентификационной информации. Для защиты сеансовых ключей при их хранении и передаче используются ключи следующего уровня – ключи шифрования ключей. На верхнем уровне иерархии располагается мастер-ключ, используемый для защиты ключей шифрования ключей. Обычно в каждом компьютере используется один мастер-ключ.

Учитывая главенствующую роль в иерархии мастер-ключа, применяемого в течение длительного времени, его защите уделяется особое внимание:

мастер-ключ хранится в защищенном от считывания, записи и разрушающих воздействий модуле системы защиты;

мастер-ключ распространяется неэлектронным способом, исключающим его компрометацию;

в системе должен существовать способ проверки аутентичности мастер-ключа.

Один из способов аутентификации мастер-ключа показан на рис. 7.3. В памяти компьютера хранится пара (m, c), где m – некоторый массив данных, $c = E_k(m)$ – результат его зашифрования на мастер-ключе k_M. Всякий раз, когда требуется проверка аутентичности мастер-ключа, берется код m из памяти и подается на вход криптомодуля. Полученный с выхода последнего зашифрованный текст c' сравнивается с шифротекстом, хранящемся в памяти. При положительном результате сравнения, аутентичность мастер-ключа считается установленной [25].

При генерации сеансовых ключей код с выхода генератора ПСЧ рассматривается как результат шифрования $E_k(k_s)$ сеансового ключа k_s, полученный с использованием мастер-ключа k_M, и поэтому может храниться в том виде, в котором он был

получен (рис. 7.4). При шифровании сообщения на вход криптомодуля подаются шифротекст $E_k(k_s)$ и сообщение m. Криптомодуль сначала «восстанавливает» сеансовый ключ, а затем с его помощью шифрует сообщение [25].

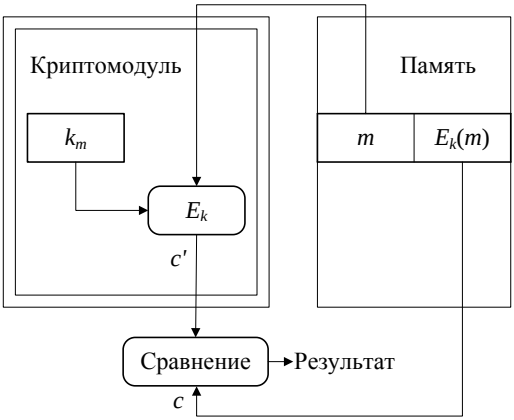


Рис. 7.3. Схема аутентификации мастер-ключа

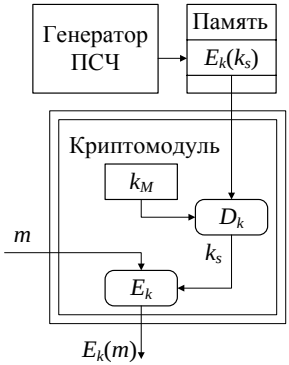


Рис. 7.4. Схема защиты сеансового ключа

Проще всего хранить ключи криптосистемы с одним пользователем. В распоряжении последнего один из следующих вариантов в порядке возрастания надежности:

запоминание пароля *pw* и в случае необходимости автоматическое получение из него ключа *k* с использованием хеш-функции $h(x)$ по формуле $k = h(pw)$;

запоминание начального заполнения качественного генератора ПСЧ, формирующего ключ;

использование пластикового ключа с размещенным на нем ПЗУ (ROM-key) или интеллектуальной карточки.

Следует помнить, что в первых двух случаях объем ключевого пространства зависит не только от длины пароля или ключевой фразы, но и от ограничения на вид используемых символов. В табл. 7.2 приведены количество возможных ключей при использовании 8-символьной ключевой фразы и время полного перебора при скорости перебора 10^6 ключей в секунду в зависимости от ограничений на используемые символы. Также следует помнить о возможности проведения противником так называемой *атаки со словарем*, включающем в себя наиболее вероятные ключевые слова. Пароли и символьные строки начального заполнения генератора ПСЧ следует выбирать случайным образом, а не на основе лишь критерия простоты запоминания.

Таблица 7.2

Количество возможных ключей и время их полного перебора при различных ограничениях на используемые символы

Символы ключа	Число символов	Число возможных ключей	Время полного перебора
Заглавные буквы	32	$1{,}1 \cdot 10^{12}$	13 дней
Заглавные буквы и цифры	42	$9{,}7 \cdot 10^{12}$	112 дней
Все 8-разрядные коды	256	$1{,}9 \cdot 10^{19}$	600 000 лет

В последнем случае ни сам ключ, ни исходная информация, необходимая для его получения, пользователю неизвестны, а значит, он не может и скомпрометировать ее. Использование ROM-key, имеющего тот же вид, что и привычный ключ от входной двери, позволяет пользователю чисто интуитивно избегать многих ошибок, связанных с хранением криптографических ключей [25].

7.5. Распределение ключей

Распределение ключей между участниками информационного обмена в сети реализуется двумя способами (рис. 7.5):

использованием одного или нескольких центров распределения или трансляции ключей;

прямым обменом сеансовыми ключами между пользователями сети.

Недостатком первого подхода является возможность злоупотреблений со стороны центра, которому известно, кому и какие ключи распределены. Во втором случае проблема заключается в необходимости проведения более качественной, чем в первом случае, процедуры аутентификации для проверки подлинности участников сеанса взаимодействия и достоверности самого сеанса. Примером первого подходя является схема Kerberos, примером второго – схема Диффи–Хеллмана (гл. 6).

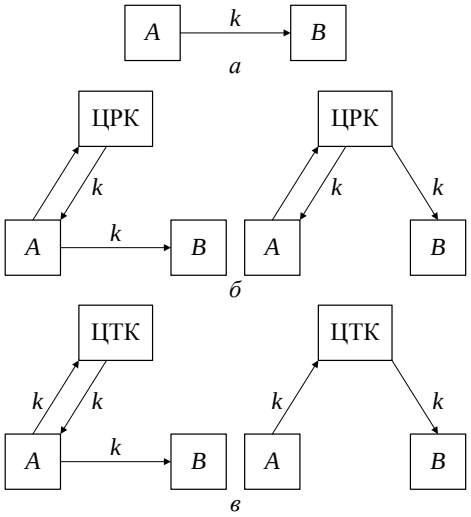


Рис. 7.5. Возможные варианты построения схемы распределения ключей:
а – прямой обмен между пользователями в сети (схема типа «точка-точка»);
б – схемы с использованием центра распределения ключей (ЦРК);
в – схемы с использованием центра трансляции ключей (ЦТК);
A, *B* – участники информационного обмена, *k* – ключ

Схемы, показанные на рис. 7.5, *б* и *в*, предполагают, что абоненты *A* и *B* предварительно разделили знание своих секретных ключей с центром *C*, а схемы с ЦРК – что именно он формирует сеансовые ключи. Схемы, показанные на рис. 7.5,*в*, отличаются от предыдущего варианта тем, что ЦТК обеспечивает только перешифрование полученной ключевой информации [25].

В двухключевых асимметричных криптосистемах существует опасность подмены открытого ключа одного или нескольких участников информационного обмена. Задача защиты открытых ключей от подделки является «ахиллесовой пятой» всей технологии. Пусть, например, противник *W*, имеющий пару ключей

$$\left(k_W^{(public)},\ k_W^{(secret)}\right)$$

подменил своим открытым ключом открытый ключ $k_A^{(public)}$ абонента *A*. В результате противник может:

читать все сообщения, адресованные *A*, так как обладает секретным ключом $k_A^{(secret)}$ из фальшивой пары ключей;

снова зашифровать расшифрованное им сообщение настоящим ключом $k_A^{(public)}$ и отправить его абоненту *A*, при этом никто ничего не заметит;

формировать от имени *A* электронную подпись, которая будет казаться подлинной, так как все для ее проверки будут использовать фальшивый ключ.

Учитывая вышеизложенное, одна из важнейших функций центра доверия – *сертификация открытых ключей* взаимодействующих абонентов. Например, сертификат $\tilde{c}_A$ открытого ключа абонента *A* суть шифрограмма, полученная на секретном ключе центра распределения *C*, которая содержит метку времени *ts*, время действия сертификата *lt*, идентификатор ID_A и открытый ключ $k_A^{(public)}$, т.е.

$$\tilde{c}_A = D_C\left(ts,\ lt,\ ID_A,\ k_A^{(public)}\right),$$

где D_C – функция шифрования с использование секретного ключа $k_A^{(secret)}$.

В самом общем случае функциями центра доверия могут являться:

генерация, хранение, распределение и контроль времени жизни ключей;

сертификация ключей;

аутентификация участников информационного обмена;

аудит;

служба единого времени;

нотариальная служба;

служба депонирования ключей.

Для уменьшения количества сеансов пересылки ключей применяется процедура модификации ключей, которая может быть основана, например, на получении нового ключа путем хеширования старого. Если правило смены ключей соблюдается и отправителем и получателем, то в каждый момент времени они имеют одинаковый ключ. Постоянная смена ключа затрудняет противнику решение задачи раскрытия информации. При использовании такого приема следует помнить, что вся ключевая последовательность будет скомпрометирована, если противнику станет известен хотя бы один из ранее использовавшихся старых ключей [25].

7.6. Время жизни ключей

Любой ключ должен использоваться в течение ограниченного отрезка времени, длительность которого зависит от:

частоты использования ключей (ключи щифрования ключей, например, применяются гораздо реже ключей шифрования данных);

величины ущерба от компрометации ключа, которая зависит, в частности, от ценности защищаемой информации;

объема и характера защищаемой информации (например, при шифровании случайным образом сформированных ключей закрытию подвергается информации, о содержимом которой противнику заранее ничего не известно).

При определении времени жизни ключа учитываются следующие соображения:

чем дольше используется ключ, тем больше вероятность его компрометации и тем больший потенциальный ущерб может нанести его компрометация;
чем больший объем информации, зашифрованной на одном ключе, перехватывает противник, тем легче проводить атаку на него;
при длительном использовании ключа у противника появляется дополнительный стимул потратить на его вскрытие значительные ресурсы, так как выгода в случае успеха оправдает все затраты [25].

Контрольные вопросы

1. Укажите, от чего зависит длительность использования ключевой информации.
2. Каково минимальная разрядность ключа для криптосистем с секретным ключом?
3. Каково минимальная разрядность ключа для криптосистем с открытым ключом?
4. Перечислите не менее пяти способов формирования ключевой информации.
5. Что такое неоднородное ключевое пространство?
6. Что такое мастер-ключ?
7. Опишите процедуру аутентификации мастер-ключа.
8. Что такое сертификат открытого ключа?

ЧАСТЬ 2. ОСНОВЫ ТЕОРИИ, ПРИМЕНЕНИЯ И ОЦЕНКИ КАЧЕСТВА ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

ГЛАВА 8. ПРИНЦИПЫ ПОСТРОЕНИЯ ГЕНЕРАТОРОВ ПСЕВДОСЛУЧАЙНЫХ ЧИСЕЛ

Сфера применения генераторов генераторов псевдослучайных чисел (ПСЧ) чрезвычайно широка. Качественные псевдослучайные последовательности (ПСП), являясь по своей сути детерминированными, обладают, тем не менее, практически всеми свойствами реализаций истинно случайных процессов и успешно их заменяют во многих приложениях, так как случайные последовательности чрезвычайно сложно формировать.

Часть 2 данного пособия посвящена в первую очередь генераторам ПСЧ, ориентированным на использование в системах защиты информации от случайных и умышленных деструктивных воздействий, иначе говоря, генераторам, к которым предъявляются наиболее жесткие требования.

В главе 8 дается классификация генераторов ПСЧ, рассматриваются общие принципы проектирования непредсказуемых генераторов ПСЧ, требования к таким устройствам, описываются основные строительные блоки, используемые при их создании. Уделяется внимание перспективным типам генераторов: генераторам на регистрах сдвига с линейными обратными связями, генераторам ПСЧ на основе стохастических сумматоров или *R*-блоков. Криптографические блочные и поточные генераторы ПСЧ, а также генераторы ПСЧ на основе односторонних функций были рассмотрены в предыдущих главах. Конгруэнтные генераторы ПСЧ не рассматриваются в связи с их неудовлетворительным качеством.

В главе 9 приведены примеры использования стохастических методов при обеспечении безопасности электронных платежных систем, а в главе 10 – информация по методам оценки качества генераторов ПСЧ.

Результаты исследований генераторов ПСЧ различных типов представлены в [16].

8.1. Стохастические методы защиты информации

Стохастическими методами защиты в широком смысле принято называть методы защиты компьютерных систем, прямо или косвенно основанные на использовании генераторов ПСЧ или хеш-генераторов, если учесть, что результат работы и тех, и других непредсказуем. При этом эффективность защиты в значительной степени определяется качеством используемых алгоритмов соответственно генерации ПСЧ или хеширования. Крипто- и стеганографические методы защиты, таким образом, являются лишь частным случаем стохастических методов.

Генераторы ПСЧ и хеш-генераторы успешно решают практически все задачи, стоящие перед разработчиками систем обеспечения безопасности информации (ОБИ). При реализации большинства методов защиты информации используются генераторы ПСЧ или хеш-генераторы. Иначе говоря, эти методы – стохастические. Более того, наиболее перспективный метод защиты, сутью которого является внесение неопределенности в работу компьютерных систем (реализация которого в принципе невозможна без использования генераторов ПСЧ), универсален. Он может использоваться совместно с любым другим методом защиты, автоматически повышая его качество.

Функции генераторов ПСЧ в системах ОБИ:

формирование тестовых воздействий на входы проверяемых компонентов системы при автономном или встроенном диагностировании;

реализация счетчиков команд или адреса (в том числе самопроверяемых) компьютерных систем;

определение соответствия между адресом порта ввода-вывода и запрашиваемой функцией при реализации плавающих протоколов взаимодействия ПО и устройств ввода-вывода (УВВ), механизма скрытых функций УВВ;

формирование элементов вероятностного пространства при внесении неопределенности в результат работы алгоритмов защиты информации (вероятностное шифрования, технология ОАЕР);

задание последовательности выполнения при внесении неопределенности в последовательность выполнения отдельных шагов алгоритма;
задание длительности выполнения при внесении неопределенности в длительность выполнения отдельных шагов алгоритма для защиты от утечки по побочным каналам;
формирование элементов вероятностного пространства при внесении неопределенности в механизм работы программных средств (полиморфизм, метаморфизм, запутывание программ (obfuscating));
формирование гаммы при шифровании информации в режимах гаммирования и гаммирования с обратной связью;
формирование ключей и паролей пользователей;
формирование случайных запросов при аутентификации удаленных абонентов по принципу «запрос–ответ»;
формирование долей секрета в протоколах разделения секрета;
формирование затемняющих множителей при слепом шифровании (например, для скрытия содержимого электронного документа при формировании слепой подписи);
формирование прекурсоров для защиты прав собственников информации (протокол слепой подписи).

Функции хеш-генераторов в системах ОБИ:

формирование контрольных кодов целостности информации или правильности выполнения шагов алгоритма;
необратимое преобразование паролей для защиты парольных систем разграничения доступа;
необратимое сжатие информации перед формированием электронной цифровой подписи (ЭЦП) для повышения производительности схемы ЭЦП;
необратимое преобразование случайных запросов при аутентификации по принципу запрос–ответ;
необратимое преобразование информации для защиты от утечки (например, в схеме двойной электронной подписи);
необратимое преобразование информации (прекурсора) с целью защиты прав ее владельца (например, при формировании серийного номера цифровой купюры);
необратимое преобразование информации при реализации метода внесения неопределенности в результат работы

криптоалгоритмов (например, при создании несепарабельных режимов шифрования, предназначенных для повышения стойкости симметричных криптоалгоритмов с небольшой разрядностью ключа или реализации технологии OAEP для асимметричных криптоалгоритмов).

Таким образом, можно сделать обоснованный вывод об универсальности стохастических методов (необходимо только помнить, что они являются методами двойного назначения, так как используются как при атаках, так и организации защиты компьютерных систем) и определяющей роли качественных быстродействующих генераторов ПСЧ. В случае их наличия можно эффективно строить все другие криптографические примитивы симметричной криптографии.

После обоснования утверждения о решающей роли генераторов ПСЧ при построении системы ОБИ, появляется возможность подойти к решению задач защиты компьютерных систем с единых исходных позиций. Ранее научные дисциплины, в той или иной степени связанные с решением задач ОБИ, развивались обособленно друг друга. Речь в первую очередь идет о технической диагностике, теории помехоустойчивого кодирования, криптографии и криптоанализе, стеганографии и стегоанализе, информационной безопасности, технологии безопасного программирования.

8.2. Требования к генераторам ПСЧ

Эффективность стохастических методов ОБИ определяется в первую очередь качеством используемых генераторов ПСЧ. Сформулируем, что входит в понятие «качественный генератор ПСЧ».

Качественный генератор ПСЧ**,** ориентированный на использование в задачах защиты информации, должен удовлетворять следующим требованиям:

непредсказуемость (по сути, криптографическая стойкость);

хорошие статистические свойства, ПСП по своим статистическим свойствам не должна отличаться от истинно случайной последовательности;

большой период формируемой последовательности, учитывая, что при преобразовании больших массивов данных каждому элементу входной последовательности необходимо ставить в соответствие свой элемент псевдослучайной последовательности;

эффективная программная и аппаратная реализация.

При использовании непредсказуемого генератора ПСЧ три следующие задачи для противника, не знающего ключевой информации, вычислительно неразрешимы:

1) определение предыдущего $(i - 1)$-го элемента γ_{i-1} последовательности на основе известного фрагмента последовательности $\gamma_i \gamma_{i+1} \gamma_{i+2} \ldots \gamma_{i+b-1}$ конечной длины b (*непредсказуемость влево*);
2) определение последующего $(i + b)$-го элемента γ_{i+b} последовательности на основе известного фрагмента гаммы $\gamma_i \gamma_{i+1} \gamma_{i+2} \ldots \gamma_{i+b-1}$ конечной длины b (*непредсказуемость вправо*);
3) определение ключевой информации по известному фрагменту последовательности конечной длины.

Справедливо следующее утверждение [3, 4]: *непредсказуемый влево генератор ПСЧ является криптостойким.*

Криптоаналитик, знающий принцип работы такого генератора, имеющий возможность анализировать фрагмент выходной последовательности конечной длины, но не знающий используемой ключевой информации, для определения предыдущего выработанного элемента последовательности не может предложить лучшего способа, чем подбрасывание жребия.

Непредсказуемость генератора ПСЧ чрезвычайно сложно оценить количественно. Чаще всего обоснования стойкости нелинейной функции F_k генератора ПСЧ сводятся к недоказуемым предположениям о том, что у аналитика не хватит ресурсов (вычислительных, временных или стоимостных) для того, чтобы при неизвестном k обратить (инвертировать) эту функцию. Теория сложности вычислений, к сожалению, не может дать строгую нижнюю границу трудоемкости решения подобных задач.

В рамках другого подхода к построению качественного генератора ПСЧ предлагается свести задачу построения криптогра-

фически сильного генератора к задаче построения *статистически безопасного генератора*.

Статистически безопасный генератор ПСЧ должен удовлетворять следующим требованиям:

ни один статистический тест не обнаруживает в ПСП каких-либо закономерностей (иными словами, не отличает эту последовательность от истинно случайной);

нелинейное преобразование F_k, зависящее от секретной информации (ключа k), используемое для построения генератора, обладает свойством «размножения» искажений – все выходные (преобразованные) вектора e' возможны и равновероятны независимо от исходного вектора e (рис. 8.3);

при инициализации случайными значениями генератор порождает статистически независимые ПСП.

8.3. Классификация генераторов ПСЧ

Классификация генераторов ПСЧ показана на рис. 8.1. В качестве параметров выбраны:

тип используемого нелинейного преобразования;

структура генератора;

наличие или отсутствие внешних источников случайности.

Тип функции. По этому параметру генераторы ПСЧ можно разделить на некриптографические и криптографические. К некриптографическим относятся конгруэнтные генераторы, генераторы, функционирующие в конечных полях, и генераторы на регистрах сдвига с нелинейными обратными связями. К криптографическим – блочные и поточные генераторы (см. соответственно разделы 2.4 и 2.8), генераторы на основе односторонних функций (см. раздел 5.8).

Достоинство некриптографических генераторов – эффективная программная и аппаратная реализация. Недостаток – предсказуемость. Аддитивные генераторы Галуа и Фибоначчи, функционирующие в конечных полях, генераторы на регистрах сдвига с нелинейными обратными связями можно использовать лишь в качестве строительных блоков при разработке качественных генераторов ПСЧ.

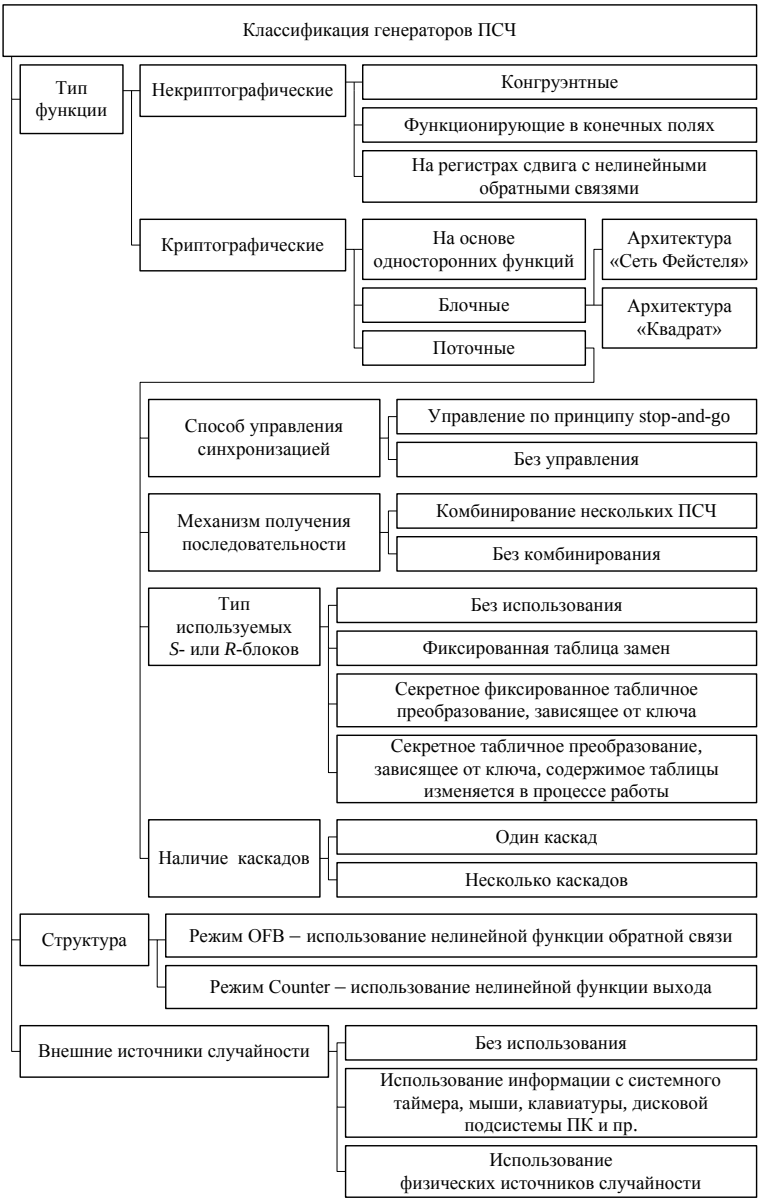


Рис. 8.1. Классификация генераторов ПСЧ

В криптографических генераторах ПСЧ в качестве нелинейного преобразования F_k используется функция зашифрования либо симметричной (блочные и поточные генераторы ПСЧ), либо асимметричной криптосистемы (генераторы ПСЧ на основе односторонних функций с секретом). При использовании в качестве нелинейной функции F_k функции зашифрования E_k одноключевой или двухключевой криптосистемы применение криптостойких функций E_k автоматически придает аналогичное по сути свойство непредсказуемости и генератору ПСЧ.

При построении блочных криптографических генераторов в первую очередь уделяется внимание их непредсказуемости. Нелинейное преобразование, определяющее свойство непредсказуемости, суть многократное повторение одной и той же раундовой операции.

Основной целью построения поточных генераторов является высокая скорость работы при приемлемой для большинства приложений непредсказуемости. В отличие от блочных генераторов ПСЧ здесь нет единого принципа построения. Можно выделить лишь следующие перспективные тенденции:

применение в качестве строительных блоков генераторов, функционирующих в конечных полях (генераторы SOBER, PANAMA, SNOW и др.) [24];

использование при построении нелинейных функций блоков пространственного сжатия (space compression) информации для необратимости результирующего преобразования [29];

применение таблиц замен или стохастического преобразования, непрерывно изменяющихся в процессе работы (RC4, SQ1-R и др.) [24, 34].

Наиболее обоснованными математически следует признать генераторы с использованием односторонних функций. Непредсказуемость данных генераторов основывается на сложности решения ряда математических задач (например, задачи дискретного логарифмирования (в том числе на эллиптических кривых) или задачи разложения больших чисел на простые множители). Существенный недостаток генераторов этого класса – низкая производительность.

Анализ криптографических генераторов позволяет сделать два основных вывода:

1) существует трудноразрешимое противоречие между качеством формируемых ПСЧ, с одной стороны, и эффективностью программной и аппаратной реализации генераторов, с другой;
2) непредсказуемость криптографических генераторов основывается на недоказуемых предположениях о том, что у аналитика не хватит ресурсов (вычислительных, временных или стоимостных) для того, чтобы обратить (инвертировать) нелинейную функцию обратной связи или нелинейную функцию выхода генератора ПСЧ.

Структура генератора. Можно выделить два подхода при использовании в составе генераторов ПСЧ нелинейных функций:

1) применение нелинейной функции непосредственно в цепи обратной связи (рис. 8.2,*а*)
2) двухступенчатая структура (рис. 8.2,*б*), в которой задача первой ступени (по сути счетчика) заключается всего лишь в обеспечении максимально большого периода при данном количестве N элементов памяти генератора.

Первая схема носит название OFB (Output FeedBack – обратная связь по выходу), вторая – Counter.

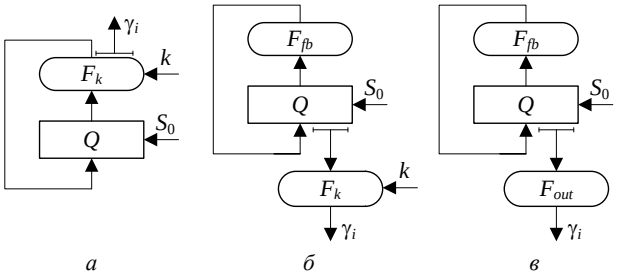


Рис. 8.2. Варианты построения генератора ПСЧ:
а – с нелинейной внутренней логикой (режим OFB);
б – с нелинейной внешней логикой (режим Counter); *в* – общая схема.
Q – элементы памяти генератора; S_0 – начальное состояние генератора;
F_{fb} – линейная или нелинейная функция обратной связи;
F_k – нелинейная функция, результат работы которой зависит от секретного параметра k (ключа);, γ_i – элемент выходной последовательности;
F_{out} – линейная или нелинейная функция выхода генератора

Уравнения работы генераторов, функционирующих в режимах OFB и Counter, имеют соответственно вид

$$\begin{cases} \gamma_t = Q_t, \\ Q_{t+1} = F_{fb}(Q_t), \end{cases} \tag{8.1}$$

$$\begin{cases} \gamma_t = F_{out}(Q_t), \\ Q_{t+1} = F_{fb}(Q_t), \end{cases} \tag{8.2}$$

где $Q_0 = S_0$; Q_t и Q_{t+1} – состояние генератора в моменты времени t и $(t + 1)$ соответственно; γ_t – значение выхода в момент времени t,

Необходимые свойства используемой нелинейной функции иллюстрирует рис. 8.3, где e – входной вектор изменений (ошибок), содержащий 1 в разрядах, соответствующих измененным (искаженным) битам, e' – преобразованный (выходной) вектор изменений. При случайном выборе ключа k при любых изменениях на входе значения преобразованных векторов e' равномерно распределены на интервале $[1; 2^n - 1]$, где n – разрядность выходного слова (рис. 8.3,*а*). Аналогично, при случайном выборе входного слова x при любых изменениях ключа значения преобразованных векторов e' ошибок также равномерно распределены на интервале $[1; 2^n - 1]$ (рис. 8.3,*б*). Такие свойства автоматически имеют место при использовании в роли F_k функции зашифрования качественного блочного криптоалгоритма, обеспечивающей рассеивание и перемешивание информации.

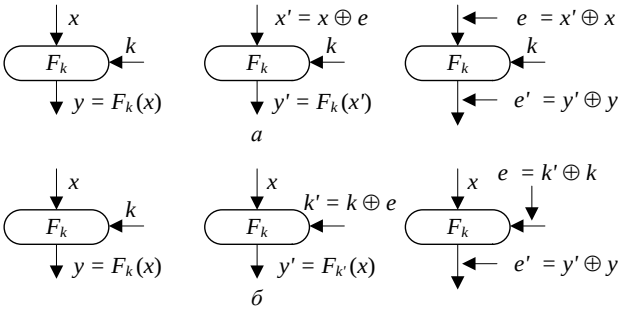


Рис. 8.3. Свойства нелинейной функции генератора ПСЧ:
а и *б* – входной и преобразованный векторы соответственно при изменении на входе и изменении ключа k

Внешние источники случайности. В тех приложениях, где требуется неоднократно формировать одну и ту же последовательность (например, при генерации гаммы) генераторы ПСЧ не имеют внешних источников случайности, во всех же других случаях их использование необходимо. При программной реализации в качестве источников случайности обычно используются системный таймер, клавиатура, «мышь», дисковая подсистема персонального компьютера, стек протоколов TCP/IP и др. Классическим примером такого генератора ПСЧ может служить проект Yarrow фирмы Counterpane Systems Б. Шнайера. При аппаратной реализации в качестве источников случайности могут использоваться физические датчики случайных чисел.

На рис. 8.4 показана схема аппаратно-программного комплекса генерации ПСЧ, использующего внешние источники случайности. Можно перечислить следующие принципы его проектирования:

простота встраивания в уже готовые системы;

простота использования, позволяющая квалифицированному программисту, даже не имеющему знаний в криптографии, безопасно работать с ним;

использование существующих сильных криптографических примитивов.

В схеме применены два криптографических примитива – хеш-функция в блоках хеширования и блочный генератор ПСЧ (в режиме OFB или Counter) в блоке генерации. Безопасность комплекса определяется в первую очередь безопасностью этих криптографических примитивов, а также отсутствием уязвимостей программной реализации используемых алгоритмов.

Ключ, используемый функцией зашифрования блочного криптографического преобразования, регулярно обновляется на основе текущего значения и информации, поступающей от внешних источников случайности (источников энтропии на рис. 8.4). Информация с выхода внешних источников случайности после преобразования в первом блоке хеширования собирается в накопителе. Задача блока хеширования – не дать противнику возможности предсказывать значения выборок из накопителей, используемых для обновления ключа. При выключении

питания желательно хранить часть внутреннего секретного состояния блока генерации или ключ в энергонезависимой памяти. Это позволит генератору при перезапуске оказаться в непредсказуемом состоянии. В промежутке между соседними обновлениями ключа устройство суть генератор гаммы поточного режима блочного шифра.

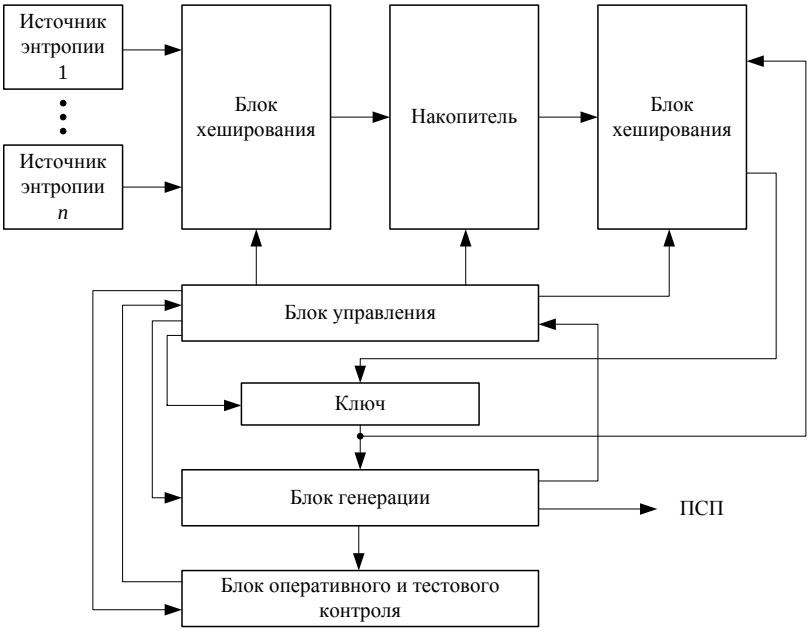


Рис. 8.4. Аппаратно-программный комплекс генерации ПСЧ, использующий внешние источники случайности

Синхронизацию работы всех блоков генератора, а также определение моментов инициирования процедуры обновления ключа обеспечивает блок управления. Процедура обновления инициируется при накоплении достаточного количества энтропии и возникновении одного из трех событий:

1) наступило время планового обновления ключа;
2) блок генерации оказался в «слабом» состоянии, из которого в течение длительного времени не может выйти самостоятельно;

3) обнаружены нарушения статистической безопасности выходной последовательности.

Функции блока оперативного и тестового контроля:

проверка работоспособности (методом сравнения с эталоном) устройства в заданные моменты времени (при включении питания, по запросу пользователя, периодическая и т.п.);

анализ статистической безопасности формируемых фрагментов выходной последовательности, в том числе анализ корреляции между соседними выборками;

анализ ключа и внутреннего состояния блока генерации на предмет попадания в пространство «слабых» значений;

информирование блока управления о проблемах в работе блока генерации (например, о необходимости обновления ключа).

Блочные генераторы ПСЧ следует признать лучшими по совокупности двух критериев – непредсказуемости и быстродействию.

Поточные генераторы ПСЧ, являясь самыми быстродействующим типом генераторов (в ущерб остальным характеристикам), применяются в тех случаях, когда необходима передача больших информационных потоков в реальном масштабе времени или близком к нему.

В отличие от блочных генераторов ПСЧ, чьи нелинейные функции строятся по итеративному принципу, при проектировании поточных генераторов используется огромное множество приемов и методов, классифицировать которые очень сложно. Можно выделить всё же следующие параметры:

способ управления синхронизацией;

механизм получения выходной последовательности;

тип используемых *S*- или *R*-блоков;

наличие или отсутствие каскадов.

Способ управления синхронизацией. По этому параметру генераторы делятся на две группы: работающие по принципу stop-and-go и не использующие управления синхронизацией. Классические примеры генераторов ПСЧ первой группы – генераторы ПСЧ A5 и PIKE.

Механизм получения выходной последовательности. По этому параметру генераторы можно разделить на две группы – соответственно использующие и не использующие для получения результирующей последовательности комбинирование нескольких ПСП. Классический пример первого подхода – перемешивание двух ПСП под управлением третьей. Примерами генераторов первого типа являются упомянутые выше генераторы A5 и PIKE, в которых комбинирование осуществляется с использованием операции сложения по модулю 2.

Тип испольуемых S- или R-блоков. По этому параметру генераторы можно разделить на четыре группы:

1) вообще без табличных преобразований;
2) использующие фиксированные несекретные таблицы;
3) использующие фиксированные секретные (ключевые или зависящие от ключа) таблицы;
4) использующие секретные таблицы, непрерывно изменяющиеся в процессе функционирования генератора.

Последний вариант следует признать наиболее предпочтительным, так как он делает лишенными особого смысла действия аналитика, связанные с восстановлением ключевой таблицы по имеющемуся фрагменту ПСП. Классическим примером такого подхода является один из лучших поточных генераторов ПСЧ, генератор RC4, разработанный Р. Ривестом.

Наличие или отсутствие каскадов. По этому параметру генераторы можно разделить на две группы – по отсутствию либо наличию каскадов соответственно. В последнем случае принцип построения качественного генератора ПСЧ суть каскадное включение нескольких генераторов относительно невысокого качества.

Генераторы ПСЧ на основе односторонних функций следует признать наиболее математически обоснованным. Тем не менее эти генераторы не нашли широкого распространения в первую очередь по причине чрезвычайно низкого быстродействия. Они работают в сотни раз медленнее блочных генераторов ПСЧ, которые сами считаются медленными.

Непредсказуемые (криптографически стойкие) генераторы ПСЧ могут быть построены на основе использования в качестве нелинейного преобразования F_{fb} или F_{out} *односторонних функ-*

ций. Справедливо следующее утверждение: *криптостойкие генераторы ПСЧ существуют тогда и только тогда, когда существуют односторонние функции* [3, 4].

8.4. Генераторы на регистрах сдвига с линейными обратными связями

В данном разделе описываются принципы построения наиболее эффективных (в первую очередь из-за простоты программной и аппаратной реализации и хороших статистических свойств) некриптографических генераторов ПСЧ. Используемый при их анализе математический аппарат – теория линейных последовательностных машин и теория конечных полей (полей Галуа).

Эти генераторы активно используются в качестве строительных блоков при построении поточных криптографических генераторов ПСП, помехоустойчивых кодов БЧХ, Рида–Соломона и других циклических кодов. Они обладают очень интересными свойствами, которые позволяют решать целый ряд специфических задач, связанных с обеспечением безопасности.

В работе [29] теория двоичных последовательных генераторов на регистрах сдвига с линейными обратными связями (РСЛОС) обобщается на случай формирования недвоичных (L-ричных) последовательностей. Рассматриваются теоретические основы построения двоичных параллельных генераторов ПСЧ, недвоичные генераторов последовательного и параллельного типа, анализируются их свойства. Описываются принципы построения нелинейных генераторов произвольной длины, в том числе максимально возможной при заданном числе элементов памяти генератора, генераторов с предпериодом, генераторов с самоконтролем, универсальных программируемых генераторов.

8.4.1. Последовательные РСЛОС

Последовательности, формируемые двоичными генераторами на основе регистров сдвига с линейными обратными связями – LFSR (Linear Feedback Shift Register), являются важнейшим классом ПСП. Основные достоинства этих генераторов:

простота программной и аппаратной реализации;

максимальное быстродействие;

хорошие статистические свойства формируемых последовательностей;

возможность построения на их основе генераторов, обладающих свойствами, ценными при решении специфических задач защиты информации (формирование последовательностей произвольной длины, формирование последовательностей с предпериодом, формирование ПСП с произвольным законом распределения, построение генераторов, обладающих свойством самоконтроля и т.п.).

РСЛОС, к сожалению, не являются непредсказуемыми, поэтому применяются при решении задач защиты компьютерных систем от умышленных деструктивных воздействий лишь в качестве строительных блоков.

Наиболее известные примеры использования РСЛОС и математического аппарата полей Галуа:

CRC-коды – идеальное средство контроля целостности информации при случайных искажениях информации;

реализация концепции самотестирования БИС и СБИС;

поточные алгоритмы A5, PANAMA, SOBER, SNOW и др.;

блочный алгоритм RIJNDAEL, принятый в 2001 г. в качестве Американского стандарта шифрования XXI века – AES.

Исходная информация для построения двоичного РСЛОС – так называемый образующий многочлен. Степень этого многочлена определяет разрядность регистра сдвига, а ненулевые коэффициенты – характер обратных связей.

В общем случае двоичному образующему многочлену степени N

$$\Phi(x) = \sum_{i=0}^{N} a_i x^i \,(\mathrm{mod}\, 2),$$

где $a_N = a_0 = 1,\ a_j \in \{0,\ 1\},\ j = \overline{1, (N-1)}$, соответствуют устройства, показанные на рис. 8.5,*а* (схема Фибоначчи) и 8.5,*б* (схема Галуа). Уравнения работы генераторов Фибоначчи и Галуа имеют вид соответственно

$$\begin{cases} q_1(t+1) = \sum_{i=1}^{N} a_i q_i(t); \\ q_j(t+1) = q_{j-1}(t), \\ j = \overline{2,\ N}; \end{cases} \tag{8.3}$$

$$\begin{cases} q_1(t+1) = a_n q_N(t); \\ q_j(t+1) = q_{j-1}(t) \oplus a_{N-i+1} q_N(t), \\ j = \overline{2,\ N}, \end{cases} \tag{8.4}$$

где $q_i(t)$ – состояние i-го разряда регистра в момент времени t, $i = \overline{1,\ N}$, или в матричной форме

$$Q(t + 1) = T \cdot Q(t),$$

где $Q(t) = \begin{vmatrix} q_1(t) \\ q_2(t) \\ ... \\ q_N(t) \end{vmatrix}$ – состояние генератора в момент времени t, а T – квадратная матрица порядка N вида (8.5) для генератора Фибоначчи или (8.6) для генератора Галуа

$$\begin{vmatrix} a_1 & a_2 & ... & a_{N-1} & a_N \\ 1 & 0 & ... & 0 & 0 \\ 0 & 1 & ... & 0 & 0 \\ & & ... & & \\ 0 & 0 & ... & 1 & 0 \end{vmatrix} \tag{8.5}$$

$$\begin{vmatrix} 0 & ... & 0 & 0 & a_N \\ 1 & ... & 0 & 0 & a_{N-1} \\ & ... & & & \\ 0 & ... & 1 & 0 & a_2 \\ 0 & ... & 0 & 1 & a_1 \end{vmatrix} \tag{8.6}$$

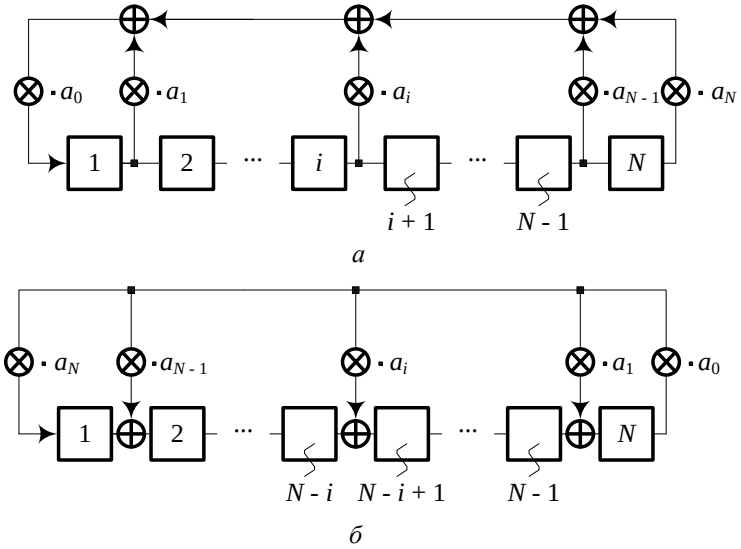


Рис. 8.5. Общий вид LFSR, соответствующих
$\Phi(x) = a_N x^N + a_{N-1}x^{N-1} + \ldots + a_2 x^2 + a_1 x + a_0$, $a_i \in \{0, 1\}$:
а – схема генератора Фибоначчи; *б* – схема генератора Галуа.
При $a_i = 1$ умножение на a_i равносильно наличию связи,
при $a_i = 0$ умножение на a_i равносильно отсутствию связи

8.4.2. Параллельные РСЛОС

Рассмотренные последовательные LFSR могут использоваться только для генерации битовых ПСП. Если необходима n-разрядная последовательность, можно предложить два возможных метода решения.

В первом случае выбираем образующий многочлен степени $N > n$ (еще лучше $N >> n$), выбираем схему Фибоначчи или Галуа и считываем очередной n-разрядный элемент ПСП с соседних разрядов регистра сдвига каждые n тактов работы LFSR. Недостатком такого решения очевидно является низкое быстродействие. Второй метод предполагает синтез схемы генератора, работающего в n раз быстрее исходного LFSR (иначе го-

воря, выполняющего за один такт своей работы преобразования, которые в исходном LFSR выполняются за n тактов). Второй вариант решения более эффективен с точки зрения затрат памяти и быстродействия при программной реализации в случае использования разреженного многочлена (многочлена с относительно небольшим числом ненулевых коэффициентов) $\Phi(x)$, в особенности когда образующий многочлен генератора Фибоначчи имеет вид $\Phi(x) = x^N + x^j + 1$, а i кратно n.

Исследуем свойства генератора многоразрядной ПСП. Общий вид генератора двоичных последовательностей, соответствующего уравнению

$$Q(t + 1) = T^k Q(t),$$

показан на рис. 8.6.

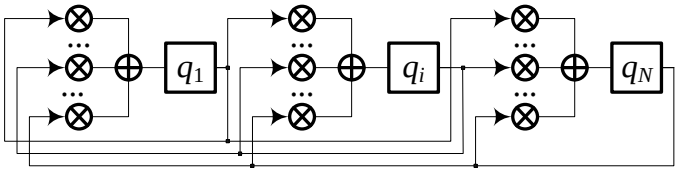


Рис. 8.6. Генератор двоичных последовательностей, соответствующий уравнению $Q(t + 1) = T^k Q(t)$

Величина, на которую происходит умножение в каждом блоке умножения (БУ), определяется соответствующим коэффициентом $a_{ij} \in \{0,1\}$ сопровождающей матрицы

$$V = T^k \, (i = \overline{1, N}, \, i = \overline{1, N}).$$

Если $a_{ij} = 0$, это эквивалентно отсутствию связи между выходом i-го разряда регистра генератора и входом j-го сумматора по модулю два. Если $a_{ij} = 1$, это эквивалентно наличию связи между выходом i-го разряда регистра генератора и входом j-го сумматора по модулю два.

Так как нулевое состояние всех элементов памяти генератора является запрещенным, максимально возможное число состояний устройства, а значит, и максимально возможная длина формируемой двоичной последовательности, снимаемой с выхода любого разряда, равны $2^N - 1$. В этом случае диаграмма состояний генератора состоит из одного тривиального цикла и цикла максимальной длины $2^N - 1$.

Многочлен $\Phi(x)$ степени N называется *примитивным*, если он не делит нацело ни один многочлен вида $x^S - 1$, где $S < 2^N - 1$. Примитивные многочлены существуют для любого N. Показателем многочлена $\Phi(x)$ называется наименьшее натуральное число e, при котором $x^e - 1$ делится на $\Phi(x)$ без остатка.

Пусть $\Phi(x)$ – примитивный многочлен степени N, тогда справедливо следующее утверждение.

Формируемая последовательность имеет максимальный период $S = 2^N - 1$ тогда и только тогда, когда наибольший общий делитель чисел S и k равен 1 (т.е. S и k взаимно просты).

При $k = 1$ примитивность $\Phi(x)$ является необходимым и достаточным условием получения последовательности максимальной длины.

Последовательность максимальной длины принято называть М-последовательностью, а формирующий ее генератор – генератором М-последовательности. Именно генераторы М-последовательностей обычно используются для формирования ПСП. Список примитивных многочленов приведен в приложениях 1 и 2.

Каждая матрица V имеет характеристический многочлен $\varphi(x)$, которым является определитель матрицы $V - xE$, т.е. $\varphi(x) = |V - xE|$, где E – единичная матрица. Многочлен $\Phi(x)$ определяет (образует) только структуру генератора, свойства же последнего зависят именно от $\varphi(x)$. Каждая квадратная матрица удовлетворяет своему характеристическому уравнению, т.е. $\varphi(V) = 0$. Характеристический и образующий многочлен генератора связаны следующим соотношением:

$$\varphi(x) = \Phi(x^{-1})x^N,$$

т.е. являются взаимно-обратными. Примитивность $\varphi(x)$ автоматически означает примитивность $\Phi(x)$ и наоборот.

Децимацией последовательности $\{q_i(t)\}$ *по индексу* k называется формирование новой последовательности $\{q^*_i(t)\}$, состоящей из k-х элементов $\{q_i(t)\}$, т.е. $q^*_i(t) = q_i(kt)$. Если период последовательности, полученной в результате децимации М-последовательности, равен максимальному, децимация называется собственной или нормальной.

Последовательность, снимаемая с выхода i-го разряда генератора, изображенного на рис. 8.6, является децимацией по индексу k последовательности, снимаемой с выхода i-го разряда генераторов, изображенных на рис. 8.5. Если Q – начальное состояние генератора, то последовательности состояний, в которых будут находиться устройства в следующие моменты времени, имеют вид $Q, QV, QV^2, QV^3, \ldots$ (для генератора, показанного на рис. 8.6) и $Q, QT, QT^2, QT^3, \ldots$ (для генераторов, показанных на рис. 8.5). Учитывая, что $V = T^k$, можно сделать вывод, что в генераторе, показанном на рис. 8.6, за один такт осуществляются преобразования, которые в генераторах, показанных на рис. 8.5, – за k тактов. Таким образом, генератор, показанный на рис. 8.6, в котором содержимое первых k разрядов (при $k \le N$) полностью обновляется в каждом такте, может использоваться для генерации k-разрядной ПСП, что нельзя сказать про генераторы, показанные на рис. 8.5, которые в тех же разрядах формируют лишь сдвинутые копии одной и той же двоичной последовательности.

Пусть $k = 1$, $F(x)$ – примитивный многочлен. Генераторы Галуа, образующий многочлен которых имеет вид

$$\Phi(x) = (x + 1)F(x)$$

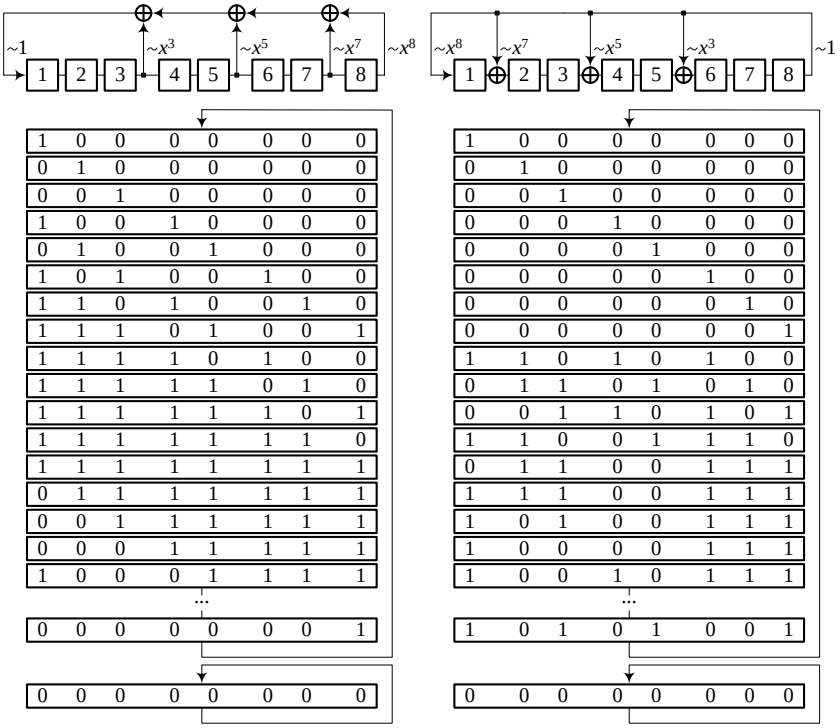
обладают свойством самоконтроля: при правильной работе генератора свертка по модулю два содержимого всех разрядов не меняет своего значения, иначе говоря, уравнение самоконтроля имеет вид

$$\sum_{i=1}^{N} q_i(t) \pmod 2 = \text{const}.$$

8.4.3. Примеры двоичных РСЛОС

Рассмотрим примеры практической реализации генераторов ПСЧ, рассмотренных в данном разделе.

Двоичные последовательные РСЛОС. Пусть образующий многочлен имеет вид $\Phi(x) = x^8 + x^7 + x^5 + x^3 + 1$. Этой ситуации соответствуют два генератора М-последовательности, показанные на рис. 8.7, *а* и *б*.



а *б*

Рис. 8.7. Два варианта построения LFSR, соответствующего образующему многочлену $\Phi(x) = x^8 + x^7 + x^5 + x^3 + 1$:
а – генератор Фибоначчи и его диаграмма состояний;
б – генератор Галуа и его диаграмма состояний

Двоичные параллельные РСЛОС. На рис. 8.8 и 8.9 показаны схемы двоичных параллельных генераторов, формирующих 8-разрядную и 32-разрядную ПСП длиной $2^{65} - 1$ соответственно. Рассматривается случай

$$N = 65,\ \Phi(x) = x^{65} + x^{32} + 1,\ T = T_1.$$

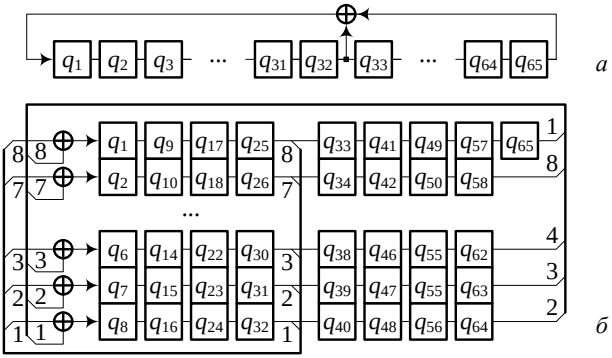


Рис. 8.8. Байтовый генератор ПСЧ:
а – битовый генератор Фибоначчи, соответствующий $\Phi(x) = x^{65} + x^{32} + 1$;
б – байтовый генератор Фибоначчи, соответствующий $\Phi(x) = x^{65} + x^{32} + 1$

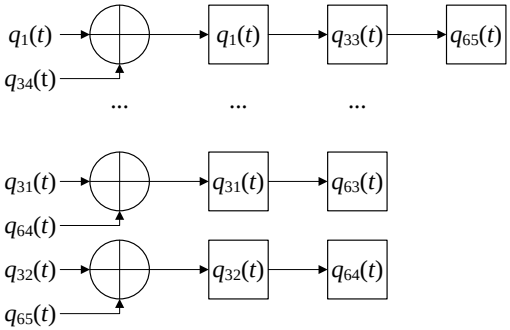


Рис. 8.9. Генератор двоичной М-последовательности,
соответствующий $\Phi(x) = x^{65} + x^{32} + 1$, $k = 32$, $T = T_1$

Программная реализация этого генератора на языке Ассемблера IBM PC в предположении, что

$$[q_{32}(t) \dots q_2(t)\ q_1(t)] = \text{EBX},$$
$$[q_{64}(t) \dots q_{34}(t)\ q_{33}(t)] = \text{EAX},$$
$$q_{65}(t) = \text{CF},$$

где q_i – состояние i-го разряда LFSR ($i = \overline{1, 65}$), потребует всего лишь трех команд на каждый такт работы устройства:

```
RCR     EAX
XCHG    EAX, EAX
XOR     EBX, EAX
```

8.4.4. Основы теории конечных полей

Рассматриваются основы теории полей Галуа, в том числе особенности построения конечных полей на основе двоичных непримитивных многочленов, а также схемотехнические основы реализации полей Галуа на основе регистров сдвига [1, 20, 24, 27, 29].

Основы теории недвоичных полей Галуа, необходимые для понимания принципов построения последовательных и параллельных недвоичных генераторов ПСЧ, изложены в [29].

Группа. *Группой G* называется непустое множество элеметов $\alpha, \beta, \gamma, \ldots,$ обладающее следующими свойствами:

1) для элементов множества *G* определена некоторая операция двух переменных, записываемая в виде $\alpha + \beta = \gamma$ или в виде $\alpha\beta = \gamma$ в первом случае операцию условно называют сложением, во втором – умножением;
2) на множестве *G* выполняются следующие законы:

   в результате применения операции к двум любым элементам группы также получается элемент группы (свойство замкнутости);

   для любых трех элементов группы справедливо $(\alpha + \beta) + \gamma = \alpha + (\beta + \gamma)$, если операция является сложением; или $(\alpha\beta)\gamma = \alpha(\beta\gamma)$, если операция является умножением (ассоциативный закон);

   в группе существует единичный элемент, который обозначается как 0 для сложения, при этом для любого элемента α группы справедливо $0 + \alpha = \alpha + 0 = \alpha$; либо как 1 для умножения, при этом для любого элемента α группы справедливо $1\,\alpha = \alpha\,1 = \alpha$;

   каждый элемент α группы обладает обратным элементом, который обозначается как $(-\alpha)$ для сложения, при этом $\alpha + (-\alpha) = (-\alpha) + \alpha = 0$; либо как α^{-1} для умножения, при этом $\alpha\,\alpha^{-1} = \alpha^{-1}\alpha = 1$.

Если для любых элементов α, β группы выполняется коммутативный закон, т.е. справедливо равенство $\alpha + \beta = \beta + \alpha$ для сложения или $\alpha\beta = \beta\alpha$ для умножения группа называется *ком-*

мутативной или *абелевой*.

Число элементов группы называется *порядком* группы. В дальнейшем будем рассматривать только конечные группы, т.е. группы, определенные на конечном множестве элементов.

Абелева группа относительно операции сложения называется *аддитивной* абелевой группой, абелева группа относительно операции умножения называется *мультипликативной* абелевой группой.

Некоторое подмножество группы G называется *подгруппой*, если оно удовлетворяет всем свойствам группы.

Конечная группа, которая состоит из степеней 1, g, g^2, g^3, … одного из ее элементов g, называется циклической группой. При этом наименьшее целое число e такое, что $g^e = 1$, называется *порядком элемента* g. Такие группы всегда коммутативны. Любая группа простого порядка и любая подгруппа циклической группы являются циклическими. Каждый элемент α любой группы G порождает в G циклическую подгруппу, состоящую из всех его степеней.

Конечное поле. *Конечным полем* или *полем Галуа* $GF(p)$ называется конечное множество из p (где $p = q^n$, q – простое, n – натуральное) элементов $\alpha, \beta, \gamma, \ldots$, обладающее следующими свойствами:

1) для элементов множества $GF(p)$ определены две операция двух переменных, записываемая в виде $\alpha + \beta = \gamma$ или в виде $\alpha\beta = \gamma$, первую операцию условно называют сложением, вторую – умножением;
2) на множестве $GF(p)$ выполняются следующие законы:

   в результате применения операции сложения или умножения к двум любым элементам группы также получается элемент группы (свойство замкнутости);

   для любых трех элементов для операции сложения справедливо $(\alpha+\beta)+\gamma = \alpha+(\beta+\gamma)$, для операции умножения справедливо $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ (ассоциативный закон);

   для любых трех элементов справедливо $(\alpha+\beta)\gamma = \alpha\gamma + \beta\gamma$;

в $GF(p)$ существует нулевой элемент, который обозначается как 0, при этом для любого элемента α поля справедливо $0 + \alpha = \alpha + 0 = \alpha$, $0\alpha = \alpha 0 = 0$;

в $GF(p)$ существует единичный элемент, который обозначается как 1, при этом для любого элемента α поля справедливо $1\,\alpha = \alpha\, 1 = \alpha$;

каждый элемент α конечного поля обладает обратными аддитивным и мультипликативным элементами, которые обозначаются как $(-\alpha)$ для сложения, при этом $\alpha + (-\alpha) = (-\alpha) + \alpha = 0$; либо как α^{-1} для умножения, при этом $\alpha\,\alpha^{-1} = \alpha^{-1}\alpha = 1$;

3) в $GF(p)$ все ненулевые элементы поля могут быть представлены в виде степени примитивного элемента ω, т.е. в виде ω^i. Таким образом, поле может быть записано в виде

$$GF(p) = \{ 0,\ \omega^0\ \omega^1\ \omega^2 \ldots\ \omega^{p-2} \},$$

где $\omega^0 = 1$, $\omega^1 = \omega$, $\omega^{p-1} = 1$, $\omega^p = \omega$ …, а множество Z^*_p всех ненулевых элементов поля $GF(p)$ образует абелеву группу порядка $p - 1$.

Простейшие поля получаются следующим образом. Пусть q – простое число. Тогда целые числа 0, 1, 2, ... , $(q - 1)$ образуют поле $GF(q)$, при этом операции сложения, вычитания, умножения и деления выполняются по модулю q. Более строго, $GF(q)$ – это поле классов вычетов по модулю q, т.е.

$$GF(q) = \{0, 1, 2, \ldots, (q - 1)\},$$

где через 0 обозначаются все числа, кратные q, через 1 – все числа, дающие при делении на q остаток 1, и т.д. С учетом этого вместо $q - 1$ можно писать -1. Утверждение $\alpha = \beta$ в $GF(q)$ означает, что $\alpha - \beta$ делится на q или что α сравнимо с β по модулю q, т.е.

$$\alpha \equiv \beta \pmod{p}.$$

Поле, содержащее $p = q^n$ элементов, где q – простое число, а n – натуральное, не может быть образовано из совокупности целых чисел по модулю p. Элементами поля $GF(q^n)$ являются все многочлены степени не более $n - 1$ с коэффициентами из поля $GF(q)$. Сложение в $GF(q^n)$ выполняется по обычным правилам сложения многочленов, при этом операции приведения подоб-

ных членов осуществляются по модулю q.

Многочлен с коэффициентами из $GF(q)$ (т.е. многочлен над полем $GF(q)$), не являющийся произведением двух многочленов меньшей степени, называется неприводимым.

Многочлен $\Phi(x)$ степени N с коэффициентами из $GF(p)$ называется неприводимым, если он не делится ни на один другой многочлен степени, меньшей N и большей 0.

Многочлен $\Phi(x)$ степени N с коэффициентами из $GF(p)$ называется примитивным, если он не делит нацело ни один многочлен вида $x^S - 1$, где $S < p^N - 1$. Примитивный многочлен автоматически является неприводимым.

Число примитивных многочленов степени N равно

$$\frac{\varphi\left(p^N - 1\right)}{N},$$

где $\varphi(n)$ – функция Эйлера (число натуральных чисел, меньших n и взаимно простых с n).

Выберем фиксированный неприводимый многочлен $\varphi(x)$ степени n. Тогда произведение двух элементов поля получается в результате их перемножения с последующим взятием остатка после деления на $\varphi(x)$. Таким образом, поле $GF(q^n)$ можно представить как поле классов эквивалентности многочленов над $GF(q)$. Два таких многочлена объявляются эквивалентными, если их разность делится на $\varphi(x)$. Конечные поля порядка q^n существуют для всех простых q и всех натуральных n.

Рассмотрим два возможных варианта: когда многочлен $\varphi(x)$ – примитивный и когда $\varphi(x)$ не является примитивным.

Пусть

$$q = 2,\ n = 4,\ \varphi(x) = x^4 + x + 1 -$$

примитивный над $GF(2)$. Элементы поля $GF(2^4)$ имеют вид

$$0,\ 1,\ x,\ x + 1,\ \ldots,\ x^3 + x^2 + x + 1.$$

Так как $\varphi(x)$ – примитивный, ему соответствует устройство, диаграмма состояний которого состоит из цикла максимальной длины $2^4 - 1$ и одного тривиального цикла, включающего состояние 0000, переходящее само в себя (рис. 8.10,*а*). Таким образом, в качестве ω можно взять корень $\varphi(x)$, а устройство, для которого $\varphi(x) = x^4 + x + 1$ является характеристическим многочленом,

объявить генератором ненулевых элементов поля. В результате соответствие между различными представлениями элементов поля имеет вид, представленный на рис. 8.10,*б*. Состояния генератора определяют список элементов $GF(2^4)$ в порядке возрастания степеней ω, т.е. один такт работы устройства, соответствующего характеристическому многочлену $\varphi(x)$, суть умножение текущего состояния устройства на ω, иначе говоря, на x по модулю $\varphi(x)$.

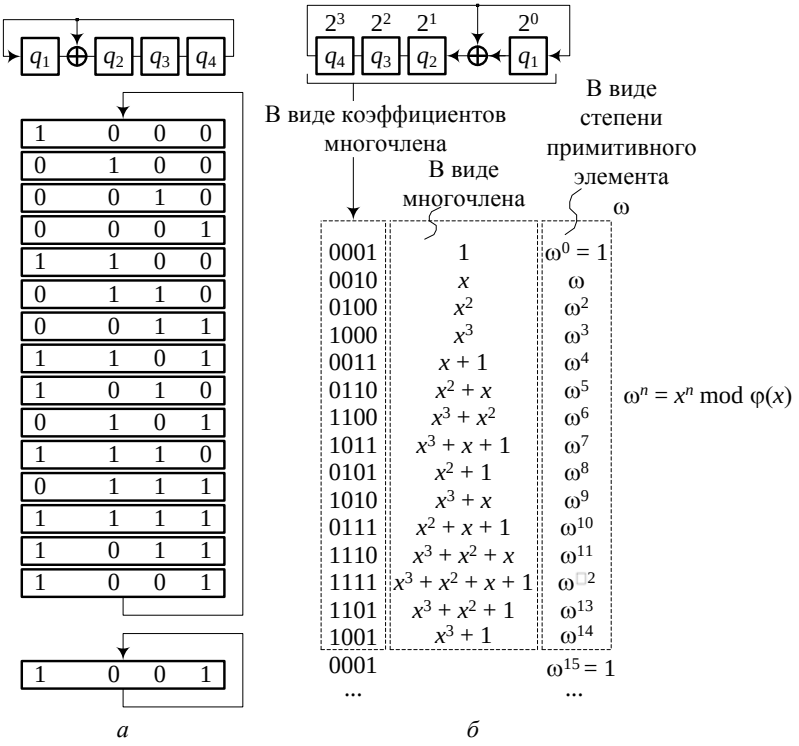


Рис. 8.10. Конечное поле $GF(2^4)$:
а – LFSR, соответствующий примитивному многочлену $\varphi(x) = x^4 + x + 1$ и его диаграмма состояний; *б* – соответствие между различными формами представления элементов поля

Пусть

$$q = 2,\ n = 4,\ \varphi(x) = x^4 + x^3 + x^2 + x + 1 -$$

неприводимый над $GF(2)$.

На рис. 8.11,*а* показан LFSR, соответствующий характеристическому многочлену $\varphi(x) = x^4 + x^3 + x^2 + x + 1$. Его диаграмма состояний состоит из трех циклов длиной 5 и одного тривиального цикла, а значит, LFSR, один такт которого суть умножение на x по модулю $\varphi(x)$ не может использоваться в качестве генератора элементов поля. Такая ситуация всегда имеет место, когда $\varphi(x)$ не является примитивным. Определим структуру устройства (генератора элементов поля), позволяющего сопоставить каждому ненулевому элементу поля соответствующую степень примитивного элемента.

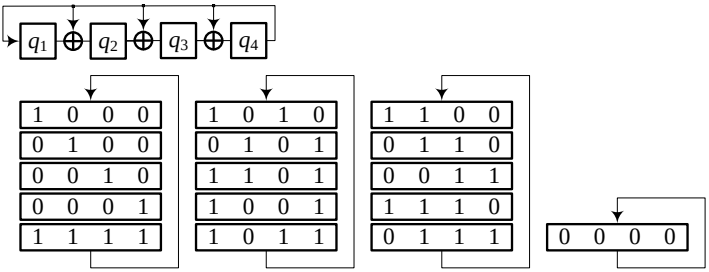
Имеем

$$\varphi(x + 1) = (x + 1)^4 + (x + 1)^3 + (x + 1)^2 + (x + 1) + 1 =$$
$$= x^4 + x^3 + 1 = \psi(x),$$

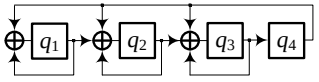
$\psi(x)$ – примитивный многочлен, а значит, соответствие между различными формами представления элементов $GF(2^4)$ можно получить, моделируя работу устройства, показанного на рис. 8.11,*б*. Один такт его работы суть умножение на $\omega = x + 1$.

8.5. Аддитивные генераторы ПСЧ

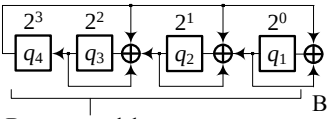
Не имеет себе равных с точки зрения производительности разновидность конгруэнтных генераторов ПСЧ, называемая аддитивным генератором. Самостоятельного значения эти генераторы в силу низкого качества формируемых последовательностей не имеют, но могут использоваться в качестве строительных блоков при создании стойких генераторов ПСЧ, а самое главное – служить в качестве эталона для оценки эффективности программной реализации других генераторов.



а



б



В виде коэффициентов многочлена	В виде многочлена	В виде степени примитивного элемента ω
0001	1	$\omega^0 = 1$
0011	$x + 1$	ω
0101	$x^2 + 1$	ω^2
1111	$x^3 + x^2 + x + 1$	ω^3
1110	$x^3 + x^2 + x$	ω^4
1101	$x^3 + x^2 + 1$	ω^5
1000	x^3	ω^6
0111	$x^2 + x + 1$	ω^7
1001	$x^3 + 1$	ω^8
0100	$x^2 + 1$	ω^9
1100	$x^3 + x^2$	ω^{10}
1011	$x^3 + x + 1$	ω^{11}
0010	x	ω^{12}
0110	$x^2 + x$	ω^{13}
1010	$x^3 + x$	ω^{14}
0001		$\omega^{15} = 1$
...		...

$\omega^n = (x + 1)^n \bmod \varphi(x)$

в

Рис. 8.11. Конечное поле $GF(2^4)$:
а – LFSR, соответствующий не примитивному неприводимому многочлену $\varphi(x) = x^4 + x^3 + x^2 + x + 1$ и его диаграмма состояний; *б* – LFSR, такт работы которого суть умножение на $x + 1$ по модулю $\varphi(x)$; *в* – соответствие между различными формами представления элементов поля

Аддитивный генератор состоит из N регистров разрядностью M каждый и сумматора по модулю 2^M. Начальным заполнением (ключом) генератора является массив

$$Q_1(0)\,Q_2(0)\ldots Q_N(0)$$

M-битовых слов. Уравнения работы аддитивного генератора Фибоначчи имеют вид

$$Q_1(t+1)=\sum_{i=1}^{N} a_i\,Q(t) \bmod 2^M,$$

$$Q_j(t+1)=Q_{j-1}(t),\ j=\overline{2,\ N},$$

где $Q_i(t)$ – состояние i-го регистра в момент времени t, а a_i – коэффициенты многочлена $\Phi(x)$ степени N, примитивного над $GF(2)$. Начальное заполнение выбирается таким образом, чтобы хотя бы в одном из регистров младший бит содержал «1». В этом случае младшие биты регистров образуют генератор двоичной М-последовательности. Учитывая, что при большом числе ненулевых коэффициентов $\Phi(x)$ быстродействие схемы снижается, возможна модификация схемы генератора с распределением двухвходовых блоков сложения по модулю 2^M между регистрами по схеме Галуа. На примере аддитивного генератора обсудим программную реализацию линейных генераторов ПСЧ, рассмотренных ранее, и нелинейных генераторов ПСЧ, которые будут рассмотрены в последующих разделах. Предлагаемый прием программирования самой производительной схемы генерации ПСЧ, суть которого – использование «плавающих» обратных связей аддитивного генератора, может использоваться и при программной реализации (в случае разреженных многочленах обратной связи) двоичных параллельных и недвоичных генераторов ПСЧ, функционирующих в поле $GF(p)$; а также генераторов ПСЧ со стохастическими сумматорами в цепи обратной связи. Последние, как будет показано ниже, отличаются от аддитивных генераторов лишь заменой сумматора по модулю 2^M на R-блок.

На рис. 8.12 показан пример аддитивного генератора для случая, когда $\Phi(x)=x^9+x^4+1$. Генератор формирует рекуррентную

последовательность $\{Q_i\}$ в соответствии с формулой

$$Q_i = (Q_{i-9} + Q_{i-4}) \bmod 2^M.$$

Можно предложить два способа программной реализации аддитивного генератора Фибоначчи. Первый заключается в реализации схемы, показанной на рис. 8.12, когда обратные связи зафиксированы и происходит сдвиг содержимого регистров.

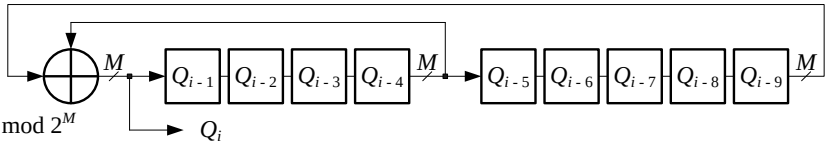


Рис. 8.12. Пример аддитивного генератора Фибоначчи

Второй способ предполагает фиксацию содержимого тех регистров, которые не являются приемниками сигнала обратной связи, «двигаются» лишь отводы обратной связи. Очевидно, что эффективность данного алгоритма возрастает с ростом степени N образующего многочлена.

8.6. Стохастические сумматоры. *R*-блоки

Эффективным строительным блоком при построении генераторов ПСЧ является *блок стохастического преобразования информации*.

В качестве одного из алгоритмов нелинейного преобразования элементов x_i n-разрядной информационной последовательности

$$x = x_1x_2x_3 \ldots x_i \ldots x_m, x_i \in GF(2^n),$$

длиной m под управлением ключевой k-разрядной последовательности

$$\gamma = \gamma_1\gamma_2\gamma_3 \ldots \gamma_i \ldots \gamma_m, \gamma_i \in GF(2^k),$$

такой же длины и качественного генератора ПСЧ с числом состояний M, $M \geq 2^n$, и начальным состоянием Q_0 предлагается следующий (рис. 8.13). Для каждого элемента x_i $\left(i = \overline{1, m}\right)$ повто-

ряем нижеприведенную последовательность действий:

загружаем очередной элемент x_i входной последовательности в память генератора ПСЧ;

выполняем γ_i тактов работы генератора;

часть ($q = q_{1i}\, q_{2i} \ldots x_{n'i}$, $n' \le L$) состояния $Q_i = (q_{1i}\, q_{2i} \ldots q_{Li})$ элементов памяти генератора после γ_i тактов работы объявляем результатом y_i преобразования элемента x_i.

После преобразования всех элементов исходной последовательности будет получена результирующая последовательность

$$y = y_1 y_2 y_3 \ldots y_i \ldots y_m,\ y_i \in GF(2^{n'}),$$

длиной m, для каждого элемента которой справедливо

$$y_i = R(x_i, \gamma_i).$$

Данное преобразование может эффективно использоваться для решения различных задач, связанных с защитой информации.

Схема одного из возможных простейших вариантов построения блока R стохастического преобразования и его условное графическое обозначение показаны соответственно на рис. 8.14 и 8.15.

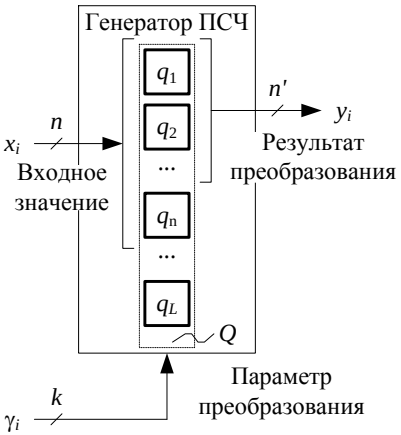


Рис. 8.13. Схема стохастического преобразования

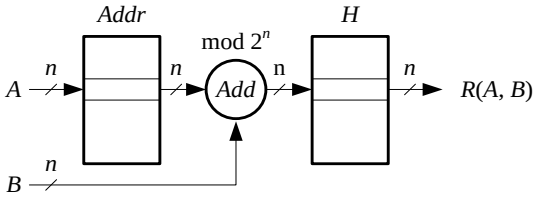


Рис. 8.14. Логика работы *R*-блока

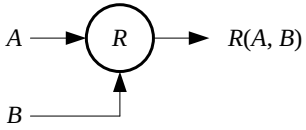


Рис. 8.15. Условное графическое обозначение *R*-блока

Ключевая информация *R-блока* – заполнение таблицы

$$H = \{H(m)\},\ m = \overline{0, (2^n - 1)}$$

размерности $n \times 2^n$, содержащей элементы $GF(2^n)$, перемешанные случайным образом, т.е. $H(m) \in GF(2^n)$. Результат $R_H(A, B)$ преобразования входного n-разрядного двоичного набора A зависит от заполнения таблицы H и параметра преобразования B, задающего смещение в таблице относительно ячейки, содержащей значение A, следующим образом:

$$R_H(A, B) = H((m_A + B) \bmod 2^n),$$

где m_A – адрес ячейки таблицы H, содержащей код A, т.е. $H(m_A) = A$. Другими словами результат работы *R*-блока суть считывание содержимого ячейки таблицы H, циклически смещенной на B позиций в сторону старших адресов относительно ячейки, содержащей код A. Для ускорения преобразования в состав *R*-блока вводится вспомогательный адресный массив

$$Addr = \{Addr(j)\}$$

размерности $n \times 2^n$, причем

$$\forall j = \overline{0, (2^n - 1)}\ \ Addr(j) = m_j.$$

Иными словами, ячейка с адресом j в массиве $Addr$ хранит адрес ячейки массива H, содержащей код j. Заслуживают внимания следующие факты:

в частном случае при $Addr = \{0, 1, 2, \ldots, (2^n - 1)\}$ и $B = 0$ получаем классический *S*-блок (блок замены) с таблицей замен *H*;

при записи в каждую ячейку массивов *H* и *Addr* ее собственного адреса получаем классический сумматор по модулю 2^n, а значит, с полным на то основанием *R*-блок может быть назван *стохастическим сумматором*, т.е. сумматором с непредсказуемым результатом работы, зависящим от заполнения ключевой таблицы *H*.

Ключевая информация, необходимая для работы *R*-блока, – содержимое таблицы *H* стохастического преобразования. В качестве алгоритма заполнения таблицы *H* может использоваться, например, алгоритм заполнения таблицы замен, специфицированный в криптоалгоритме RC4.

Возможен вариант использования *R*-блока, когда содержимое массива *H* (а значит, и содержимое массива *Addr*) зафиксировано, а ключевая информация подается на вход *B* параметра преобразования. В этой ситуации для обеспечения возможности вычисления результата преобразования «на лету» (без использования таблиц) в качестве содержимого массива *H* можно выбрать последовательные состояния генератора ПСЧ, который допускает эффективную программную реализацию.

В [29] предлагается схема нелинейного стохастического генератора ПСП, получившего название *RFSR* (Random Feedback Shift Register) (рис. 8.16). В состав RFSR входят *N* регистров Q_0, Q_1, …, Q_{N-1} разрядностью *n* каждый, *N* блоков стохастического преобразования R_1, R_2, …, R_N той же разрядности. Уравнения работы *RFSR* имеют вид

$$Q_0(t + 1) = R_N(Q_{N-1}(t), A(t)),$$

$$Q_i(t + 1) = R_i(Q_{i-1}(t), R_N(Q_{N-1}(t), A(t))), \; i = \overline{1, (N - 1)},$$

где $A(t)$ – значение на управляющем входе в момент времени *t*; $Q_j(t)$ и $Q_j(t + 1)$ – состояние *j*-го регистра соответственно в моменты времени *t* и $t + 1$, $j = \overline{0, (N - 1)}$. Выходная последовательность снимается с выхода одного из регистров. Оптимальное значение *n* равно 8, в этом случае размерность таблицы *H* стохастического преобразования равна 8×256. Ключевая информация – заполнение таблиц *H*, определяющих логику работы

R-блоков. В качестве вектора инициализации (синхропосылки) может использоваться начальное состояние регистров

$$Q_0(0)\ Q_1(0)\ \ldots\ Q_{N-1}(0).$$

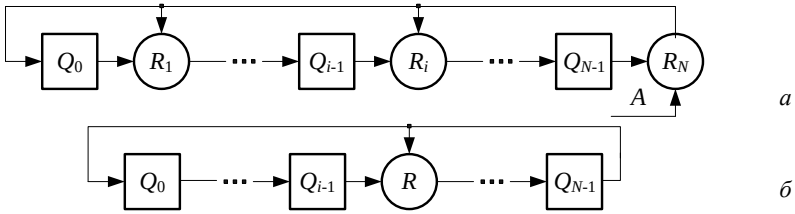


Рис. 8.16. Регистры сдвига со стохастической обратной связью:
а – общая схема RFSR; *б* – RFSR с одним *R*-блоком

8.6.1. Модификация существующих алгоритмов поточного шифрования

Криптостойкость существующих поточных шифров, использующих в своей работе блоки сложения по модулю 256 или линейные генераторы ПСЧ, можно увеличить простой их заменой соответственно на стохастические сумматоры (*R*-блоки) и RFSR, рассмотренные выше. В качестве примера на рис. 8.17 показан модифицированный генератор ПСЧ PIKE. На быстродействии алгоритмов такая замена скажется незначительно, учитывая главное достоинство стохастических генераторов ПСЧ – эффективную программную и аппаратную реализацию.

Можно выделить следующие перспективные направления использования блоков стохастического преобразования:

- модификация существующих поточных алгоритмов за счет замены сумматоров по модулю 256 на восьмиразрядные *R*-блоки или замены линейных генераторов ПСЧ на RFSR;
- использование прямого и обратного стохастических преобразований для выполнения операции гаммирования;
- реализация вероятностных поточных режимов использования блочных шифров;
- реализация поточных криптоалгоритмов и алгоритмов хеширования в тех приложениях, где требуется обеспечить их эффективную программную реализацию.

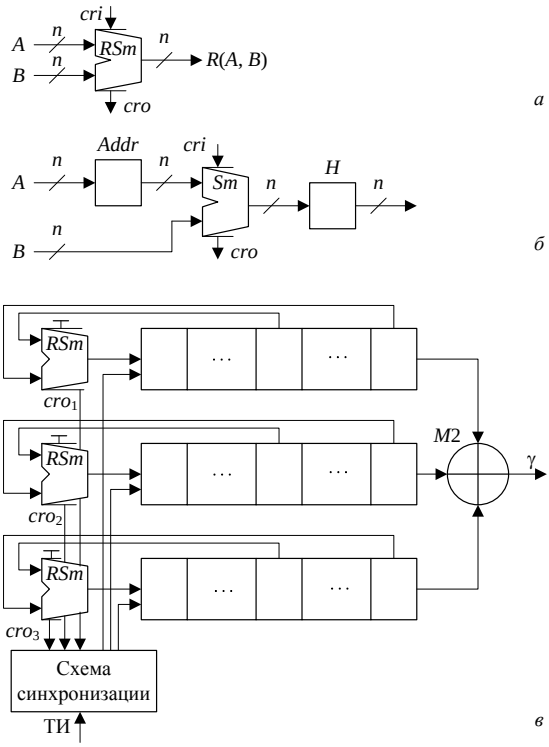


Рис. 8.17. Модифицированный вариант генератора ПСЧ PIKE:
а – УГО *R*-блока со входом (cri) и выходом (cro) переноса; *б* – схема *R*-блока со входом и выходом переноса; *в* – схема генератора ПСЧ

8.6.2. Нелинейные M-последовательности на основе R-блоков

Как отмечалось в [29], при соответствующем выборе таблицы стохастического преобразования выходная ПСП по сути – это нелинейная М-последовательность, т.е. последовательность максимальной длины, по своим статистическим свойствам превосходящая классическую М-последовательность с выхода LFSR той же разрядности и имеющая совершенно другую структуру (рис. 8.18). При этом все схемотехнические приемы, рассмотренные в [29] применительно к LFSR, работают и в случае RFSR.

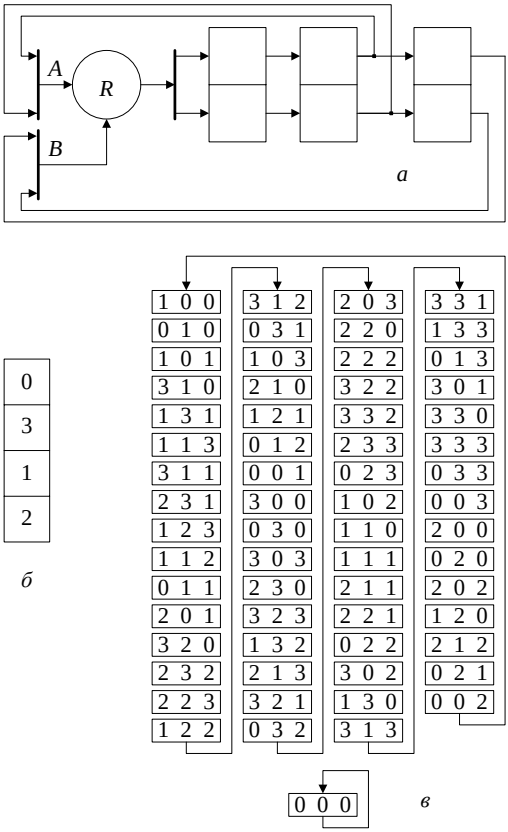


Рис. 8.18. Нелинейный генератор М-последовательности при $N = 3$:
а – схема генератора; *б* – таблица стохастического преобразования;
в – диаграмма переключений

Контрольные вопросы

1. Укажите функции генераторов ПСЧ в системах ОБИ.
2. Разработайте схему алгоритма одного такта работы РСЛОС.
3. Разработайте схему алгоритма одного такта работы аддитивного генератора.

4. Какие требования предъявляются к качественному генератору ПСЧ.
5. Сформулируйте принципы построения блочного генератора ПСЧ.
6. Перечислите основные строительные блоки, используемые при построении поточных генераторов ПСЧ.
7. Приведите примеры стохастических методов защиты информации.
8. Сформулируйте требования к статистически безопасному генератору ПСЧ.

ГЛАВА 9. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ СТОХАСТИЧЕСКИХ МЕТОДОВ В ЗАДАЧАХ ЗАЩИТЫ ИНФОРМАЦИИ В ЭЛЕКТРОННЫХ ПЛАТЕЖНЫХ СИСТЕМАХ

Электронная коммерция (ЭК) – это разновидность коммерческой деятельности, в которой взаимодействие между ее участниками на всех или некоторых ее этапах осуществляется электронным способом. Иначе говоря, электронная коммерция предполагает взаимодействие между партнерами с использованием информационных технологий (в первую очередь сетевых), что существенно повышает гибкость, эффективность и масштабность бизнес процессов. ЭК одна из двух базовых составляющих электронного бизнеса. Электронный бизнес – совокупность технологии поддержания внешних бизнес-контактов (электронной коммерции) и комплексной автоматизации внутренней деятельности компании.

Развитие систем электронного бизнеса невозможно без решения следующих задач защиты информации:

аутентификации участников информационного взаимодействия;

обеспечения конфиденциальности и целостности коммерческой информации (документов, удостоверяющих факт сделки, платежных документов, счетов, заказов и пр.) при ее передаче по каналам связи;

обеспечения невозможности отказа от факта получения какого-либо сообщения;

обеспечения юридической значимости пересылаемых электронных документов.

Решение указанных задач невозможно без применения стохастических методов (в первую очередь, протоколов электронной подписи, инфраструктуры открытых ключей, рассмотренных в предыдущих главах).

Электронная платежная система (ЭПС) – аналог традиционной платежной системы, обеспечивающий денежные расчеты между поставщиками и потребителями в электронном виде (без шелеста купюр, рукописных подписей и пр.).

Участники электронной платежной системы:

банки, объединенные договорными обязательствами;
предприятия торговли и сервиса, образующие сеть точек обслуживания клиентов;
процессинговые центры;
держатели платежных средств.

Кроме перечисленных выше задач ОБИ, которые решаются традиционными методами, существуют также задачи, актуальные именно при построении электронных платежных систем. В первую очередь речь идет об анонимности и неотслеживаемости электронных платежей.

При осуществлении оплаты за товар необходимо обеспечить конфиденциальность платежных данных потребителя, платежная информация (номер пластиковой карточки, номер счета и т. п.) должна быть известны только тому, кто имеет законное право ее знать (например, банку-эмитенту платежного средства). Безопасная транзакция требует, чтобы поставщик не знал платежные данные потребителя.

Проблема конфиденциальности тесто связана с проблемой анонимности потребителя при осуществлении коммерческой транзакции. Анонимность потребителя включает в себя анонимность платежа и анонимность взаимодействия. Анонимность платежа предполагает отсутствие взаимосвязи между платежом и личностью инициирующего его потребителя. Неотслеживаемость платежей означает, что два платежа, совершенные одним и тем же потребителем, не могут быть соотнесены друг с другом ни при каких условиях.

9.1. Цифровые деньги

Уже достаточно давно банки и другие коммерческие структуры используют при проведении деловых операций электронный обмен данными EDI (Electronic Data Interchange) и электронный перевод денежных средств EFT (Electronic Funds Transfer). В современных платежных системах весь процесс от начала до конца происходит в электронной (цифровой) форме. При этом для обеспечения безопасности и признания законности (конфиденциальности пересылаемых электронных документов, аутентификации участников информационного обмена) повсеместно при-

меняются криптосис-темы с открытым ключом для шифрования и формирования электронной подписи.

Учитывая, что традиционные денежные купюры суть не что иное как защищенный подделки документ логичным представляется переход к использованию *цифровых денег*. Защиту от подделки при этом может обеспечить электронная подпись банка, которая очевидно имеет большую надежность, чем традиционные водяные знаки, металлические полосы и т.п.

При осуществлении платежей с помощью кредитной карточки всегда, когда владелец карточки с ее помощью делает покупки, оплачивает налоги и т. д., где-то в базе данных делается отметка об этом событии. Соединив вместе эти в отдельности малозначимые данные, можно собрать большое количество информации об отдельном человеке. При этом последний не в состоянии выяснить, кому и что известно о его частной жизни; не в состоянии управлять точностью данной информации или определять, кто может ее получать. Необходимо реализовать такую систему доступа к ресурсам и услугам, в которой одновременно с решением задач идентификации, аутентификации и авторизации претендента решена задача обеспечения *анонимности* последнего. Обычные бумажные деньги обеспечивают все эти свойства. Если запрашиваемый ресурс – какой-либо товар, то наличие у покупателя достаточного количества купюр является доказательством его права на доступ к ресурсу. Хотя каждая из купюр имеет уникальный номер, отслеживать купюры по номерам практически невозможно. Итак, проблема состоит в том, чтобы сделать электронные платежи в такой же степени анонимными, как и расчет с помощью обычных денег. Иначе говоря, возникает задача обеспечения *неотслеживаемости* электронных документов и, в частности, цифровых денег.

«Отцом» цифровых денег с полным основанием можно назвать Д. Чаума, основателя и исполнительного директора фирмы DigiCash и одновременно признанного специалиста в области криптографии. DigiCash разработала и запатентовала криптографическую технологию безопасных электронных платежей.

Выглядит цифровая купюра приблизительно так, как показано на рис. 9.1. Документ содержит номинал и номер купюры, а также подпись банка-эмитента, которая получена на его секретном

ключе. При этом *электронная подпись надежно защищает купюру от подделки, но совершенно не защищает от копирования.* Чтобы избежать превращения цифровой купюры в «неразменный пятак» (см. А. Стругацкий, Б. Стругацкий «Понедельник начинается в субботу»), банк-эмитент должен контролировать каждую сделку.

Рис.9.1. Цифровая купюра

Рассмотрим возможную процедуру осуществления платежей с использованием цифровых денег (рис. 9.2).

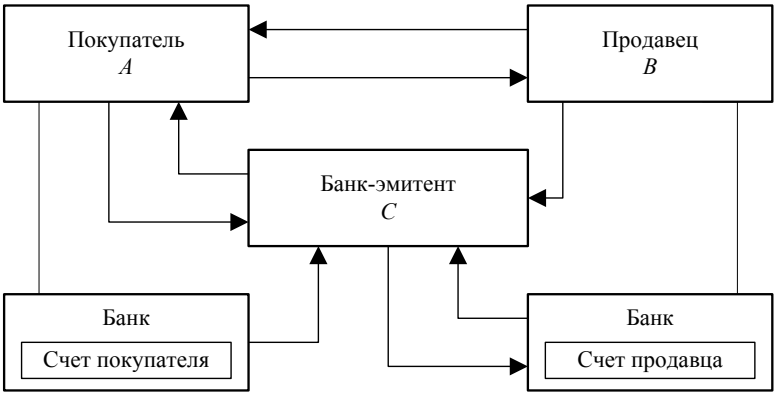


Рис. 9.2. Схема расчета цифровыми деньгами.
Покупатель перечисляет деньги в банк-эмитент либо напрямую, либо через банк – участник системы. Взамен покупатель получает цифровые деньги. Получив их при оплате товара, продавец, проверив их подлинность, отдает товар, перечисляет цифровые деньги банку-эмитенту, а тот переводит обычные денежные средства на счет продавца

Клиент *A* (будущий покупатель), желающий получить определенную сумму цифровых денег, посылает в банк-эмитент, в котором у него имеется счет, «полуфабрикат» цифровой купюры, имеющий вид, аналогичный показанному на рис. 9.1. Только подписан этот документ самим клиентом *A*. Так как он является клиентом банка, там знают его открытый ключ, а значит, могут проверить подпись. Убедившись, что именно *A* заказал цифровые деньги, банк удаляет его подпись, ставит свою, вычитает со счета *A* сумму, равную номиналу купюры, и отсылает последнюю *A*.

Получив электронную купюру *A* может потратить ее сам, переслав (или передав) ее в обмен на товар продавцу B_1 в магазине, принимающим цифровые деньги; либо переслать (или передать) ее другому субъекту B_2. Чтобы осуществить процесс передачи цифровой купюры участники обмена *A* и *B*, а также банк-эмитент *C* должны одновременно находиться на связи. Перед передачей купюры *B* участник *A* подписывает ее своим секретным ключом. Получив купюру, *B* проверяет подпись, а затем, убедившись в ее подлинности, удаляет ее и ставит свою, после чего отправляет в банк-эмитент. Банк проверяет, не получал ли он уже эту банкноту (защита от копирования!), т.е. ищет ее номер в специальном списке купюр, предъявленных к оплате ранее. Если банкнотой никто раньше не пользовался, ее номер заносят в список использованных купюр (второй раз ее к оплате не примут!) и переводят сумму, равную номиналу цифровой купюры, на счет своего клиента *B*.

Рассмотренная схема пока не обеспечивает неотслеживаемости электронных платежных средств, так как позволяет проследить за движением денег от *A* к *B*. Когда клиент *A* присылает заявку на цифровую купюру достоинством допустим 100 у.е., банк узнает номер купюры. В результате, когда эту купюру предъявляет к оплате клиент *B*, банк узнает, что *A* заплатил *B* 100 у.е. Таким образом, при данной схеме не обеспечивается анонимность платежей, банк в состоянии составить полное досье на любого своего клиента: кто, сколько и кому платил, сколько и от кого получал.

Для того чтобы сделать электронную купюру эквивалентной обычной бумажной того же номинала, Д. Чаум предложил *про-*

токол слепой подписи (blind signature). Заказывая цифровую купюру, клиент *A* создает «полуфабрикат» купюры, в которой указывает номинал и серийный номер купюры. Затем «затемняет» номер и посылает «полуфабрикат» купюры банку. Банк подписывает его и возвращает его *A*. Получив подписанный банком «полуфабрикат», *A* снимает «затемнение» и получает полноценную купюру, формат которой соответствует показанному на рис. 9.1. Если впоследствии эта купюра будет предъявлена банку, банк вынужден принять ее, так как на ней стоит его подпись. Банк так же, как и раньше, заносит ее номер в список купюр, предъявленных к оплате, но у него нет возможности узнать, от кого получена эта купюра. Когда банк ее подписывал, он не мог видеть ее номер.

Рассмотренная система платежей, требующая участия банка во всех транзакциях, называется *централизованной*. В отличие от последней *автономная система платежей* предполагает, что продавец *B* сам, без обращения к банку *C*, проверяет подлинность предъявленной покупателем *A* цифровой наличности. Понятно, что в этом случае банк идет на определенный риск, так как используемые схемы обеспечивают обнаружение злоупотреблений со стороны *A* постфактум. Основная идея соответствующих протоколов – однозначно идентифицировать нарушителя.

9.2. Защита информации в ЭПС на основе цифровых денег

Цифровые деньги (цифровая наличность) – защищенное от подделки электронное платежное средство. Более того, это единственное платежное средство, обеспечивающее анонимность и неотслеживаемость платежей. Ни одно из множества других электронных платежных средств (платежные карты, электронные чеки и пр.) такими свойствами не обладает.

Проблемы информационной безопасности, которые необходимо решить:

как защититься от кражи купюры (задача защиты прав собственника информации);

как защититься от повторного использования купюры, учитывая, что электронный документ и ЭЦП можно копировать сколь угодно много раз;

как защититься от подделки номинала цифровой купюры, учитывая, что ЭЦП банка-эмитента ставится только на номере купюры;

как обеспечить анонимность и неотслеживаемость платежей, иначе говоря, как получить полный электронный аналог бумажных денег, обладающих такими свойствами.

В табл. 9.1 приведены методы решения, из таблицы видно, что три из четырех задач решаются стохастическими методами.

Таблица 9.1

Методы защиты

Задача	Механизм решения	Участник платежной системы, обеспечивающий решение
Защита прав владельца цифровой купюры	Стохастический метод решения – хеширование случайного прекурсора для получения номера цифровой купюры	Будущий владелец купюры
Защита от повторного использования цифровой купюры	Поддержание списка номеров ранее использованных цифровых купюр и его анализ при авторизации	Банк
Защита от подделки номинала цифровой купюры	Стохастический метод решения – использование для каждого возможного номинала своей пары ключей формирования и проверки ЭЦП	Банк-эмитент
Обеспечение анонимности и неотслеживаемости платежей	Стохастический метод решения – схема слепой электронной подписи	Будущий владелец купюры, банк-эмитент

9.3. Слепая электронная подпись

Итак, целями абонента *A*, инициатора протокола, является, во-первых, формирование серийного номера *S* цифровой купюры, во-вторых, получение цифровой подписи абонента *C* (банка) на документе, в качестве которого в рассматриваемой ситуации выступает *S*, таким образом, чтобы абонент *C* не узнал содержи-

мого документа. Пусть $N = pq$, где p и q – два больших различных простых числа; e – открытый ключ (ключ проверки подписи), а d – закрытый ключ (ключ формирования подписи) абонента C, при этом

$$ed \equiv 1 \bmod [(p-1)(q-1)].$$

В результате для любого положительного $x < N$ справедливо $x^{ed} \bmod N = x$. Пусть $H(x)$ – общеизвестная хеш-функция.

Схема слепой электронной подписи (рис. 9.3)

1. Абонент A формирует случайное число S', называемое прекурсором (precursor).
2. Хешируя прекурсор, A формирует серийный номер купюры $S = H(S')$. Такая последовательность получения S необходима для защиты прав владельца будущей купюры. Так как только он в силу свойств функции $H(x)$ в случае возникновения споров может предъявить арбитру прекурсор, на основе которого сформирован серийный номер.
3. Абонент A формирует случайное число R – затемняющий множитель, единственное требование к которому – существование обратного $R^{-1} \bmod N$.
4. A шифрует затемняющий множитель на открытом ключе абонента C, умножает результат на S и посылает получившееся сообщение
$$y_A = S \times R^e \bmod N$$
абонента C.
5. Абонент C, получив сообщение y_A подписывает его на своем секретном ключе.
6. Затем посылает сформированное сообщение
$$y_C = \left(S \times R^e \bmod N\right)^d \bmod N =$$
$$= S^d \times R^{ed} \bmod N = S^d \times R \bmod N$$
обратно абоненту A.
7. Абонент A снимает действие затемняющего множителя, вычисляя
$$y_C \times R^{-1} \bmod N = S^d \bmod N,$$

и получает в результате серийный номер, подписанный *C*, при этом абонент *C* ничего не узнал о содержимом подписанного им документа.

9.4. Структура централизованной платежной системы на основе цифровой наличности. Анализ жизненного цикла цифровой купюры

Рассмотрим централизованную (on-line) платежную систему на основе цифровой наличности. Проанализируем жизненный цикл цифровой купюры на примере ситуации (рис. 9.4), когда покупатель (абонент *A*) и продавец (абонент *B*) – клиенты некоего банка (абонент *C*). Этот же банк является эмитентом цифровой наличности.

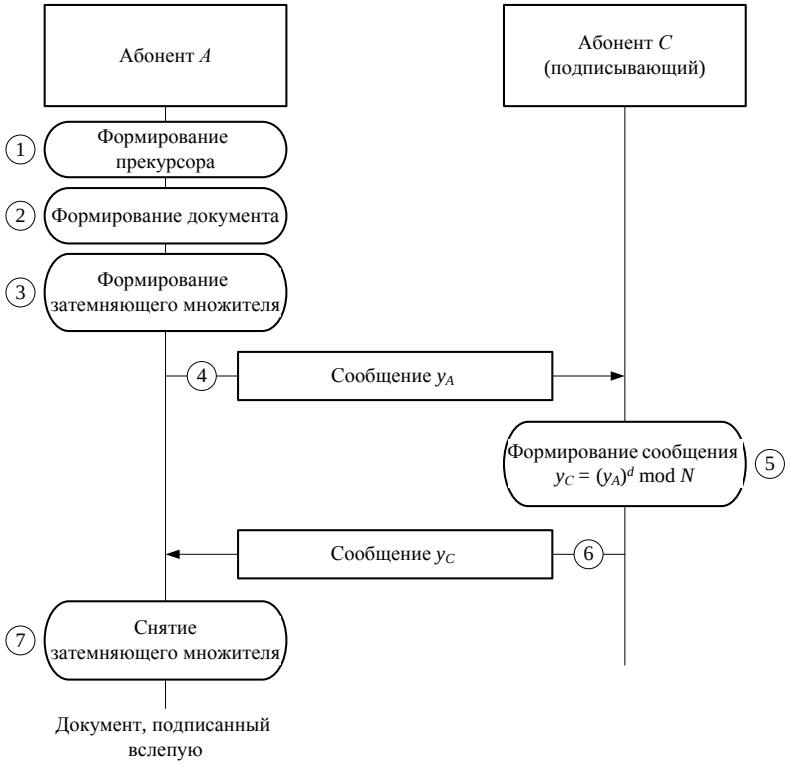


Рис. 9.3. Протокол слепой ЭЦП RSA

Предположим, стоимость представлена 4-разрядным двоичным кодом. В этом случае номинал купюры может принимать любое значение от 1 до 15. Каждому разряду соответствует одно из простых чисел, делителей банковской открытой экспоненты RSA. Пусть для представления стоимости, равной 1, используется экспонента 3. Аналогично стоимости, равные 2, 4 и 8, соотносятся с открытыми экспонентами, равными соответственно 5, 7 и 11.

Банковская открытая экспонента для представления купюры максимального достоинства, таким образом, равна

$$h = 11 \times 7 \times 5 \times 3.$$

Пусть клиент хочет приобрести две цифровые купюры, каждая достоинством 15 условных единиц. Каждая купюра имеет свой собственный серийный номер, соответственно S_1 и S_2. Соответствующие затемняющие множители равны R_1 и R_2.

Рассмотрим случай, когда абонент A после получения цифровых купюр совершает две покупки – одну на сумму 10 у.е., а другую – на сумму 12 у.е. [50].

Транзакция снятия со счета

Шаг 1. Абонент A с помощью генератора ПСЧ формирует два прекурсора $S_1^{'}$ и $S_2^{'}$, осуществляет их хеширование с целью получения номеров $S_1 = H(S_1)$ и $S_2 = H(S_2)$ двух цифровых купюр. Абонент A с помощью генератора ПСЧ формирует два затемняющих множителя R_1 и R_2,

Шаг 2. Абонент A формирует «полуфабрикаты» купюр, скрывая их номера с использованием затемняющих множителей, и посылает их в банк для простановки слепой подписи. Сообщение, посылаемое в банк, суть конкатенация двух запросов:

$$\left(S_1 \times R_1^h\right) \bmod N \,\|\, \left(S_2 \times R_2^h\right) \bmod N,$$

где $N = pq$, p и q – различные простые числа.

Шаг 3. Банк (абонент C) снимает со счета абонента A соответствующую сумму (30 у.е.), подписывает вслепую купюры и затем возвращает составное сообщение

$$\left(\left(S_1 \times R_1^h\right)^{1/h}\right) \bmod N \parallel \left(\left(S_2 \times R_2^h\right)^{1/h}\right) \bmod N =$$
$$= \left(S_1^{1/h} \times R_1\right) \bmod N \parallel \left(S_2^{1/h} \times R_2\right) \bmod N.$$

Инверсия $1/h$ суть банковская секретная экспонента (секретный ключ SK_C), которая вычисляется по формуле

$$h \times (1/h) = 1 \ mod \ [(p - 1)(q - 1)].$$

Эта экспонента так же, как и открытая, определяет общую стоимость купюры.

Шаг 4. Абонент *A* снимает действие затемняющих множителей и получает две полноценные цифровые купюры достоинством 15 у.е., подписанные банком-эмитентом, соответственно $S_1^{1/h} \bmod N$ и $S_2^{1/h} \bmod N$.

Транзакция покупки 1

Шаг 5. При выполнении платежа первой купюрой за товар (или услугу) стоимостью 10 единиц (двоичное представление 1010), абонент *A* использует экспоненту 7×3 в своем запросе. Он составляет сообщение для отправки абоненту *B* (продавцу) в следующем виде

$$S_1^{1/h(7\times3)} \bmod N = S_1^{1/(11\times5)} \bmod N.$$

Для получения «сдачи» абонент *A* становится неотслеживаемым кредитором для банка *C*. Абоненту *A* необходимо удержать остаток в 5 у.е. Для этого он формирует число *T* (по сути, вторую цифровую купюру) и скрывает его с помощью затемняющего множителя R_{A1}, зашифрованного с использованием экспоненты 7×3, соответствующей сумме в 5 у.е.

$$\left(T \times R_{A1}^{(7\times3)}\right) \bmod N.$$

Покупатель (абонент *A*) хочет заплатить продавцу (абоненту *B*) 10 у.е. Для этого абонент *A* формирует составное сообщение

$$S_1^{1/(11\times5)} \bmod N \parallel \left(T \times R_{A1}^{(7\times3)}\right) \bmod N$$

и посылает его абоненту *B*.

Шаг 6. Абонент *B* пересылает это сообщение банку.

Шаг 7. Банк проверяет номер S_1, обращаясь к списку номеров ранее использованных купюр. В случае положительного результата сравнения на счет *B* переводится сумма в 10 у.е.

Шаг. 8. Банк возвращает абоненту *B* купюру *T*, подписанную с использованием экспоненты $1/(7 \times 3)$

$$\left(T^{1/(7\times3)} \times R_{A1}\right) \bmod N .$$

Шаг 9. Абонент *B* пересылает это сообщение (по сути, сдачу) абоненту *A*.

Шаг.10. Абонент *A* снимает действие затемняющего множителя и получает подписанную банком купюру достоинством 5 у.е.

$$T^{1/(7\times3)} \bmod N .$$

Транзакция покупки 2

Шаг 11. Допустим второй платеж осуществляется на сумму в 12 у.е. (двоичное представление 1100), открытая экспонента в этом случае равна 11×7. Составное сообщение от *A* к *B* с использованием затемняющего множителя R_{A2} в этом случае будет иметь вид

$$S_2^{1/(11\times7)} \bmod N \left\| \left(T^{1/(7\times3)} \times R_{A2}^{(5\times3)}\right) \right. \bmod N .$$

Шаг 12. Абонент *B* пересылает это сообщение банку.

Шаг 13. Банк проверяет номер S_2, обращаясь к списку номеров ранее использованных купюр. В случае положительного результата сравнения на счет *B* переводится сумма в 12 у.е.

Шаг 14. Банк возвращает абоненту *B* купюру *T*, подписанную с использованием экспоненты $1/(5 \times 3)$

$$\left(T^{1/(7\times3)(5\times3)} \times R_{A2}\right) \bmod N .$$

Шаг 15. Абонент *B* пересылает это сообщение абоненту *A*.

Шаг 16. Абонент *A* снимает действие затемняющего множителя и получает подписанную банком купюру с «накопленным» достоинством $(5 + 3)$ у.е.

$$T^{1/(7\times3)(5\times3)} \bmod N.$$

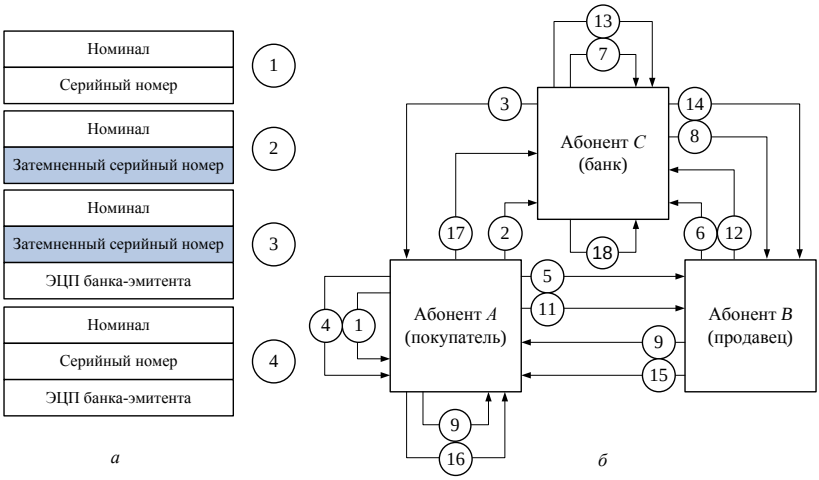


Рис. 9.4. Централизованная платежная система:
а – формат цифровой купюры; *б* – жизненный цикл цифровой купюры – обмен с использованием цифровых купюр различного достоинства и получением сдачи

Транзакция зачисления на счет

Шаг 17. В транзакции депозита абонент *A* может положить накопленную сумму на свой счет. Для этого он посылает в банк сообщение

$$T^{1/(7\times3)(5\times3)} \bmod N,$$

указывая при этом значение накопленной суммы.

Шаг 18. Банк проверяет, не использовалась ли ранее купюра-накопитель, вычисляет накопленную сумму и переводит ее на счет абонента *A*.

9.5. Механизмы обеспечения безопасности информации в ЭПС на основе протокола SET

Протокол SET предназначен для защиты транзакций по банковским картам, проводимых через открытые сети типа Интернет. SET работает на уровне приложений независимо от транспортного слоя. SET обеспечивает безопасность исключительно программными средствами.

Участники платежной системы на основе SET: держатель платежной карты, сервер продавца, платежный шлюз, центр сертификации, эмитент платежной карты, организация-эквайер (банк продавца). Архитектура платежной системы на основе SET показана на рис. 9.5.

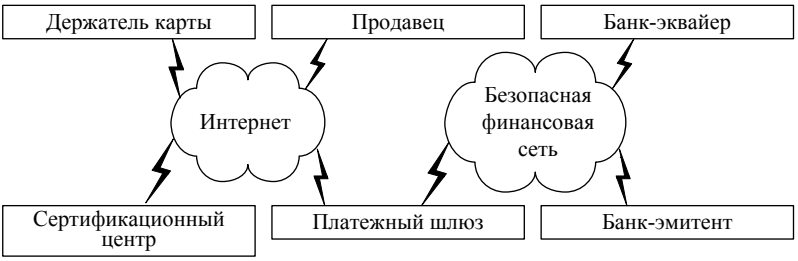


Рис. 9.5. Электронная платежная система на основе SET

С точки зрения обеспечиваемого уровня безопасности SET является одним из лучших прикладных протоколов. Наибольшего внимания при этом заслуживают три стохастических механизма ОБИ:

1) технология OAEP (Optimal Asymmetric Encryption Padding);
2) схема двойной электронной подписи (рис. 9.6);
3) последовательность обработки сообщений покупателя (рис. 9.7) с использованием гибридного шифрования.

Рис. 9.6. Схема двойной электронной подписи

Технология OAEP суть технология дополнения шифруемых данных незначащей случайной информацией для повышения криптостойкости.

Пусть m – открытое сообщение, $G(R)$ – ПСП, полученная с выхода генератора при его инициализации значением R, $|G(R)| = |m|$.

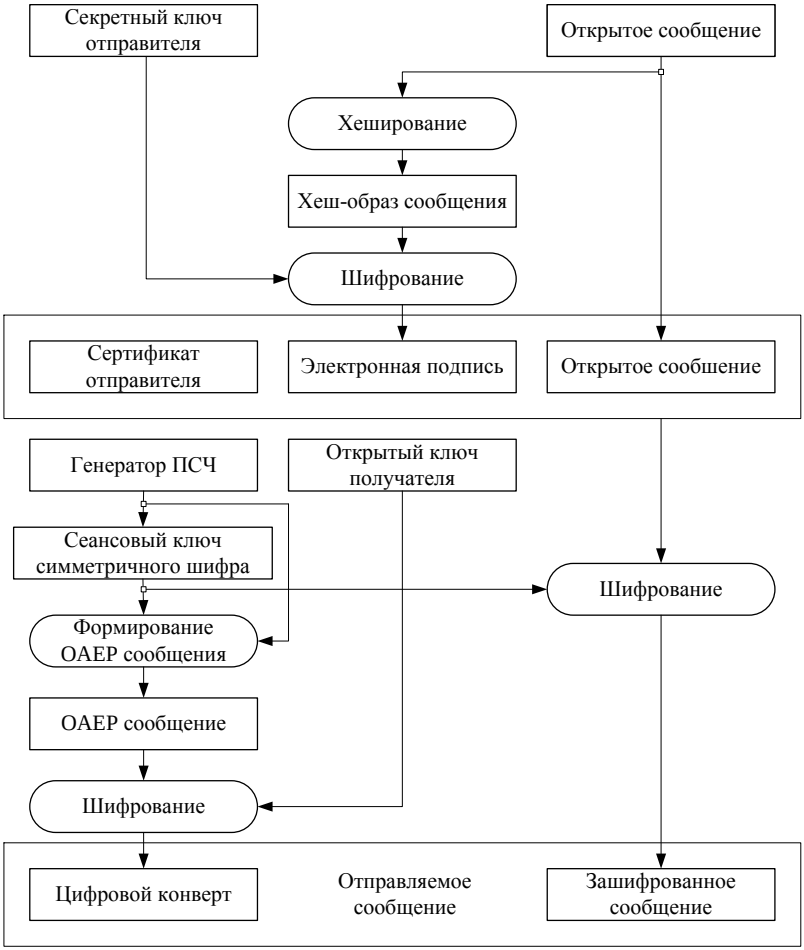


Рис. 9.7. Стохастическая обработка сообщений покупателя

OAEP, схема зашифрования

1. Отправитель (абонент A) формирует секретное случайное число R.
2. Отправитель инициализирует генератор значением R и получает последовательность $G(R)$.
3. Отправитель формирует преобразованное сообщение в виде

$$m' = (m \oplus G(R)) \,\|\, (R \oplus h(m \oplus G(R))).$$

4. Отправитель шифрует преобразованное сообщение на открытом ключе получателя (абонента B)
$$c = E_{PK_B}(m') = PK_B\{(m \oplus G(R)) \,\|\, (R \oplus h(m \oplus G(R)))\}.$$

OAEP, схема расшифрования

1. Получатель расшифровывает полученное сообщение на своем секретном ключе
$$m' = D_{SK_B}(c) = (m \oplus G(R)) \,\|\, (R \oplus h(m \oplus G(R))).$$
2. Получатель вычисляет хеш-образ $h(m \oplus G(R))$ последовательности $m \oplus G(R)$.
3. Получатель вычисляет число R, которое использовалось при создании сообщения m'
$$R = (R \oplus h(m \oplus G(R))) \oplus h(m \oplus G(R)).$$
4. Используя вычисленное значение R, получатель формирует $G(R)$.
5. Получатель вычисляет исходное сообщение m
$$m = (m \oplus G(R)) \oplus G(R).$$

Двойная электронная подпись. Одной из инноваций, представленных в SET, является схема двойной электронной подписи, позволяющая связывать элементы двух сообщений, каждое из которых отсылается своему адресату с одной ЭЦП во избежание лишнего обмена данными. Каждый получатель может прочитать сообщение, адресованное лично ему и проверить целостность других сообщений, не имея возможности прочесть их. Схема используется в транзакции покупки SET, при выполнении которой покупатель одновременно посылает (через платежный шлюз) продавцу ордер заказа, а банку – платежные инструкции. При этом покупателю необходимо иметь уверенность, что платежные инструкции выполнились, только после того, как продавец примет ордер заказа.

Исходная ситуация – абонент A (покупатель) имеет сообщение m_1 (ордер заказа), предназначенное для абонента B (сервер

продавца) и сообщение m_2 (платежные инструкции), предназначенное для абонента *C* (платежный шлюз). Пусть $h(x)$ – общеизвестная хеш-функция, PK_A и SK_A – соответственно открытый и закрытый ключи абонента *A*, PK_C и SK_C – соответственно открытый и закрытый ключи абонента *C*. Протокол двойной электронной подписи (см. рис. 9.6) имеет следующий вид:

1) абонент *A* вычисляет хеш-образы сообщений m_1 и m_2 и формирует сообщение
$$m_3 = h(m_1) \| h(m_2);$$
абонент *A* вычисляет хеш-образ $h(m_3)$ сообщения m_3;
2) абонент *A* формирует сообщение
$$m_1 \| h(m_2) \| PK_C\{m_2 \| h(m_2)\} \| \{h(m_3)\}SK_A$$
и отправляет его абоненту *B*;
3) абонент *B* вычисляет хеш-образ $h(m_1)$ сообщения m_1; формирует
$$m_3 = h(m_1) \| h(m_2)$$
и вычисляет хеш-образ m_3 – $h(m_3)$; абонент *B* проверяет аутентичность сообщения m_1, сравнивая вычисленное значение $h(m_3)$ с результатом
$$PK_A\{\{h(m_3)\} SK_A\}$$
расшифрования $\{h(m_3)\}SK_A$ на открытом ключе абонента *A*;
4) абонент *B* формирует сообщение
$$h(m_1) \| PK_C\{m_2 \| h(m_2)\} \| \{h(m_3)\}SK_A$$
и отправляет его абоненту *C*;
5) абонент *C* расшифровывает $PK_C\{m_2 \| h(m_2)\}$ на своем секретном ключе и находит m_2 и $h(m_2)$; абонент *C* формирует
$$PK_C\{m_2 \| h(m_2)\}$$
и вычисляет хеш-образ m_3 – $h(m_3)$; абонент *C* проверяет

аутентичность сообщения m_2, сравнивая вычисленное значение $h(m_3)$ с результатом

$$PK_A\{\{h(m_3)\}SK_A\}$$

расшифрования $\{h(m_3)\}SK_A$ на открытом ключе абонента *A*.

В результате выполнения протокола абонент *B* получил адресованное ему сообщение m_1, убедился в его аутентичности и ничего не узнал о содержимом сообщения m_2. Аналогично, абонент *C* получил адресованное ему сообщение m_2, убедился в его аутентичности и ничего не узнал о содержимом сообщения m_1.

Контрольные вопросы

1. Как в ЭПС на основе цифровой наличности защищаются интересы покупателя?
2. Как в ЭПС на основе цифровой наличности защищаются интересы продавца?
3. Как в ЭПС на основе цифровой наличности защищаются интересы банка-эмитента?
4. Опишите формат цифровой купюры.
5. Что такое анонимность и неотслеживаемость платежей?
6. Опишите суть технологии OAEP.

ГЛАВА 10. ИССЛЕДОВАНИЕ СТАТИСТИЧЕСКОЙ БЕЗОПАСНОСТИ ГЕНЕРАТОРОВ ПСЧ

10.1. Тесты, применяемые для оценки качества генераторов ПСЧ

Для исследования ПСП применяются две группы тестов:

1) *графические*;
2) *оценочные*.

В первом случае статистические свойства последовательностей отображаются в виде графических зависимостей, по виду которых делают выводы о свойствах исследуемой ПСП. Во втором случае статистические свойства последовательностей определяются числовыми характеристиками. На основе оценочных критериев делаются заключения о степени близости свойств анализируемой и истинно случайной последовательностей.

В отличие от графических тестов, где результаты интерпретируются пользователями, вследствие чего возможны различия в трактовке результатов, оценочные тесты характеризуются тем, что они выдают численную характеристику, которая позволяет однозначно сказать, пройден тест или нет.

Описания тестов с примерами их использования приведены в [16, 18]. Наиболее известная система статистического тестирования генераторов ПСЧ это DIEHARD Дж. Марсальи (Флоридский Государственный Университет, США) [44]. Наиболее полное описание тестов, применяемых для анализа генераторов ПСЧ, ориентированных на использование в системах защиты информации, содержится в Руководстве НИСТ по статистическому тестированию генераторов ПСЧ (в дальнейшем просто Руководство НИСТ) [33].

DIEHARD – это специализированная система, ориентированная в первую очередь на проверку конгруэнтных генераторов, разработанных Дж. Марсальи. Можно перечислить следующие недостатки системы DIEHARD:

параметры тестирования жестко заданы (например, размер области тестирования предопределена, иначе говоря, независимо от размера файла анализируется определенное число байт);

полностью отсутствуют справочная служба и методика трактовки результатов;
большинство тестов основано не на теоретических расчетах, а на результатах испытаний.

10.2. Руководство по статистическому тестированию НИСТ

Различные статистические тесты могут применяться к ПСП, для того чтобы сравнить ее с истинно случайной последовательностью. Случайность – вероятностное свойство; это означает, что свойства случайной последовательности могут быть охарактеризованы и описаны в терминах вероятности. Вероятный результат статистических тестов, применяемых к истинно случайной последовательности, известен априорно и может быть описан в вероятностных терминах. Существует огромное количество возможных статистических тестов, оценивающих присутствие или отсутствие «образца», который, при обнаружении, указал бы, что последовательность неслучайна. Поскольку существует так много тестов, оценивающих, является ли последовательность случайной или нет, никакой определенный конечный набор тестов не считают «законченным». Кроме того, результаты статистического теста должны интерпретироваться с некоторой осторожностью, чтобы избежать неправильных заключений об определенном генераторе.

Статистический тест формулируется для проверки определенной *нулевой гипотезы* H_0 о том, что проверяемая последовательность является *случайной.* С этой нулевой гипотезой связана альтернативная гипотеза H_a о том, что последовательность *не случайна*. Для каждого применяемого теста получают заключение о принятии или отклонении нулевой гипотезы, основываясь на сформированной исследуемым генератором последовательности.

Для каждого теста должна быть выбрана подходящая статистика случайности для принятия или отклонения нулевой гипотезы. Согласно предположению о случайности, такая статистика имеет распределение возможных значений. Теоретическое эталонное распределение этой статистики для нулевой гипотезы определяется математическими методами. Из этого эталонного

распределения определяется *критическое значение*. Во время проведения теста, вычисляется значение тестовой статистики. Оно сравнивается с критическим значением. Если значение тестовой статистики превышает критическое значение, нулевая гипотеза для случайности отклоняется. В противном случае нулевая гипотеза принимается.

Проверка статистических гипотез работает, потому что эталонное распределение и критическое значение зависят и генерируются согласно предварительному предположению о случайности. Если предположение о случайности последовательности истинно, то результирующее значение тестовой статистики для нее будет иметь очень низкую вероятность превышения критического значения (например, 0,01).

С другой стороны, если расчетное значение тестовой статистики превышает критическое значение (т.е. происходит событие с низкой вероятностью), то, с точки зрения проверки статистической гипотезы, событие с низкой вероятностью не может встретиться естественно. Поэтому когда расчетное значение тестовой статистики превышает критическое значение, делается заключение, что первоначальное предположение о случайности является подозрительным или ошибочным. В этом случае делается заключение об отклонении H_0 (случайность) и принятии H_a (неслучайность).

Проверка статистической гипотезы – процедура генерации заключения, которая имеет два возможных результата, или принять H_0 (данные случайны) или принять H_a (данные неслучайны). Табл. 10.1 связывает истинное (неизвестное) состояние данных с заключением, полученным процедурой проверки.

Если данные на самом деле случайны, то заключение об отклонении нулевой гипотезы (т.е. данные неслучайны) будет приниматься крайне редко. Это заключение называется ошибкой первого рода. Если данные неслучайны, то заключение о принятии нулевой гипотезы (т.е. данные случайны) называется ошибкой второго рода. Заключения о принятии H_0, когда данные действительно случайны, и отклонении H_0, когда данные неслучайны, являются правильными.

Таблица 10.1

Принятие заключений
по результатам проведения статистических испытаний

Ситуация	Заключение	
	Принять H_0	Принять H_a
Данные случайны (H_0 истинна)	Нет ошибки	Ошибка 1-го рода
Данные неслучайны (*На* истинна)	Ошибка 2-го рода	Нет ошибки

Вероятность ошибки 1-го рода называется *уровнем значимости* теста. Эта вероятность может быть установлена до испытания и обозначается как α. Для теста α суть вероятность того, что тест укажет на неслучайность последовательности, тогда как на самом деле она случайна. То есть на то, что последовательность имеет неслучайные свойства, даже если ее произвел *хороший* генератор.

Вероятность ошибки 2-го рода обозначается как β. Для теста β есть вероятность того, что тест укажет на случайность последовательности, тогда как это не так; то есть *плохой* генератор произвел последовательность, которая, как кажется, имеет случайные свойства. В отличие от α, β может принимать множество различных значений, потому что существует бесконечное число ситуаций, когда поток данных может быть неслучаен, и каждая из них выдает различные β. Вычисление ошибки 2-го рода является более сложным из-за множества возможных типов неслучайности.

Одна из основных целей статистических тестов – минимизация вероятности ошибки 2-го рода, иначе говоря, минимизация вероятности принятия последовательности, произведенной плохим генератором, за последовательность, произведенную хорошим генератором. Вероятности α и β связаны друг с другом и с длиной n проверяемой последовательности: если два из этих значений определены, третье определяется автоматически. На практике обычно выбирают размер n и значение для α (вероятности ошибки 1-го рода). Тогда критическая точка для данной статистики выбирается так, чтобы получить наименьшее значение β (вероятность ошибки 2-го рода).

Тестовая статистика использует вычисление значение *P-value*, которое констатирует силу доказательства против нулевой гипотезы, иначе говоря, *P-value* есть вероятность того, что совершенный генератор случайных чисел произвел бы последовательность менее случайную, чем исследуемая, для типа неслучайности, проверяемого тестом. Если *P-value* для теста равно 1, то последовательность абсолютно случайна. *P-value*, равное 0, указывает, что последовательность абсолютно неслучайна. Для теста следует выбрать уровень значимости α. Если значение *P*-value больше или равно α, то принимается нулевая гипотеза, т.е. последовательность кажется случайной. Если значение *P*-value меньше α, то нулевая гипотеза отклоняется, т.е. последовательность кажется неслучайной. Параметр α обозначает вероятность ошибки 1-го рода. Как правило, α выбирается в интервале [0,001; 0,01].

Значение α, равное 0,01, говорит о том, что из 100 случайных последовательностей не прошла бы тест лишь одна. При *P-value* $\geq$ 0,01 последовательность рассматривается как случайная с доверительностью 99 %, иначе говоря, с вероятностью 0,99 последовательность случайна. При *P-value* $<$ 0,01 последовательность рассматривается как неслучайная с доверительностью 99 %, иначе говоря, с вероятностью 0,99 последовательность неслучайна.

Значение α, равное 0,001, говорит о том, что из 1000 случайных последовательностей не прошла бы тест лишь одна. При *P-value* $\geq$ $\geq$ 0,001 последовательность рассматривается как случайная с доверительностью 99,9 %. При *P-value* $<$ 0,001 последовательность рассматривается как неслучайная с доверительностью 99,9 %.

По отношению к исследуемым последовательностям можно сделать следующие предположения.

Равномерность. В любой точке при генерации последовательности случайных или псевдослучайных битов 0 и 1 равновероятны и вероятности их появления равны 1/2. Ожидаемое число нулей (или единиц) равно $n/2$, где n – длина последовательности.

Масшабируемость. Любой тест, применимый к последовательности, может также применяться к произвольной под-

последовательности. Если последовательность случайна, то любая ее подпоследовательность также должна быть случайной. Следовательно, любая подпоследовательность должна пройти все тесты на случайность.

Полнота. Поведение генератора ПСЧ связано с начальным заполнением, поэтому неверно делать заключение о качестве генератора, основываясь на результатах анализа последовательности при каком-то одном начальном заполнении. Аналогично неверно делать заключение о генераторе случайных чисел, основываясь только на результатах анализа одного произведенного им фрагмента последовательности.

Итак, оценка статистических испытаний основана на проверке гипотезы о случайности исследуемой последовательности нулей и единиц. Табл. 10.2 показывает пошаговый процесс, позволяющий оценить конкретную двоичную последовательность.

Оценка результатов тестирования. Процесс исследования статистических свойств генератора ПСЧ состоит из следующих шагов:

генерация последовательностей для тестирования;

исполнение набора статистических тестов;

анализ прохождения статистических тестов;

принятие решения о свойствах генератора.

Таблица 10.2

Процедура оценки

Номер шага	Пошаговый процесс	Комментарии
1	Постановка гипотезы	Предполагаем, что последовательность является случайной
2	Вычисление тестовой статистики последовательности	Проводим тестирование на битовом уровне
3	Вычисление *P-value*	*P-value* $\in [0; 1]$
4	Сравнение *P-value* с α	Задаем α, где $\alpha \in [0{,}001; 0{,}01]$. Если *P-value* $\geq \alpha$ – тесты пройдены

Генерация последовательностей для тестирования. Для заданного генератора формируется m последовательностей длины *n*.

$$\varepsilon^{(1)} = \varepsilon_1^{(1)}\varepsilon_2^{(1)}..\varepsilon_n^{(1)},$$
$$\varepsilon^{(2)} = \varepsilon_1^{(2)}\varepsilon_2^{(2)}..\varepsilon_n^{(2)},$$
$$\ldots$$
$$\varepsilon^{(m)} = \varepsilon_1^{(m)}\varepsilon_2^{(m)}..\varepsilon_n^{(m)}.$$

Длина последовательности *n* выбирается таким образом, чтобы все тесты могли быть пройдены.

Исполнение набора статистических тестов. Каждая из *m* последовательностей проверяется каждым из *t* тестов набора. Результат работы каждого теста – вычисление тестовой статистики *s*(*obs*). Таким образом, после проверки всех последовательностей получается *mt* тестовых статистик, как показано в табл. 10.3.

Таблица 10.3

Результаты выполнения набора статистических тестов

Последовательность	Тест 1	Тест 2	…	Тест *t*
$\varepsilon^{(1)}$	$s_1^{(1)}(obs)$	$s_2^{(1)}(obs)$	…	$s_t^{(1)}(obs)$
$\varepsilon^{(2)}$	$s_1^{(2)}(obs)$	$s_2^{(2)}(obs)$	…	$s_t^{(2)}(obs)$
…	…	…	…	…
$\varepsilon^{(m)}$	$s_1^{(m)}(obs)$	$s_2^{(m)}(obs)$	…	$s_t^{(m)}(obs)$

Анализ прохождения статистических тестов. Анализ прохождения статистических тестов начинается с анализа тестовой статистки. Существует три варианта оценки тестовой статистики.

1. *Пороговые значения*. Если тестовая статистика больше (меньше) порогового значения, последовательность считается неслучайной.
2. *Фиксированные интервалы*. Если тестовая статистика выходит за пределы заданного интервала, последовательность считается неслучайной.
3. *Вероятностные значения*. Для тестовой статистики вычисляется *P-value*. Под *P-value* понимается вероятность того, что совершенный генератор случайных чисел произвел бы последовательность менее случайную, чем исследуемая, для типа неслучайности, проверяемого тес-

том. Для теста выбирается уровень значимости α. Если значение *P-value* больше, либо равно α, то последовательность считается случайной.

Поскольку для первых двух способов необходимо заранее рассчитывать пороговые значения и фиксированные интервалы, вычисление *P-value* представляется наиболее эффективным вариантом оценки тестовой статистики.

Таким образом, вычисляются значения *P-value* для тестовых статистик $s(obs)$ как показано в табл. 10.4.

Таблица 10.4

Результаты вычисления значений *P-value* для тестовых статистик $s(obs)$

Последовательность	Тест 1	Тест 2	…	Тест t
$\varepsilon^{(1)}$	$P\text{-}value_1^{(1)}$	$P\text{-}value_2^{(1)}$	…	$P\text{-}value_t^{(1)}$
$\varepsilon^{(2)}$	$P\text{-}value_1^{(2)}$	$P\text{-}value_2^{(2)}$	…	$P\text{-}value_t^{(2)}$
…	…	…	…	…
$\varepsilon^{(m)}$	$P\text{-}value_1^{(m)}$	$P\text{-}value_2^{(m)}$	…	$P\text{-}value_t^{(m)}$

Существует два варианта оценки прохождения набора из m последовательностей i-го теста, $i = \overline{1, t}$.

1. *Анализ числа появлений значений P-value*. Множество значений *P-value* [0; 1] разбивается на k категорий с вероятностями, равными $1/k$ в каждой категории, после чего подсчитывается ν_i, $i = \overline{1, k}$, – число последовательностей, значения *P-value* которых принадлежат i-й категории. Вычисляется статистика

$$\chi^2(obs) = \frac{\sum_{i=1}^{k}\left(\nu_i - \frac{m}{k}\right)^2}{\frac{m}{k}},$$

которая анализируется при помощи критерия χ^2 с числом степеней свободы, равным $k - 1$. Одним из преимуществ данного варианта является то, что он позволяет косвенным образом определить значение m. Поскольку эмпири-

ческое правило для критерия χ^2 гласит, что значение $\frac{m}{k}$ должно быть больше либо равным 5, то $m \geq 5k$.

2. *Анализ значений P-value*. Подсчитывается доля последовательностей, прошедших данный тест (т.е. доля последовательностей, для которых *P-value* $\geq \alpha$). Значение этой доли должно лежать в интервале

$$\left[1-\alpha-3\sqrt{\frac{\alpha(1-\alpha)}{m}};1-\alpha+3\sqrt{\frac{\alpha(1-\alpha)}{m}}\right].$$

Результаты тестирования набора из m последовательностей каждым из t тестов могут быть сведены в таблицу, аналогичную, например, табл. 10.5.

Таблица 10.5

Результаты тестирования m последовательностей набором из t статистических тестов

Последовательность	Тест 1	Тест 2	…	Тест t
$\varepsilon^{(1)}$	прошла / не прошла	прошла / не прошла	…	прошла / не прошла
$\varepsilon^{(2)}$	прошла / не прошла	прошла / не прошла	…	прошла / не прошла
…	…	…	…	…
$\varepsilon^{(m)}$	прошла / не прошла	прошла / не прошла	…	прошла / не прошла
Результат проверки генератора	прошел / не прошел	прошел / не прошел	…	прошел / не прошел

Непрохождение какого-либо теста свидетельствует о статистических слабостях в структуре генератора.

10.3. Повышение эффективности оценочных тестов

Механизм работы практически всех тестов из вышеупомянутых подборок основан на подсчете числа появлений определенных шаблонов и сравнения полученных значений с теоретическими. При этом для хранения данных о числе появлений шаблонов для последовательности длиной n, представляющей из себя наборы, состоящие из k m-разрядных чисел, требуется объем

памяти, равный $2^{km\left\lceil \log_2\left(\left\lfloor \frac{n}{k} \right\rfloor +1\right)\right\rceil}$ бит в случае непересекающихся шаблонов и $2^{km\left\lceil \log_2(n+1)\right\rceil}$ в случае пересекающихся. Учитывая, что значение n должно быть большим, существенно возрастает объем требуемой памяти. Например, для последовательности размером 1 Мбайт анализ наборов, состоящих из пяти полубайтов, потребует памяти размером 500 Кбайт в случае непересекающихся наборов и 1,25 Мбайт в случае пересекающихся. В идеальном случае, при длине последовательности, стремящемся к бесконечности, размер памяти также будет стремиться к бесконечности. Таким образом, возникает задача уменьшения объема памяти, требуемой для реализации теста.

В настоящее время большинство разработчиков оценочных тестов (в том числе и авторы наиболее популярного на сегодняшний день Руководства НИСТ США) решает данную проблему путем введения ограничений на размеры анализируемых шаблонов и длину исследуемой последовательности.

Уменьшение размеров шаблонов приводит к существенному ослаблению теста, так как, зная логику работы последнего, можно внести соответствующие изменения в алгоритм работы генератора ПСЧ или непосредственно в саму последовательность, благодаря чему свойства последней будут неотличимы от свойств истинно случайной последовательности для типа неслучайности, проверяемого заданным тестом. Данное утверждение косвенно подтверждается результатами исследований наиболее эффективных из существующих генераторов ПСЧ. Все они с легкостью проходят тесты «Проверка серий пар» и «Проверка серий троек» (размеры шаблонов – два и три бита соответственно) и полностью проваливают «Посимвольную проверку» (размер шаблона – восемь бит). И это только для шаблонов, состоящих из одного числа. Можно предположить, что увеличение количества чисел в шаблоне сделает статистику прохождения еще более удручающей.

Уменьшение длины исследуемой последовательности также снижает качество тестирования. Для примера, разработчики набора статистических тестов НИСТ США рекомендуют использовать для анализа последовательности длиной всего лишь 2^{20} бит, или 128 Кбайт, что для существующих объемов передаваемой информации, исчисляемой мегабайтами, гигабайтами и даже терабайтами, является просто недопустимым.

Один из вариантов решения данной проблемы косвенно предложен в Руководстве НИСТ. Суть его заключается в тестировании не всей последовательности целиком, а подпоследовательностей. Однако данный подход не лишен недостатков. Основной из них связан с выбором размера подпоследовательности. При увеличении размера подпоследовательности опять возникает проблема дополнительной памяти. Уменьшение размера подпоследовательностей приводит к необходимости оценки корреляции между ними для исключения влияния периодичности.

Выход из сложившейся ситуации видится в модификации механизма работы тестов, суть которой заключается в том, чтобы анализировать не число появлений определенных шаблонов, а число отсутствующих шаблонов. В этом случае для хранения информации о шаблоне достаточно будет одного бита, что приведет к уменьшению объема памяти до 2^{km} для пересекающихся и непересекающихся шаблонов, при этом цель теста не будет изменена, и он продолжит выявлять те же статистические отклонения, что и оригинальный тест. Таким образом, размер требуемой для реализации теста памяти перестает зависеть от длины исследуемой последовательности. Для примера, для реализации тестов НИСТ при рекомендуемой длине в 128 Кбайт объем требуемой памяти уменьшается в 20 раз.

Рассмотрим, как меняется механизм вычисления статистики теста. Для заданной последовательности длиной n, представляющей из себя наборы, состоящие из k m-разрядных чисел, подсчитываем число отсутствующих наборов. Дж. Марсалья показал, что число отсутствующих наборов аппроксимируется с нормальным распределением. Найдем среднее и отклонение.

Для расчета среднего необходимо рассмотреть разложение в ряд Тейлора производящей функции, соответствующей значениям k и m. Очевидно, это не очень удобно, поскольку для различных шаблонов придется заново определять производящую функцию для каждого типа набора и раскладывать ее в ряд Тейлора. Например, для $k = 2$ расчет осуществляется следующим образом:

вычисляется p_1 – коэффициент при z_n в разложении производящей функции

$$\frac{1}{1 - z + p^2 z^2};$$

вычисляется p_2 – коэффициент при z_n в разложении производящей функции

$$\frac{1 + p\,z}{1 - (1 - p)z - (p - p^2)z^2}$$

вычисляется среднее для числа отсутствующих слов

$$\mu = \left(2^{km} - 2^m\right)p_1 + 2^m p_2$$

С увеличением k возрастает число производящих функций, а также их сложность. Поэтому для расчета предлагается использовать подход Дж. Марсалья, который показал, что среднее можно вычислить (при условии, что $n > 1000$) по формуле:

$$\mu = 2^{km} e^{-\frac{n}{2^{km}}}.$$

Действительно, для случая $n = 2^{21}$, $k = 2$, $m = 10$ имеем

$$\mu = (2^{2\cdot 10} - 2^{10})\cdot 0{,}135335283236469 +$$
$$+\ 2^{10}\cdot 0{,}135599351997986596411 \approx 141909{,}60;$$

$$\mu = 2^{2\cdot 10} e^{-\frac{n}{2\cdot 10}} \approx 141909{,}33.$$

Расчет отклонения осуществляется по следующей формуле:

$$\sigma = \sqrt{\sum_{i_1=1}^{2^m}\sum_{i_2=1}^{2^m}\ldots\sum_{i_k=1}^{2^m}\operatorname{cov}\left(n_{i_1},\, n_{i_2},\, \ldots,\, n_{i_k}\right)},$$

где $n_i = \begin{cases} 1, \text{если буква, равная } 2^i, \text{ присутствует в слове;} \\ 0, \text{если буква, равная } 2^i, \text{ отсутствует в слове.} \end{cases}$

Примечание. Для вычисления отклонения рекомендуется использовать специализированные программные продукты (MathCad и др.), в которых присутствует возможность вычисления ковариации нескольких переменных.

В табл. 10.6–10.8 показано, какие тесты из наиболее популярных подборок для оценки статистических свойств могут быть модифицированы. Как можно заметить, улучшить можно больше половины тестов подборки Д. Кнута [18] и системы DIEHARD, а также четвертую часть тестов Руководства НИСТ.

Таблица 10.6

Возможность модификации тестов подборки Д. Кнута

№	Название теста	Возможность модификации
1	Проверка несцепленных серий	+
2	Проверка интервалов	–
3	Проверка комбинаций	+
4	Тест собирателей купонов	+
5	Проверка перестановок	+
6	Проверка на монотонность	–
7	Проверка корреляции	–
Итого		4 из 7

Таблица 10.7

Возможность модификации тестов DIEHARD

№	Название теста	Возможность модификации
1	Промежутки между днями рождения	–
2	Проверка пересекающихся перестановок	+
3–5	Проверка рангов матриц	+
6–10	Буквенные («обезьяньи тесты») (5 тестов)	+
11	Подсчет числа единиц в потоке байт	+
12	Подсчет числа единиц в определенных байтах	+
13	Тест НОД	–
14	Тест парковки	+
15	Тест минимальных расстояний	–
16	Тест пузырей	–
Итого		12 из 16

Таблица 10.8

Возможность модификации тестов Руководства НИСТ

№	Название теста	Возможность модификации
1	Частотный тест	–
2	Проверка кумулятивных сумм	–
3	Проверка «дырок» в подпоследовательностях	–
4	Проверка «дырок»	–
5	Проверка рангов матриц	+
6	Спектральный тест	–
7	Проверка непересекающихся шаблонов	+
8	Проверка пересекающихся шаблонов	+
9	Универсальный статистический тест Маурера	–
10	Проверка случайных отклонений	–
11	Разновидность проверки случайных отклонений	–
12	Проверка аппроксимированной энтропии	–
13	Проверка серий	+
14	Сжатие при помощи алгоритма Лемпела–Зива	–
15	Линейная сложность	–
Итого		4 из 16

Эффективность применения данного подхода обусловлена следующими факторами.

1. Размер дополнительной памяти для сбора статистики теста теперь не зависит от длины тестируемой последовательности и определяется только размерами шаблона. Это позволит выделять данную память статически, что ускорит выполнение теста при программной реализации, или прогнозировать объем

памяти, выделяемой динамически.

2. Учитывая, что значение длины последовательности теперь не влияет на объем затрачиваемой памяти, все существующие оценочные тесты можно будет применять не к части последовательности фиксированной длины (ограниченной в существующих системах несколькими мегабайтами), а в случае необходимости ко всему периоду.
3. Механизм работы модифицированных тестов предполагает досрочное завершение процесса исследования в том случае, если исследуемая статистика вышла за границы доверительного интервала, что существенно сокращает время тестирования.
4. Уменьшение объемов требуемой памяти позволит более свободно варьировать параметры тестирования, увеличивая диапазон используемых значений, что сущеественно повысит функциональность тестов и качество тестирования.

10.4. Комплекс программных средств оценки качества генераторов ПСЧ

Анализ недостатков существующих систем для тестирования генераторов ПСЧ позволяет перечислить свойства, которые должна иметь полнофункциональная система для исследования статистических свойств ПСП.

1. *Полнота тестов.* Каждый отдельно взятый (даже очень сильный) оценочный тест можно «обмануть», т.е. сформировать заведомо плохую последовательность, не удовлетворяющую совокупности сформулированных требований, но успешно проходящую рассматриваемый тест. Учитывая, что задача определения минимально допустимой совокупности тестов вряд ли имеет решение, должны быть реализованы все тесты, рекомендуемые для исследования генераторов, к качеству выходных последовательностей которых предъявляются наиболее жесткие требования. При этом система должна содержать как оценочные, так и графические тесты. Ни одна из сущест-

вующих систем не удовлетворяет этому требованию.

2. *Оценка периода.* Система должна содержать средства выявления периодичности в анализируемых последовательностях. При этом пользователь должен иметь возможность выбирать, проверять ли ему всю последовательность, фрагмент длиной в период и др. В существующих системах отсутствует возможность определения периода. Более того, зачастую длина анализируемой последовательности жестко задана.
3. *Оценка корреляции.* В качественных ПСП должна отсутствовать корреляция между отдельными выборками. Кроме того, в ряде случаев нелинейные преобразования, осуществляемые функцией обратной связи или функцией выхода генератора, состоят из нескольких шагов (раундов, уровней и пр.). Система должна содержать средства для анализа изменений, вносимых каждым шагом преобразования.
4. *Настройка параметров тестирования.* Пользователь должен иметь возможность настраивать параметры тестирования (например, задавать границы для количественных характеристик тестов, длины подпоследовательностей и т.д.). В существующих системах большинство параметров, как правило, фиксировано.
5. *Интерпретация результатов.* Результатом выполнения оценочного теста является численная характеристика. Зачастую для того, чтобы трактовать полученное значение, необходимо знать структуру теста. Поэтому система должна представлять результат в виде, понятном даже неподготовленному пользователю. В существующих системах такая возможность отсутствует.

Структура комплекса. На рис. 10.1 показана структура программного комплекса, отвечающего всем вышеперечисленным требованиям. В его состав входят следующие блоки.

Блок работы с файлами. Предназначен для считывания последовательностей из файлов и передачи их для обработки в тестовый блок.

Блок тестирования. Предназначен для исследования статистических свойств ПСП. В его состав входят три модуля:

оценки качества (набор из 7 графических и 16 оценочных тестов), *оценки периода* и *оценки корреляции* между файлами.

Блок настроек. Предназначен для определения имен тестируемых файлов или директорий, а также параметров тестов (границ для *P-value*, размеров серий и т.д.) и параметров тестирования (размера области тестирования, набора тестов и т.д.).

Блок отчета. Предназначен для просмотра, записи и печати результатов тестирования.

Блок управления. Обеспечивает согласованную работу всех блоков комплекса и взаимодействие с пользователем.

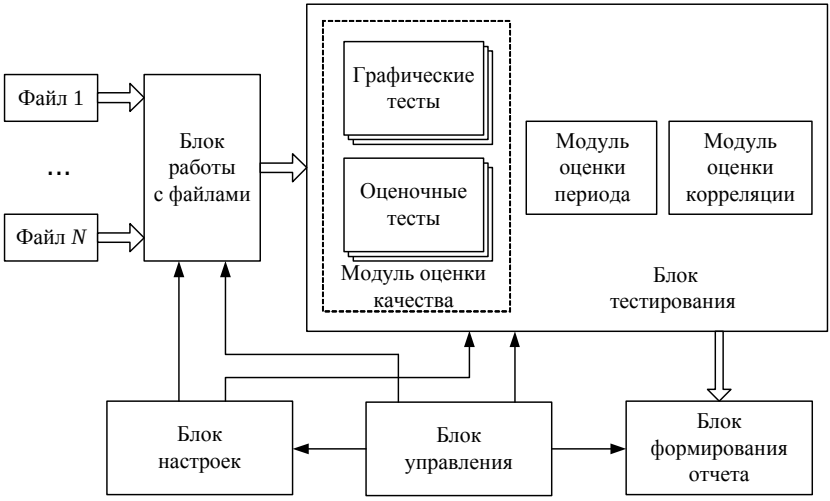


Рис. 10.1. Структура комплекса программных средств для исследования статистических свойств ПСП

Контрольные вопросы

1. Чем псевдослучайная последовательность отличается от истинно случайной?
2. Что такое графические статистические тесты?
3. Что такое оценочные статистические тесты?
4. Приведите пример графического статистического теста.

5. Приведите пример оценочного статистического теста.
6. Опишите последовательность статистического тестирования генераторов ПСЧ по методике НИСТ.
7. Сформулируйте требования к полнофункциональному комплексу оценки качества генераторов ПСЧ.

ЧАСТЬ 3. ЭЛЛИПТИЧЕСКАЯ КРИПТОГРАФИЯ

ГЛАВА 11. ЭЛЛИПТИЧЕСКИЕ КРИВЫЕ И КРИПТОГРАФИЯ

11.1. Основные понятия и определения

Эллиптической криптографией называют всё множество криптографических алгоритмов и протоколов, использующих свойства эллиптических кривых над конечными полями

Наиболее общее определение эллиптической кривой дает уравнение Вейерштрасса:

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6. \qquad (11.1)$$

Для конечного поля Галуа $GF(p)$, где $p > 3$ и является простым числом, уравнение Вейерштрасса имеет вид

$$y^2 = x^3 + ax + b \pmod p, \qquad (11.2)$$

где a и b – элементы поля $GF(p)$, такие, что справедливо выражение

$$4a^3 + 27b^2 \neq 0 \pmod p.$$

Для расширенного поля $GF(2^m)$ уравнение Вейерштрасса принимает вид

$$y^2 + xy = x^3 + ax^2 + b \qquad (11.3)$$

или

$$y^2 + cy = x^3 + ax + b, \qquad (11.4)$$

где a_1, a_2, a_3, a_4, a_6, a, b и c – элементы поля $GF(2^m)$, т.е. полиномы степени $m - 1$ над полем $GF(2)$. В уравнении (11.3) $b \neq 0$, а в уравнении (11.4) $c \neq 0$.

Уравнение (11.4) описывает суперсингулярную кривую. Для нее проблема дискретного логарифма эффективно решается. Поэтому суперсингулярные кривые неприменимы для целей криптографической защиты. Уравнение (11.3) описывает несуперсингулярную кривую.

Эллиптичная кривая E над конечным полем задается множеством точек на плоскости $P = (x_p, y_p)$, где x_p и y_p – элементы поля. Элементы поля a и b называются коэффициентами эллиптической кривой E.

Важным свойством эллиптической кривой является возможность представления множества точек кривой в виде группы.

11.2. Группа точек эллиптической кривой

Рассмотрим эллиптическую кривую E (рис. 11.1), определенную над множеством вещественных чисел и соответствующую уравнению

$$y^2 + y = x^3 - x^2.$$

На этой кривой лежат только четыре точки, координаты которых (рис. 11.1) целые числа. Это точки

$$A(0, 0),\ B(1, -1),\ C(1, 0) \text{ и } D(0, -1).$$

Покажем, что эти четыре точки в совокупности с бесконечно удалённой точкой O образуют группу. Для этого введем для них операцию сложения. Для определения операции сложения на группе точек эллиптической кривой будем считать, что:

на плоскости существует бесконечно удаленная точка $O \in E$, в которой сходятся все вертикальные прямые;

касательная к кривой пересекает точку касания P два раза (это допущение становится понятным, если вспомнить, что касательная PR суть предельное положение секущей PM (рис. 11.2) при стремлении точки M к точке P).

Теперь можно сформулировать правила сложения точек $P, Q \in E$:

проведем прямую линию через точки P и Q, найдем третью точку S пересечения этой прямой с кривой E;

проведем через точку S вертикальную прямую до пересечения с кривой E в точке T;

искомая сумма равна $P + Q = T$ (рис. 11.3).

Применив это правило к группе точек $G = \{A, B, C, D, O\}$, получим

$$A + A = B,\ A + B = C,\ A + C = D,\ A + D = O,$$

т.е. определенная нами операция сложения удовлетворяет условию замкнутости. Кроме того,

$$2A = B,\ 3A = C,\ 4A = D,\ 5A = O,\ 6A = A \text{ (рис. 11.4)},$$

иначе говоря, группа является циклической. Аналогично можно проверить и остальные свойства группы. Для любых точек $P, Q \in G$ справедливо $P + Q = Q + P$. Для любой точки $P \in G$

справедливо $P + O = P$, иными словами, точка O суть нейтральный (аддитивный единичный) элемент группы G.

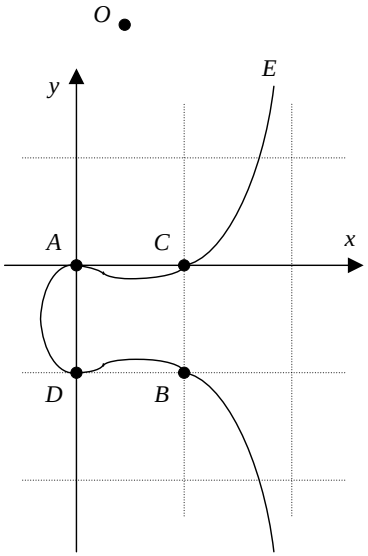


Рис. 11.1. Группа из пяти точек эллиптической кривой E, соответствующей уравнению $y^2 + y = x^3 - x^2$, O – бесконечно удаленная точка

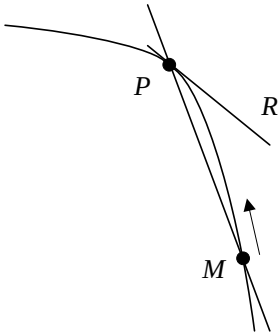


Рис. 11.2. Касательная к эллиптической кривой

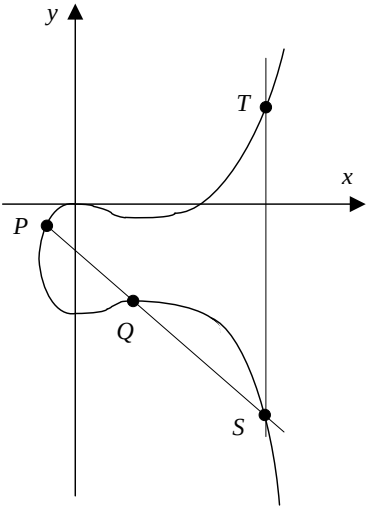


Рис. 11.3. Сложение точек на эллиптической кривой. $P + Q = T$

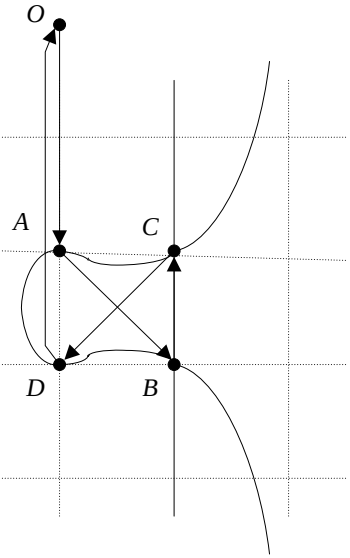


Рис. 11.4. Аддитивная абелева группа $\{A, B, C, D, O\}$
на эллиптической кривой

11.3. Сложение точек эллиптической кривой

В реальных криптосистемах часто используется уравнение

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

где $a, b \in GF(p)$, $4a^3 + 27b^2 \neq 0 \pmod p$, $p > 3$ – простое. Группа $E(GF(p))$ состоит из всех точек (x, y), где $x, y \in GF(p)$, удовлетворяющих уравнению, и бесконечно удаленной точки O. Определенная над точками из $E(GF(p))$ операция сложения алгебраически может быть описана следующим образом.

Пусть $P = (x_1, y_1)$ и $Q = (x_2, y_2)$. Тогда $P + Q = (x_3, y_3)$, где

$$x_3 = \lambda^2 - x_1 - x_2,$$

$$y_3 = \lambda(x_1 - x_3) - y_1,$$

а

$$\lambda = \begin{cases} \dfrac{y_2 - y_1}{x_2 - x_1}, \text{ если } P \neq Q; \\ \dfrac{3x_1^2 + a}{2y_1}, \text{ если } P = Q. \end{cases}$$

Число λ суть угловой коэффициент секущей, проведенной через точки $P = (x_1, y_1)$ и $Q = (x_2, y_2)$. При $P = Q$ секущая превращается в касательную, чем и объясняется наличие двух формул для вычисления λ.

Для любой точки $P \in E(GF(p))$ справедливо

$$P + O = O + P = = P.$$

Если $P = (x, y) \in E(GF(p))$, то $(x, y) + (x, -y) = O$. Точка $(x, -y) \in E(GF(p))$ является обратным элементом по отношению к P и обозначается $-P$.

Иначе говоря, вышеприведенные уравнения справедливы, кроме двух тривиальных случаев:

если одно из слагаемых совпадает с бесконечно удаленной точкой: $Q = O$, тогда $P + Q = O$;

если $P = (x, y)$, $Q = (x, -y)$, тогда $P + Q = O$.

11.4. Скалярное умножение точек эллиптической кривой

Скалярное умножение определяется для каждой точки на $E(GF(p))$ или $E(GF(2^m))$. Если точка $P \in E(GF(p))$ или $E(GF(2^m))$, а $k \in N$ – целое положительное, то

$$k \times P = \underbrace{P + P + P + \ldots + P}_{k \text{ раз}},$$

где операция (+) есть операция сложения на эллиптической кривой.

Алгоритм умножения точек (left-to-right binary method)

Вход: эллиптическая кривая E, положительное целое число k и точка $P = (x, y)$. $k = k_{m-1}2^{m-1} + \ldots + k_1 2^1 + k_0 2^0$ – двоичное представление числа k.

Выход: $Q = k \times P$.

1) Пусть $k = (k_{m-1}, \ldots, k_1, k_0)_2$.
2) $Q \leftarrow O$.
3) Для i от $m - 1$ до 0
   $Q \leftarrow 2Q$.
   Если $k_i = 1$, то $Q \leftarrow Q + P$.
4) Возвращаем Q.

Пример

Вход: $E: y^2 = x^3 + x + 1$ кривая над $GF(5)$, $k = 13$, и $P = (0, 1)$. $Q = O$, $k = (1101)_2$.

1) Вычисляем:
   $Q = O$.
Так как $k_3 = 1$, $Q = Q + P = (0,1)$.
2) Вычисляем:
   $Q = 2Q = 2(0, 1) = (4, 2)$.
Так как $k_2 = 1$, $Q = Q + P = (4, 2) + (0,1) = (2, 1)$.
3) Вычисляем:
   $Q = 2Q = 2(2, 1) = (2, 4)$.
Так как $k_1 = 0$, $Q = (2, 4)$.
4) Вычисляем:
   $Q = 2Q = 2(2, 4) = (2, 1)$.
Так как $k_0 = 1$, $Q = Q + P = (2, 1) + (0,1) = (3, 4)$.

Выход: $Q = k \times P = (3, 4)$.

11.5. Задача дискретного логарифмирования на эллиптической кривой

Наиболее распространенные асимметричные криптосистемы до недавнего времени строились на основе задачи дискретного логарифмирования (DLP – discrete logarithm problem) над группой. DLP для группы $(G, \diamondsuit)$ определяется следующим образом. Пусть x, y – элементы G, причем y получен путем последовательного применения операции $\diamondsuit$ к элементу x:

$$y = x \diamondsuit x \diamondsuit \ldots \diamondsuit x.$$

Обозначим число членов в правой части этого выражения через n. В этом случае можно записать данное выражение в виде $y = x^n$, а задача дискретного логарифмирования заключается в нахождении n, зная только x, y и характеристики группы $(G, \diamondsuit)$, такие как порядок группы, определение операции $\diamondsuit$ и др.

Сложность DLP в группе $(G, \diamondsuit)$ зависит от свойств группы. В группе общего вида DLP является *сложной* задачей в том смысле, что сложность лучших известных алгоритмов ее решения пропорциональна квадратному корню от размера группы. Однако некоторые группы имеют особую структуру, позволяющую реализовать более эффективные алгоритмы, что делает эти группы не столь подходящими для криптографических применений.
В качестве примера использования проблемы дискретного логарифмирования в криптографии можно привести протокол выработки общего секретного ключа Диффи–Хеллмана.

1. Абоненты A и B договариваются об использовании группы $(G, \diamondsuit)$ и некоторого базового элемента g. Затем абонент A формирует случайное число a, а абонент B – случайное число b.
2. Абонент A вычисляет g^a и пересылает результат абоненту B. Абонент B вычисляет g^b и пересылает абоненту A. Сложность решения задачи дискретного логарифмирования не позволяет противнику узнать a или b на основании данных, передаваемых по каналу связи.

3. Абонент A вычисляет $(g^b)^a = g^{ab}$, а абонент B – $(g^a)^b = g^{ab}$. В результате оба абонента получают один и тот же результат.
4. Теперь абоненты A и B могут на основании *секретного* элемента g^{ab} сформировать общий секретный ключ, например, для использования в алгоритме поточного шифрования. Для этого может быть использована криптографическая хеш-функция.

Пассивный противник W знает только g^a и g^b. Для вычисления g^{ab} ему необходимо узнать одно из чисел a или b, что требует от него вычисления дискретного логарифма, а это по условию является сложной задачей.

До этого момента мы не определяли, в какой именно группе производятся операции. Очевидно, интерес представляют группы, в которых дискретное логарифмирование – сложная задача. Уже на протяжении двух десятков лет в этих целях используются мультипликативные группы конечных полей $GF(p)$ и $GF(2^n)$. Речь идет, в частности, о стандартах электронной цифровой подписи ГОСТ Р34.10-94 и DSA, криптосистеме Эль-Гамаля и др. Для этих групп были найдены некоторые атаки, позволяющие вычислить дискретный логарифм за субэкспоненциальное время*, однако указанные криптосистемы все еще имеют достаточную стойкость для практического применения в случае использования простых чисел разрядностью 1024 и более.

Задача, которую вынужден решать противник при использовании криптосистемы на базе эллиптических уравнений, – своего рода задача «дискретного логарифмирования на эллиптической кривой» (ECDLP), формулируется следующим образом. Даны точки P и Q на эллиптической кривой порядка r, где r – число точек на кривой. Необходимо найти единственную точку x такую, что $P = xQ$.

* Иначе говоря, сложность этих атак равна $O(e^k)$, где k – разрядность ключа шифрования.

11.6. Эллиптические кривые над конечными полями $GF(p)$

Недостатками использования вещественных чисел в криптографических целях являются неудобство их представления в цифровом виде и сложность оценки необходимой для их хранения памяти. Таким образом, нужно найти поле, удобное для машинного представления эллиптических кривых.

В середине 1980-х годов В. Миллер и Н. Коблиц независимо предложили использовать группу точек эллиптической кривой, определенной над конечным полем, для построения асимметричных криптосистем [17, 35, 42, 43, 45, 47]. До сих пор неизвестны методы решения DLP в этой группе, имеющие сложность менее квадратного корня из размера группы, за исключением некоторых вырожденных семейств кривых.

Элементы конечных полей обладают огромным преимуществом перед действительными числами при цифровой обработке. Их двоичное представление имеет фиксированную длину, а большинство операций эффективно реализуются на микропроцессорах общего назначения.

Если задать эллиптическую кривую над полем $GF(p)$, как это было сделано ранее для множества вещественных чисел, то мы получим ряд интересных особенностей. Операция сложения точек эллиптической кривой будет определяться точно так же, как для кривой над множеством вещественных чисел, с тем лишь отличием, что все операции будут выполняться с целыми числами по модулю p. Следовательно, для реализации операций с подобными кривыми можно использовать уже существующие программные и аппаратные библиотеки для модульных вычислений.

Пусть $p = 5$, $a = b = 1$, $4a^3 + 27b^2 = 4 + 4 = 8 \neq 0$. Рассмотрим эллиптическую кривую

$$E: y^2 = x^3 + x + 1 \text{ (рис. 11.5).}$$

$E(GF(5))$ состоит из следующих точек (рис. 11.6):

$$P = (0, 1),\ 2P = (4, 2),\ 3P = (2, 1),\ 4P = (3, 4),\ 5P = (3, 1),$$
$$6P = (2, 4),\ 7P = (4, 3),\ 8P = (0, 4),\ 9P = O,$$

при этом $10P = P$.

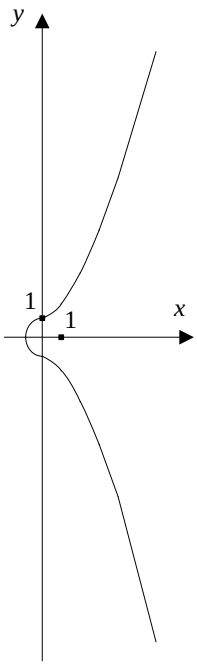


Рис. 11.5. Эллиптическая кривая, соответствующая уравнению $y^2 = x^3 + x + 1$

11.7. Эллиптические кривые над конечными полями $GF(2^m)$

Другой, также широко распространенный способ представления эллиптических кривых – использование поля $GF(2^m)$. В отличие от поля $GF(p)$ возможны разные представления элементов поля $GF(2^m)$, как следствие реализация операций в этом поле может быть также различной. Описание операций сложения и умножения в поле $GF(2^m)$ было дано в главе 3. Обычно в этом поле рассматривают кривые вида

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

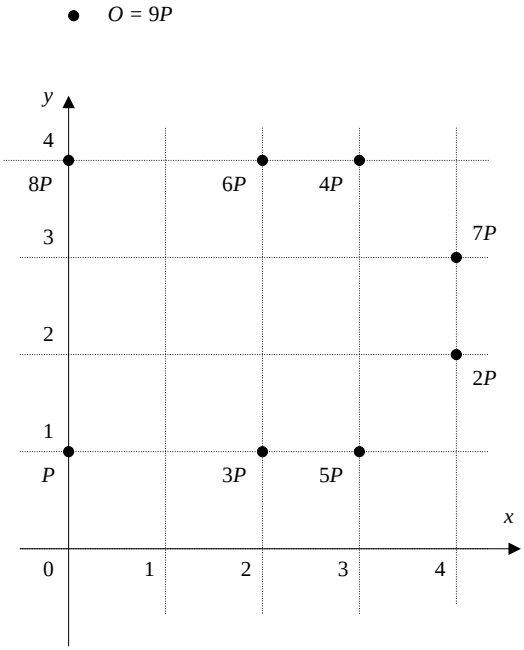


Рис. 11.6. Аддитивная группа на $E(Z_5)$

В случае использования поля $GF(2^m)$ нельзя провести очевидную аналогию с привычными вещественными числами и построить графики. Но есть одно важнейшее преимущество: реализация операций в этом поле (как программная, так и аппаратная) значительно эффективнее, чем в поле $GF(p)$. К недостаткам можно отнести то, что вычисления с большими числами в поле $GF(2^m)$ не так широко распространены, как в поле $GF(p)$, что может потребовать разработки новых криптомодулей для работы с этим видом эллиптических кривых.

Если говорить о криптографической стойкости, то при условии надлежащего выбора параметров эллиптические кривые над обоими типами полей практически эквивалентны.

Операции сложения точек на несуперсингулярной кривой над $GF(2^m)$ обладают следующими свойствами:

1) $P + O = O + P = P$, для всех точек $P \in E(GF(2^m))$;
2) для каждой точки $P = (x, y)$, $P \in E(GF(2^m))$ существует точка $Q = (x, x + y)$, $Q \in E(GF(2^m))$, такая что $P + Q = O$;

точка Q называется обратным элементом и обозначается как $(-P)$;

3) операция сложения двух точек на кривой:
если $P = (x_1, y_1) \in E(GF(2^m))$ и $Q = (x_{2,} y_2) \in E(GF(2^m))$, где $x_1 = 0$, то $P + Q = O$;
если $x_1 \neq 0$ и $Q \neq -P$, то $(x_3, y_3) = (x_{1,} y_1) + (x_2, y_2)$, где

$$x_3 = \left(\lambda^2 + \lambda + x_1 + x_2 + a\right)\left(GF(2^m)\right);$$
$$y_3 = \left(\lambda\left(x_1 + x_3\right) + x_3 + y_1\right)\left(GF(2^m)\right);$$
$$\lambda = \frac{y_1 + y_2}{x_1 + x_2}\left(GF(2^m)\right).$$

если $P = (x_1, y_1) \in E(GF(2^m))$, где $x_1 = 0$, то $2P = O$;
если $x_1 \neq 0$, $P + P = 2P = (x_3, y_3)$, где

$$x_3 = \left(\lambda^2 + \lambda + a\right)\left(GF(2^m)\right);$$
$$y_3 = \left(x_1^2 + (\lambda + 1)x_3\right)\left(GF(2^m)\right);$$
$$\lambda = x_1 + \frac{y_1}{x_1}\left(GF(2^m)\right).$$

Пример. Рассмотрим несуперсингулярную эллиптическую кривую $E_2 : y^2 + xy = x^3 + g^4x^2 + 1$ над $GF(2^4)$, где $a = g^4$ и $b = g^0 = 1$. Пусть полином, неприводимый над $GF\left(2^4\right)$, имеет вид $f(x) = x^4 + x + 1$. Элемент g = (0010) – генератор элементов поля. Степени элемента g:

g^0	0001	g^8	0101
g^1	0010	g^9	1010
g^2	0100	g^{10}	0111
g^3	1000	g^{11}	1110
g^4	0011	g^{12}	1111
g^5	0110	g^{13}	1101
g^6	1100	g^{14}	1001
g^7	1011	g^{15}	0001

Рассмотрим $E_2\left(GF\left(2^4\right)\right)$, состоящую из точки O, а также следующих точек:

$(1,\ g^{13}), (g^3,\ g^{13}), (g^5,\ g^{11}), (g^6,\ g^{14}), (g^9,\ g^{13}), (g^{10},\ g^8),$

$(g^{12},\ g^{12}), (1,\ g^6), (g^3,\ g^8), (g^5,\ g^3), (g^6,\ g^8), (g^9,\ g^{10}),$

$(g^{10},\ g), (g^{12}, 0), (0, 1).$

Пусть $P = \left(g^6, g^{14}\right)$ и $Q = \left(g^3, g^8\right)$. Найдем $P + Q$ и $2P$.

Пусть $P + Q = (x_3, y_3)$, тогда

$$\lambda = \left(g^{14} + g^8\right) \times \left(g^6 + g^3\right)^{-1} = g^6 \times g^{-2} = g^4,$$

$$x_3 = g^8 + g^4 + g^6 + g^3 + g^4 = 1,$$

$$y_3 = g^4 \times g^{13} + 1 + g^{14} = g^6.$$

Таким образом, $P + Q = (1, g^6)$.

Пусть $2P = P + P = (x_3, y_3)$, тогда

$$\lambda = g^6 + g^{14} \times g^{-6} = g^6 + g^8 = g^{14},$$

$$x_3 = g^{13} + g^{14} + g^4 = g^{10},$$

$$y_3 = g^{12}\left(g^{14} + 1\right) \times g^{10} = g^{12} + g^3 \times g^{10} = g.$$

Таким образом, $2P = (g^{10}, g)$.

11.8. Параметры криптографической схемы

При использовании эллиптической кривой в криптографических целях следует определить совокупность параметров, общих для всех участников протокола.

Для эллиптических кривых E над полем $GF(p)$ эти параметры включают:

простое число p, по модулю которого производятся вычисления;

коэффициенты $a, b \in GF(p)$ уравнения эллиптической кривой E;

целое число m – порядок группы точек эллиптической кривой E;

простое число q – порядок циклической подгруппы группы точек эллиптической кривой E, причем $m = kq$, где k – некоторое положительное целое число;

базовая точка $P \neq O$ эллиптической кривой E с координатами (x_p, y_p), удовлетворяющая условию $qP = O$.

Для кривых E над полем $GF(2^n)$ эти параметры включают:

порядок поля n;

неприводимый многочлен $f(x)$ степени n;

коэффициенты $a, b \in GF(2^n)$ уравнения эллиптической кривой E,

целое число m – порядок группы точек эллиптической кривой E;

простое число q – порядок циклической подгруппы группы точек эллиптической кривой E, причем $m = kq$, где k – некоторое положительное целое число;

базовая точка $P \neq O$ эллиптической кривой E с координатами (x_p, y_p), удовлетворяющая условию $qP = O$.

Правильный выбор этих величин – нетривиальная задача. Как правило, описание того или иного криптографического протокола включает либо алгоритм формирования параметров кривых, либо фиксированную совокупность допустимых эллиптических кривых и базовых точек.

11.9. Схема гибридного шифрования ECIES

Криптографические примитивы с открытым ключом имеют множество незаменимых свойств, однако скорость выполнения преобразований в любом случае оказывается существенно ниже, чем при использовании симметричных шифров, имеющих сопоставимую криптографическую стойкость. По этой причине на практике редко используют асимметричные криптосистемы непосредственно для шифрования данных, хотя принципиальная возможность и существует.

Если вспомнить о задаче дискретного логарифмирования, то в учебниках обычно приводится алгоритм асимметричного шифрования Эль-Гамаля, однако он практически не применяется в реальных системах, в частности из-за нестойкости по отношению к атакам с выбранным шифротекстом. Аналогичная схема для эллиптических кривых (EC El-Gamal) иногда упоминается в литературе, но лишь в качестве иллюстрации.

Гораздо больший интерес представляют гибридные криптосистемы, в которых асимметричные алгоритмы применяют лишь для формирования ключа, который уже используется непосредственно при шифровании данных.

Кратко опишем распространённый алгоритм ECIES, использующий преобразования на основе эллиптических кривых.

Пусть абонент A намерен послать сообщение M абоненту B. Перед выполнением шифрования абоненты должны договориться о параметрах шифрования, включающих:

функцию выработки ключа KDF (на текущий момент в стандарте поддерживается единственный алгоритм, основанный на хеш-функции SHA-1);

алгоритм MAC (поддерживаются два алгоритма, оба основаны на хеш-функции SHA-1);

симметричный криптоалгоритм ENC (используется либо одна из разновидностей алгоритма 3DES, либо сложение открытого текста с ключом с помощью операции XOR[*]);

схему распределения ключей DH (поддерживается две разновидности EC DH);

совокупность параметров криптографической схемы T (см. раздел 11.1.4);

тип представления точек эллиптической кривой (обычное или так называемое *сжатое* представление).

Затем абонент B должен сформировать пару ключей (d_B, Q_B) в соответствии с параметрами T, а абонент A – получить открытый ключ Q_B.

Теперь абонент A может зашифровать сообщение M по следующему алгоритму:

На основе параметров T формируется случайным образом пара сеансовых ключей (k, R), где $R = (x_R, y_R)$.

С помощью выбранного алгоритма DH получается общий элемент z на основе закрытого сеансового ключа k и открытого ключа Q_B.

[*] Заметим, что при использовании для шифрования операции XOR мы фактически получаем поточный шифр, в котором для генерации гамма-последовательности применяется функция выработки ключей KDF.

С помощью выбранной функции KDF формируется ключевая информацию K, длина которой равна сумме длин ключа шифрования и ключа MAC. K разделяется на ключ шифрования EK и ключ MAC MK.

Сообщение M зашифровывается с помощью выбранного алгоритма ENC и ключа EK. В результате получаем шифротекст EM.

Формируем MAC-код D в соответствии с выбранным алгоритмом MAC.

Возвращаем результат $C = R||EM||D$.

Абонент B может расшифровать сообщение следующим образом:

C разделяется на сеансовый ключ R, шифротекст EM и MAC-код D.

С помощью выбранного алгоритма DH получается общий элемент z на основе закрытого ключа d_B и открытого сеансового ключа R.

С помощью выбранной функции KDF формируется ключевая информация K, длина которой равна сумме длин ключа шифрования и ключа MAC. K разделяется на ключ шифрования EK и ключ MAC MK.

D проверяется в соответствии с выбранным алгоритмом MAC. Если MAC-код неверен, возвращаем ошибку.

Выполняется расшифрование с помощью выбранного алгоритма шифрования ENC, в результате чего получается исходное сообщение M.

Возвращается M.

11.10. Эллиптические алгоритмы генерации ПСЧ

В криптологии эллиптические кривые используются не только для построения криптографических протоколов. Например, существуют алгоритмы факторизации больших чисел, использующие эллиптические кривые. Кроме того, было предложено несколько алгоритмов генерации псевдослучайных чисел, основанных на свойствах эллиптических кривых [38–40].

Как правило, речь идет о реализации известных алгоритмов для нового множества – группы точек эллиптической кривой.

11.10.1. Эллиптические алгоритмы формирования ПСЧ на основе линейных конгруэнтных генераторов

В 1994 г. S. Hallgren представил линейный конгруэнтный генератор (LCG, Linear Congruential Generator) над эллиптической кривой. Это – быстрый, но небезопасный генератор. Формируемые на основе LCG последовательности были названы LCG-EC-последовательностями.

Рассмотрим линейный конгруэнтный генератор над Z_n. Уравнение линейного конгруэнтного генератора имеет вид

$$x_{i+1} = ax_i + b \pmod{n},$$

где a, b, n – константы; x_0 – начальное состояние генератора; $a, b, x_0, n \in Z_n$. Свойства последовательности зависят от выбранных параметров n, a, b и x_0.

Линейный конгруэнтный генератор производит последовательность $\{x_i\}$ максимально возможного периода n тогда и только тогда, когда выполняется каждое из следующих условий:

b – простое число относительно n;

$a \equiv 1 \pmod{p}$ для каждого простого делителя p числа n;

$a \equiv 1 \pmod{4}$, если n кратно 4.

Рассмотрим линейный конгруэнтный генератор над эллиптической кривой. Переключения генератора LCG-EC-последовательности X_0, X_1, X_2, … определяются рекуррентным соотношением $X_{i+1} = aX_i + B$ над эллиптической кривой, где a – константа, X_0 – начальная точка, B – фиксированная точка. Выходная псевдослучайная последовательность суть последовательность младших бит y-координат точек X_i.

Алгоритм генерации ПСЧ

1. Выбираем конечное поле $GF(q)$.
2. Выбираем кривую E.
3. Выбираем линейный конгруэнтный генератор, например, $X_{i+1} = aX_i + B$.
4. Выбираем фиксированное целое число a, начальную точку X_0 и фиксированную точку B, причем $aX_0 + B \neq X_0$.

5. Вычисляем последовательность состояний генератора $X_0, X_1, X_2, \ldots$, используя формулу $X_{i+1} = aX_i + B$.
6. Формируем выходную двоичную ПСП из младших бит y-координат точек X_i.

Аналогичным образом могут быть получены EC-последовательности для других типов конгруэнтных генераторов, например инверсивного. Предлагаются и другие генераторы, использующие свойства эллиптических кривых, однако в реальных системах подобные алгоритмы пока не используются.

Пример генератора ICG-E-последовательности. Рассмотрим эллиптическую кривую $y^2 = x^3 + x + 1$ над $GF(23)$. Пусть $a = 3$, $B = (11, 3)$, и $X_0 = (5, 4)$. Последовательность точек, произведенных по формуле $X_{i+1} = aX_i^{-1} + B$, показана в табл. 11.1.

Таблица 11.1

Пример ICG-EC-последовательности

i	$X_{i+1} = aX_i^{-1} + B$
0	(5, 4)
1	(3, 13)
2	(6, 4)
3	(9, 16)
4	(13, 7)
5	(19, 5)
6	(7, 11)
7	(1, 7)
8	(17, 20)
9	(0, 22)
10	(12, 4)
11	(18, 3)
12	(5, 4)

Выходная псевдослучайная ICG-EC-последовательность:

0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, …

имеет период 12.

11.10.2. Эллиптические алгоритмы формирования ПСЧ на основе регистров сдвига с линейными обратными связями

Рассмотрим конструкцию двоичных LFSR-EC-последовательностей.

Алгоритм формирования ПСЧ

1. Выбираем конечное поле $GF(q)$ (или $GF(2^m)$).
2. Выбираем эллиптическую кривую E.
3. Выбираем точку P порядка r на кривой E.
4. Выбираем примитивный многочлен $f(x)$ над $GF(q)$ (или $GF(2^m)$). Строим LFSR, используя $f(x)$ в качестве характеристического многочлена.
5. Вычисляем LFSR-EC-последовательность $\underline{P}$ с использованием М-последовательности q с выхода LFSR.
6. Преобразуем LFSR-EC-последовательность $\underline{P}$ в двоичную последовательность $\underline{u}$ с использованием оператора отображения $GF(q) \to GF(2)$ (или $GF(2^m) \to GF(2)$) (блока пространственного сжатия (БПС)).

Рассмотрим кривую E_1: $y^2 = x^3 + x + 4$ над $GF(23)$. Эта кривая имеет порядок 29. Пусть $P = (4, 7)$ – точка на E_1, порядок которой равен 29. Пусть начальное состояние $Q = O$, $P_0 = P$, $P_1 = 2P$. Выберем $f(x) = x^2 - 28x - 26 \in Z_{29}[x]$. LFSR-EC-последовательность $\underline{P} = \{P_k\}$ вычисляется с использованием LFSR, представленного на рис. 11.7. Точки показаны в табл. 11.2.

$$P_k = 28P_{k-1} + 26P_{k-2} = a_kP,\ k = 2, 3, \ldots ,$$

где $a_k = 28a_{k-1} + 26a_{k-2}$ ($k = 2, 3, \ldots$) – LFSR-последовательность над Z_{29} с начальным состоянием $(a_0, a_1) = (1, 2)$.

Последовательность LFSR над Z_{29} имеет вид

$\underline{a}$ = (1, 2, 24, 28, 16, 16, 23, 16, 2, 8, 15, 19, 23, 7, 11, 26, 28, 10, 22, 6, 15, 25, 17, 24,12, …).

LFSR-EC-последовательность имеет вид

$\underline{P}$ = (P, $2P$, $24P$, $28P$, $16P$, $16P$, $23P$, $16P$, $2P$, $8P$, $15P$, $19P$, $23P$, $7P$, $11P$, $26P$, $28P$, $10P$, $22P$, $6P$, $15P$, $25P$, $17P$, $24P$, $12P$, …).

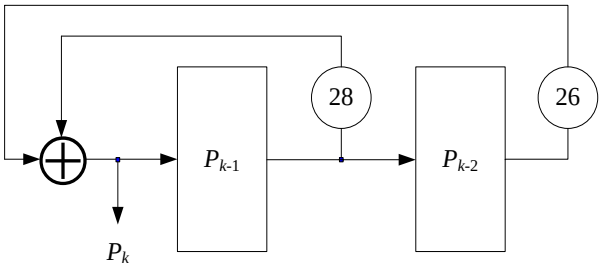


Рис. 11.7. Схема LFSR 1

Пусть g – корень многочлена $f(x) = x^5 + x^2 + 1$ и $GF(2^5)$ – поле, произведенное g. Рассмотрим несуперсингулярную кривую

$$E_2: y^2 + xy = x^3 + g^9x^2 + g^{15} \text{ над } GF(2^5).$$

Эта кривая имеет порядок 26. Пусть R точка на E_2, порядок которой равен 13. Выберем $f(x) = x^2 - 12x - 11 \in Z_{13}[x]$. Пусть начальное состояние $Q = O$, $P_0 = R$ и $P_1 = 2R$. LFSR-EC-последовательность $\underline{P} = \{P_k\}$ формируется с использованием LFSR, представленного на рис. 11.8. Точки показаны в табл. 11.3.

$$P_k = 12P_{k-1} + 11P_{k-2} = a_kR,\ k = 2, 3, ...,$$

где $a_k = 12a_{k-1} + 11a_{k-2}$ ($k = 2, 3,\ldots$) – LFSR-последовательность с начальным состоянием $(a_0, a_1) = (1, 2)$.

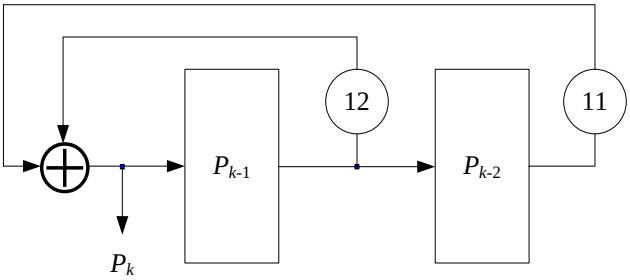


Рис. 11.8. Схема LFSR 2

Таблица 11.2

Значения iP

i	iP	i	iP	i	iP	i	iP
1	(4, 7)	9	(9, 12)	16	(0, 21)	23	(1, 12)
2	(10, 18)	10	(11, 9)	17	(14, 5)	24	(8, 15)
3	(13, 11)	11	(17, 9)	18	(17, 14)	25	(15, 17)
4	(15, 6)	12	(14, 18)	19	(11, 14)	26	(13, 12)
5	(8, 8)	13	(0, 2)	20	(9, 11)	27	(10, 5)
6	(1, 11)	14	(22, 5)	21	(18, 14)	28	(4, 16)
7	(7, 20)	15	(22, 18)	22	(7, 3)	29	O
8	(18, 9)						

Последовательность LFSR над Z_{13} имеет вид

$\underline{a}$ = (1, 2, 9, 0, 8, 5, 5, 11, 5, 12, 4, 11, 7, 10, 2, 4, 5, 0, 3, 10, 10, 9, 10, 11, ...).

LFSR-EC-последовательность имеет вид

$\underline{P}$ = (R, 2R, 9R, O, 8R, 5R, 5R, 11R, 5R, 12R, 4R, 11R, 7R, 10R, 2R, 4R, 5R, O, 3R, 10R, 10R, 9R, 10R, 11R, ...).

Таблица 11.3

Значения iP

i	iP	i	iP	I	iP
1	(g, g^7)	10	(g^{24}, g^{17})	19	(g^{14}, g^{10})
2	(g^{21}, g)	11	(g^2, g^{18})	20	(g^9, g^{29})
3	(g^{30}, g^{29})	12	(g^{22}, g^5)	21	(g^{23}, g^{25})
4	$(g^{26}, 1)$	13	$(0, g^{23})$	22	(g^{26}, g^{28})
5	(g^{23}, g^{28})	14	(g^{22}, g^4)	23	(g^{30}, g^{16})
6	(g^9, g^{17})	15	(g^2, g^{11})	24	(g^{21}, g^9)
7	(g^{14}, g^{20})	16	(g^{24}, g^8)	25	(g, g^{28})
8	$(1, g^{27})$	17	(g^{28}, g^2)	26	O
9	(g^{28}, g^{30})	18	$(1, g^6)$		

11.11. Сравнение систем на основе эллиптических кривых с другими асимметричными криптосистемами

Любая асимметричная криптосистема строится на основе некоторой сложной математической задачи. На сегодняшний день наибольшее распространение получили задачи факторизации, дискретного логарифмирования в конечном поле и дискретного логарифмирования на эллиптической кривой.

Долгие годы неофициальным стандартом на шифрование с открытым ключом являлась система RSA, основанная на про-

блеме разложения на множители больших целых чисел. На сегодняшний день для обеспечения надежности этой криптосистемы приходится использовать числа разрядностью более 1024 бит. В системах на основе дискретного логарифмирования в конечном поле ситуация лучше, но не намного. Однако применение эллиптических кривых позволяет использовать ключи, размер которых в несколько раз меньше. При реализации арифметики на эллиптических кривых для записи чисел, как правило, бывает достаточно 256 разрядов. В результате типовые криптографические операции, такие, как формирование подписи или выработка общего секретного ключа требуют в несколько раз меньших вычислительных затрат, чем при использовании других асимметричных криптоалгоритмов. Это особенно актуально для мобильных устройств и смарт-карт, для которых криптографические методы становятся все более актуальными.

Кроме того, уменьшение стойкости системы RSA с течением времени вызвано не только экспоненциальным повышением производительности компьютеров, но и совершенствованием методов факторизации больших чисел. Проблема дискретного логарифмирования (как в конечном поле, так и на эллиптической кривой) выглядит в этом отношении более перспективно.

В последние годы прослеживается четкая тенденция – во многие международные и национальные стандарты стали включаться криптоалгоритмы, основанные на свойствах эллиптических кривых. Это означает, что такие системы уже достаточно исследованы, чтобы применять их в критических областях, таких как финансовые операции, дипломатическая переписка и т.д. При этом использование эллиптических кривых позволяет экономить вычислительные ресурсы, что приводит к упрощению и удешевлению конечной аппаратуры.

Контрольные вопросы

1. Сформулируйте задачу дискретного логарифмирования для группы точек эллиптической кривой.
2. Сравните криптосистемы RSA и ECCS по следующим параметрам: криптостойкость, быстродействие, эффективность программной и аппаратной реализации.

3. Приведите пример группы точек на эллиптической кривой над: а) $GF(p)$; б) $GF(2^m)$.
4. Приведите пример эллиптического генератора ПСЧ на основе РСЛОС.
5. Перечислите параметры криптографической схемы, основанной на свойствах эллиптической кривой.

ГЛАВА 12. КРИПТОСИСТЕМЫ, ОСНОВАННЫЕ НА СВОЙСТВАХ ЭЛЛИПТИЧЕСКИХ КРИВЫХ

Множество криптографических протоколов можно разделить на два больших класса: базовые протоколы и протоколы высшего уровня (соответственно примитивные и прикладные протоколы). К базовым можно отнести протоколы распределения ключей, электронной подписи, разделения секрета и т. п. Протоколы высшего уровня основываются на базовых протоколах. Например, если говорить о гипотетическом протоколе проведения банковских транзакций, то для него потребуются как минимум протоколы электронной подписи и аутентификации сообщений. Здесь необходимо отметить, что замена одного протокола электронной подписи другим (с аналогичной стойкостью) не повлияет ни на безопасность протокола, ни на его функциональность.

Если говорить о протоколах, основанных на свойствах эллиптических кривых, то отличия от старых проверенных схем на основе проблемы дискретного логарифмирования можно проследить лишь на базовых протоколах, да и то не на всех, так как фактически отличия заключаются только во множестве элементов, над которым производятся вычисления.

В этой главе рассматриваются несколько важнейших базовых схем, основанных на эллиптических кривых. Сейчас наблюдается тенденция постепенного перехода систем на основе дискретного логарифма к эллиптическим кривым, причем для большинства существующих протоколов это происходит совершенно безболезненно.

12.1. Схема симметричного шифрования на эллиптических кривых

Зашифрование информации (абонент A защищает сообщение M, предназначенное для абонента B) (рис. 12.1):

A преобразовывает открытый текст M в точки эллиптической кривой;

A выбирает секретный ключ K;

A вычисляет закрытый текст C

$$C = M + K.$$

Расшифрование информации (абонент *B* преобразует закрытый текст *C*) (рис. 12.2):

B вычисляет обратный ключ (–*K*).

B восстанавливает открытый текст *M*

$M = C + (-K)$.

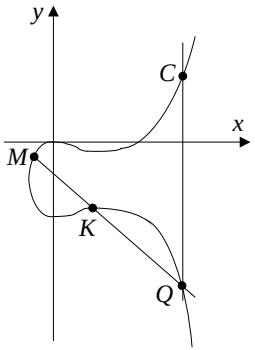


Рис. 12.1.
Прямое преобразование

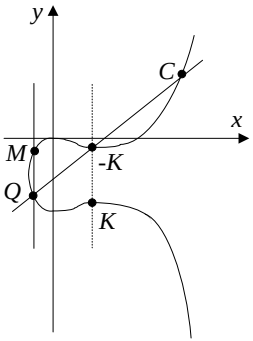


Рис. 12.2.
Обратное преобразование

Пример. Пусть задана кривая *E* (*GF*(23)) вида $y^2 = x^3 + x + 1$.

Зашифрование информации (абонент *A* защищает сообщение *M*, предназначенное для абонента *B*):

A преобразовывает открытый текст *M* в точку эллиптической кривой

$M = (12, 19)$;

A выбирает секретный ключ $K = (13, 7)$;

A вычисляет закрытый текст *C*

$C = M + K = (12, 19) + (13, 7) = (4, 0)$.

Расшифрование информации (абонент *B* восстанавливает закрытый текст *C*, полученный от абонента *A*):

B вычисляет обратный ключ (–*K*)

$-K = (13, -7) \bmod 23 = (13, 16)$;

B восстанавливает открытый текст *M*

$M = C + (-K) = (4, 0) + (13, 16) = (12, 19)$.

12.2. Схема асимметричного шифрования на эллиптических кривых

В этом и последующих алгоритмах должны быть определены следующие открытые параметры, общие для всех пользователей системы:

конечное поле $GF(p)$;

эллиптическая кривая $E(GF(p))$;

порядок n, который является простым числом;

базовая точка G порядка n.

Генерация ключей. Каждый пользователь системы генерирует пару ключей следующим образом:

выбирается случайное целое число d, $1 < d < n-1$;

вычисляется точка $Q = d \times G$.

Секретным ключом пользователя является число d, открытым ключом – точка Q.

Зашифрование информации (абонент A защищает сообщение M, предназначенное для абонента B):

A выбирает случайное целое число k, $1 < k < n-1$;

A вычисляет точку $(x_1, y_1) = k \times G$;

A вычисляет точку $(x_2, y_2) = k \times Q$, используя открытый ключ Q пользователя B;

A вычисляет $c_1 = M \oplus x_2$;

закрытые данные: $C = (x_1, y_1, c_1)$.

Расшифрование информации (абонент B восстанавливает закрытые данные $C = (x_1, y_1, c_1)$, полученные от абонента A):

B вычисляет точку $(x_2, y_2) = d \times (x_1, y_1)$, используя свой секретный ключ d;

B восстанавливает исходное сообщение $M = c_1 \oplus x_2$.

Пример

Параметры:

конечное поле $GF(23)$;

эллиптическая кривая E ($GF(23)$) вида $y^2 = x^3 + x + 1$;

базовая точка G = (13, 7);

порядок n = 7.

Генерация ключей:

выбирается случайное целое число $d = 3$;

вычисляется точка $Q = d \times G = 3(13, 7) = (17, 3)$.

Секретным ключом пользователя является число $d = 3$, открытым ключом – точка $Q = (17, 3)$. Пусть сообщение $M = 9$.

Зашифрование информации (абонент *A* защищает сообщение *M*, предназначенное для абонента *B*):

A выбирает случайное целое число $k = 4$;

A вычисляет точку $(x_1, y_1) = k \times G = 4(13, 7) = (17, 20)$;

A вычисляет точку $(x_2, y_2) = k \times Q = 4(17, 3) = (5, 19)$;

A вычисляет $c_1 = M \oplus x_2 = 9 \oplus 5 = 12$;

закрытые данные: $C = (x_1, y_1, c_1) = (17, 20, 12)$.

Расшифрование информации (абонент *B* восстанавливает закрытые данные $C = (x_1, y_1, c_1) = (17, 20, 12)$, полученные от абонента *A*):

B вычисляет точку $(x_2, y_2) = d \times (x_1, y_1) = 3(17, 20) = (5, 19)$;

B восстанавливает исходное сообщение

$$M = c_1 \oplus x_2 = 12 \oplus 5 = 9.$$

12.3. Схема электронной цифровой подписи на эллиптических кривых

Выбор параметров и процедура генерации ключей описаны в разделе 12.2.

Формирование подписи (абонент *A* подписывает сообщение *M*):

A вычисляет хеш-образ сообщения $e = h(M)$;

A выбирает случайное целое число k, $1 < k < n - 1$;

A вычисляет точку $(x_1, y_1) = k \times G$;

A вычисляет $r = x_1 + e \bmod n$;

A вычисляет $s = k - (r \times d) \bmod n$, используя свой секретный ключ *d*. В том случае, если $r = 0$ или $s = 0$, выбор *k* осуществляется заново;

подписью сообщения является пара (r, s).

Проверка подписи (абонент *B* проверяет подпись (r, s) абонента *A* на сообщении *M*):

B вычисляет точку $(x_1, y_1) = (s \times G) + (r \times Q)$, используя открытый ключ *Q* абонента *A*;

B вычисляет хеш-образ сообщения $e = h(M)$;

B вычисляет $r_1 = x_1 + e \bmod n$;

подпись считается верной, если $r_1 = r$.

Пример

Параметры:

конечное поле *GF*(23);

эллиптическая кривая *E* (*GF*(23)) вида $y^2 = x^3 + x + 1$;

базовая точка *G* = (13, 7);

порядок *n* = 7.

Генерация ключей:

выбирается случайное целое число *d* = 3;

вычисляется точка $Q = d \times G$ = 3(13, 7) = (17, 3).

Секретным ключом пользователя является число *d* = 3, открытым ключом – точка *Q* = (17, 3). Пусть сообщение *M* = 10111001.

Формирование подписи (абонент *A* подписывает сообщение *M*):

A вычисляет хеш-образ сообщения $e = h(M)$ = 6;

A выбирает случайное целое число *k* = 4;

A вычисляет точку $(x_1, y_1) = k \times G$ = 4(13, 7) = (17, 20);

A вычисляет $r = x_1 + e \bmod n$ = (17 + 6) mod 7 = 2;

A вычисляет $s = k - (r \times d) \bmod n$ = 4 – (2 × 3) mod 7 = 5;

Подписью сообщения является пара (*r*, *s*) = (2, 5).

Проверка подписи (абонент *B* проверяет подпись (*r*, *s*) = (2, 5) абонента *A* на сообщении *M*):

B вычисляет точку

$(x_1, y_1) = (s \times G) + (r \times Q)$ = 5(13, 7) + 2(17, 3) = (17, 20);

B вычисляет хеш-образ сообщения $e = h(M)$ = 6;

B вычисляет $r_1 = x_1 + e \bmod n$ = (17 + 6) mod 7 = 2;

подпись считается верной в случае r_1 = 2.

12.4. Схема ЭЦП на эллиптических кривых (ECDSA)

Выбор параметров и процедура генерации ключей описаны в разделе 12.2.

Формирование подписи (абонент *A* подписывает сообщение *M*):

A вычисляет хеш-образ сообщения $e = h(M)$;

A выбирает случайное целое число k, $1 < k < n - 1$;

A вычисляет точку $(x_1, y_1) = k \times G$ и $r = x_1 \bmod n$;

A вычисляет $s = k^{-1} \times (e + (r \times d)) \bmod n$, используя свой секретный ключ d; в случае, если $r = 0$ или $s = 0$, повторяется выбор k;

подписью сообщения являются (r, s).

Проверка подписи (абонент *B* проверяет подпись сообщения (r, s) абонента *A* под сообщением *M*):

B вычисляет хеш-образ сообщения $e = h(M)$;

B вычисляет $u = s^{-1} \times e \bmod n$ и $v = s^{-1} \times r \bmod n$;

B вычисляет точку $(x_1, y_1) = (u \times G) + (v \times Q)$, используя открытый ключ Q абонента *A*;

B вычисляет $r_1 = x_1 \bmod n$;

подпись считается верной, если $r_1 = r$.

Пример

Параметры:

конечное поле $GF(23)$;

эллиптическая кривая E ($GF(23)$) вида $y^2 = x^3 + x + 1$;

базовая точка G = (13, 7);

порядок n = 7.

Генерация ключей:

выбирается случайное целое число d = 3;

вычисляется точка $Q = d \times G$ = 3(13, 7) = (17, 3).

Секретным ключом пользователя является число d = 3, открытым ключом – точка Q = (17, 3). Пусть сообщение M = 10111001.

Формирование подписи (абонент *A* подписывает сообщение *M*):

A вычисляет хеш-образ сообщения $e = h(M)$ = 6;

A вычисляет случайное целое число k = 3;

A вычисляет точку $(x_1, y_1) = k \times G$ = 3(13, 7) = (17, 3) и $r = x_1 \bmod n$ = 17 mod 7 = 3;

A вычисляет

$$s = k^{-1} \times (e + (r \times d)) \bmod n = 3^{-1} \times (6 + (3 \times 3)) \bmod 7 =$$

$$= 5 \times (6 + 9) \bmod 7 = 5;$$

подписью сообщения являются (*r*, *s*) = (3, 5).

Проверка подписи (абонент *B* проверяет подпись сообщения (*r*, *s*) = (3, 5) абонента *A* под сообщением *M*):

B вычисляет хеш-образ сообщения $e = h(M)$ = 6;

B вычисляет

$$u = s^{-1} \times e \bmod n = 5^{-1} \times 6 \bmod 7 = 3 \times 6 \bmod 7 = 4$$

и

$$v = s^{-1} \times r \bmod n = 5^{-1} \times 3 \bmod 7 = 2;$$

B вычисляет точку

$$(x_1, y_1) = (u \times G) + (v \times Q) = 4(13, 7) + 2(17, 3) = (17, 3);$$

B вычисляет $r_1 = x_1 \bmod n$ = 17 mod 7 = 3;

подпись считается верной в случае $r_1 = 3$.

12.5. Протокол выработки общего секретного ключа (ECKEP)

Это протокол, где два абонента *A* и *B* устанавливают общий секретный ключ *K*, называемый сеансовым ключом. Он позволяет создать защищенный канал связи между абонентами и служит для решения задач обеспечения конфиденциальности и аутентичности информации.

Процедуры определения параметров системы и генерации ключей аналогичны рассмотренным ранее. Предположим, что общий секретный ключ вычисляется абонентами *A* и *B*. Абонент *A* имеет секретный ключ *a* и открытый ключ $Q_A = a \times G = (x_A, y_A)$, аналогично *b* и $Q_B = b \times G = (x_B, y_B)$ являются соответственно секретными и открытыми ключами абонента *B*.

Общий секретный ключ вычисляется в четыре этапа:

1) абонент *A* выполняет следующие действия:

выбирает случайное целое число k_A, $1 < k_A < n - 1$;

вычисляет точку $R_A = k_A \times G$;

вычисляет точку $(x_1, y_1) = k_A \times Q_B$;

вычисляет $s_A = k_A + (a \times x_A \times x_1) \bmod n$;

пересылает R_A абоненту *B*;

2) абонент *B* выполняет следующие действия:

выбирает случайное целое число k_B, $1 < k_B < n-1$;

вычисляет точку $R_B = k_B \times G$;

вычисляет точку $(x_2, y_2) = k_B \times Q_A$;

вычисляет $s_B = k_B + (b \times x_B \times x_2) \bmod n$;

пересылает R_B абоненту A;

3) абонент *A* выполняет следующие действия:

вычисляет точку $(x_2, y_2) = a \times R_B$.

Вычисляет общий секретный ключ

$$K = s_A(R_B + (x_B \times x_2 \times Q_B));$$

4) абонент *B* выполняет следующие действия:

вычисляет точку $(x_1, y_1) = b \times R_A$;

вычисляет общий секретный ключ

$$K = s_B(R_A + (x_A \times x_1 \times Q_A)),$$

что эквивалентно

$$K = s_A(R_B + (x_B \times x_2 \times Q_B)).$$

Пример

Параметры:

конечное поле *GF*(23);

эллиптическая кривая *E* (*GF*(23)) вида $y^2 = x^3 + x + 1$;

базовая точка *G* = (13, 7);

порядок *n* = 7.

Генерация ключей:

A выбирает случайное целое число a = 3;

A вычисляет точку $Q_A = a \times G = (x_A, y_A)$ = 3(13, 7) = (17, 3);

B выбирает случайное целое число b = 6;

B вычисляет точку

$$Q_B = b \times G = (x_B, y_B) = 6(13, 7) = (13, 16).$$

Общий секретный ключ вычисляется в четыре этапа:

1) абонент *A* выполняет следующие действия:

выбирает случайное целое число k_A = 4;

вычисляет точку $R_A = k_A \times G$ = 4(13, 7) = (17, 20);

вычисляет точку

$(x_1, y_1) = k_A \times Q_B$ = 4(13, 16) = (17, 3);

вычисляет

$s_A = k_A + (a \times x_A \times x_1) \bmod n$ =

= 4 + (3 × 17 × 17) mod 7 = 3;

пересылает R_A = (17, 20) абоненту *B*;

2) абонент *B* выполняет следующие действия:

выбирает случайное целое число k_B = 2;

вычисляет точку $R_B = k_B \times G$ = 2(13, 7) = (5, 4);

вычисляет точку

$(x_2, y_2) = k_B \times Q_A$ = 2(17, 3) = (13, 16);

вычисляет

$s_B = k_B + (b \times x_B \times x_2) \bmod n$ =

= 2 + (6 × 13 × 13) mod 7 = 1;

пересылает R_B = (5, 4) абоненту *A*;

3) абонент *A* выполняет следующие действия:

вычисляет точку $(x_2, y_2) = a \times R_B$ = 3(5, 4) = (13, 16);

вычисляет общий секретный ключ

$K = s_A(R_B + (x_B \times x_2 \times Q_B))$ =

= 3((5, 4) + (13 × 13 × (13, 16))) = (17, 3);

4) абонент *B* выполняет следующие действия:

вычисляет $(x_1, y_1) = b \times R_A$ = 6(17, 20) = (17, 3);

вычисляет общий секретный ключ

$K = s_B(R_A + (x_A \times x_1 \times Q_A))$ =

= 1((17, 20) + (17 × 17 × (17, 3))) = (17, 3).

12.6. Схема гибридного шифрования (ECKAP)

В схеме ECKAP строится защищенный канал связи между абонентами на основе использования сеансового ключа. Процедуры определения параметров системы и генерации ключей аналогичны рассмотренным ранее.

Обозначения:

M – сообщение;

Q – открытый ключ абонента B;

d – секретный ключ абонента B;

K_S – сеансовый ключ;

C_M – шифротекст;

C_K – зашифрованный ключ;

E_S, D_S – соответственно за- и расшифрование с использованием симметричного криптоалгоритма;

E_A, D_A – соответственно за- и расшифрование с использованием асимметричного криптоалгоритма.

Зашифрование информации (абонент A защищает сообщение M, предназначенное для абонента B):

A вычисляет C_M, преобразуя сообщение M на сеансовом ключе K_S с использованием алгоритма E_S;

A вычисляет C_K, преобразуя на открытом ключе Q абонента B сеансовый ключ K_S с использованием алгоритма E_A;

сообщение для абонента B включает закрытый текст C_M и преобразованный ключ C_K.

Расшифрование информации (абонент B восстанавливает полученное сообщение, включающее закрытый текст C_M и преобразованный ключ C_K):

B восстанавливает сеансовый ключ K_S, преобразуя C_K на своем секретном ключе d с использованием алгоритма D_A;

B восстанавливает исходное сообщение M, преобразуя C_M на полученном сеансовом ключе K_S с использованием алгоритма D_S.

Пример

Параметры:

конечное поле $GF(23)$;

эллиптическая кривая E ($GF(23)$) вида $y^2 = x^3 + x + 1$;

базовая точка G = (13, 7);

порядок n = 7.

Генерация ключей:

B выбирает секретный ключ $d = 6$;

B вычисляет открытый ключ

$$Q = d \times G = 6(13, 7) = (13, 16).$$

Пусть открытый текст $M = 6$ и сеансовый ключ $K_S = (17, 3)$.

Зашифрование информации (абонент *A* защищает сообщение *M*, предназначенное для абонента *B*):

A вычисляет закрытый текст $C_M = M \oplus K_S = 6 \oplus 17 = 23$;

A вычисляет преобразованный ключ C_K:

A выбирает случайное целое число $k = 2$;

A вычисляет точку $(x_1, y_1) = k \times G = 2(13, 7) = (5, 4)$;

A вычисляет точку $(x_2, y_2) = k \times Q = 2(13, 16) = (5, 19)$;

A вычисляет $C_K = K_S \oplus x_2 = (17, 3) \oplus 5 = (20, 6)$;

закрытие данных:

$$C_M = 23,\ C_K = (20, 6),\ (x_1, y_1) = (5, 4).$$

Расшифрование информации (абонент *B* восстанавливает полученное от абонента *A* сообщение $C_M = 23$, используя ключ $C_K = (20, 6)$ и $(x_1, y_1) = (5, 4)$):

абонент *B* восстанавливает сеансовый ключ K_S:

вычисляет точку

$$(x_2, y_2) = d \times (x_1, y_1) = 6(5, 4) = (5, 19);$$

восстанавливает сеансовый ключ K_S

$$K_S = C_K \oplus x_2 = (20, 6) \oplus 5 = (17, 3);$$

абонент *B* восстанавливает исходное сообщение:

$$M = C_M \oplus K_S = 23 \oplus 17 = 6.$$

12.7. Российский стандарт на ЭЦП ГОСТ Р 34.10-2001

ГОСТ Р 34.10-2001 – российский стандарт, описывающий алгоритмы формирования и проверки электронной цифровой подписи. Введен в действие вместо ГОСТ Р 34.10-94. Старый стандарт описывал алгоритм ЭЦП на основе проблемы дискретного

логарифмирования в конечном поле, новый же предписывает использовать эллиптические кривые над полем $GF(p)$.

В отличие от западных аналогов в российском стандарте не регламентируются алгоритмы формирования ключей, параметров кривых и начальных точек, однако задаются определенные условия, которым должны удовлетворять используемые кривые.

Обозначения:

V_{256} – множество всех двоичных векторов длиной 256 бит;

V_∞ – множество всех двоичных векторов произвольной конечной длины;

Z – множество всех целых чисел;

p – простое число, $p > 3$;

$GF(p)$ – конечное простое поле из p элементов $\{0, 1, 2, ..., (p - 1)\}$;

$b \bmod p$ – минимальное неотрицательное число, сравнимое с b по модулю p;

h – двоичный вектор, например, двоичный вектор длиной 256 бит может быть представлен в виде

$$h = (\alpha_{255} \ldots \alpha_1 \alpha_0),\ \alpha_i \in \{0, 1\},$$

числу α соответствует вектор h, если выполняется равенство $\alpha = \Sigma\alpha_i 2^i$;

M – сообщение, $M \in V_\infty$;

(h_1, h_2) – конкатенация (объединение) двух двоичных векторов, пусть

$$h_1 = (\alpha_{255} \ldots \alpha_1 \alpha_0),\ \alpha_i \in \{0, 1\},$$

$$h_2 = (\beta_{255} \ldots \beta_1 \beta_0),\ \beta_i \in \{0, 1\},$$

тогда $(h_1, h_2) = (\alpha_{255} \ldots \alpha_1 \alpha_0 \beta_{255} \ldots \beta_1 \beta_0)$;

a, b – коэффициенты эллиптической кривой;

m – порядок группы точек эллиптической кривой;

q – порядок подгруппы группы точек эллиптической кривой;

O – бесконечно удаленная точка эллиптической кривой;

P – точка эллиптической кривой порядка q;

d – целое число, ключ формирования подписи;

Q – точка эллиптической кривой, ключ проверки подписи;

$sign\ (M)$ – цифровая подпись под сообщением M.

Механизм цифровой подписи включает в себя две основные процедуры:

1) формирование подписи (рис. 12.3);
2) проверку подписи (рис. 12.4).

Схема реализована с использованием операций группы точек эллиптической кривой, определенной над конечным простым полем, а также хеш-функции. Стойкость схемы цифровой подписи основывается на сложности решения задачи дискретного логарифмирования на группе точек эллиптической кривой, а также стойкости используемой хеш-функции. Алгоритм хеширования определен в ГОСТ 34.11.

Разрядность цифровой подписи устанавливается равной 512 бит.

Математические предпосылки

Инвариантом эллиптической кривой E называется величина $J(E)$, удовлетворяющая тождеству

$$J(E) \equiv 1728 \cdot (4a^3/(4a^3 + 27b^2)) \bmod p.$$

Коэффициенты a и b эллиптической кривой по известному инварианту J определяются следующим образом:

$$a \equiv 3k \bmod p,$$

$$b \equiv 2k \bmod p,$$

где

$$k \equiv J(E)/(1728 - J(E)) \bmod p,\ J(E) \neq 0,\ J(E) \neq 1728.$$

Для абелевой группы порядка m, которую образуют точки эллиптической кривой E, справедливо неравенство

$$p + 1 - 2\sqrt{p} \leq m \leq p + 1 + 2\sqrt{p}\,.$$

Точка Q называется точкой кратности k или просто кратной точкой эллиптической кривой E, если для некоторой точки P выполняется равенство $Q = kP$.

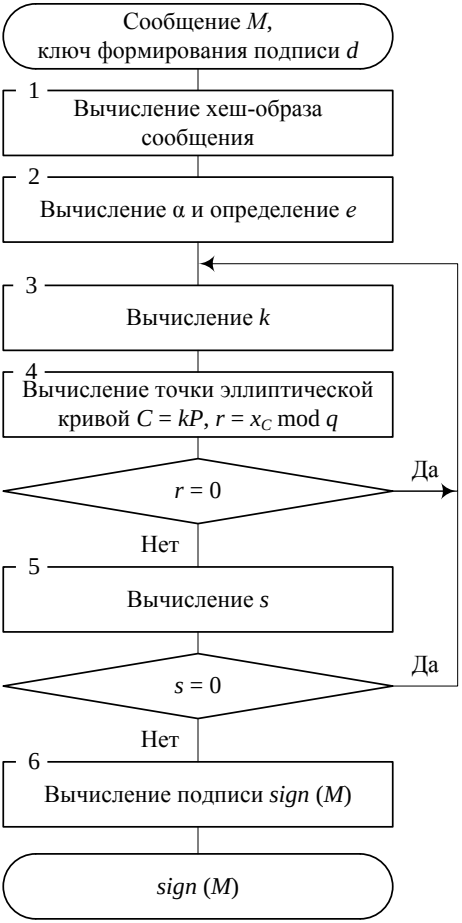


Рис. 12.3. ГОСТ Р 34.10-2001. Схема формирования цифровой подписи

Параметры схемы цифровой подписи:

простое число $p > 2^{255}$, при этом $p^i \neq 1 \bmod q$ для всех целых i, меньших 32;

эллиптическая кривая E, задаваемая своим инвариантом или коэффициентами a и b;

целое число m – порядок группы точек эллиптической кривой E ($m \neq p$);

простое число q – порядок циклической подгруппы группы точек кривой E, для которого выполняются условия

$m = nq$, $n \in Z$, $n \geq 1$,
$2^{254} < q < 2^{256}$;

точка $P \neq O$ кривой E с координатами (x_P, y_P), удовлетворяющая равенству $qP = O$.

Ключ формирования подписи – целое число d, удовлетворяющее неравенству $0 < d < q$.

Ключ проверки подписи – точка эллиптической кривой Q с координатами (x_Q, y_Q), удовлетворяющая равенству $dP = Q$.

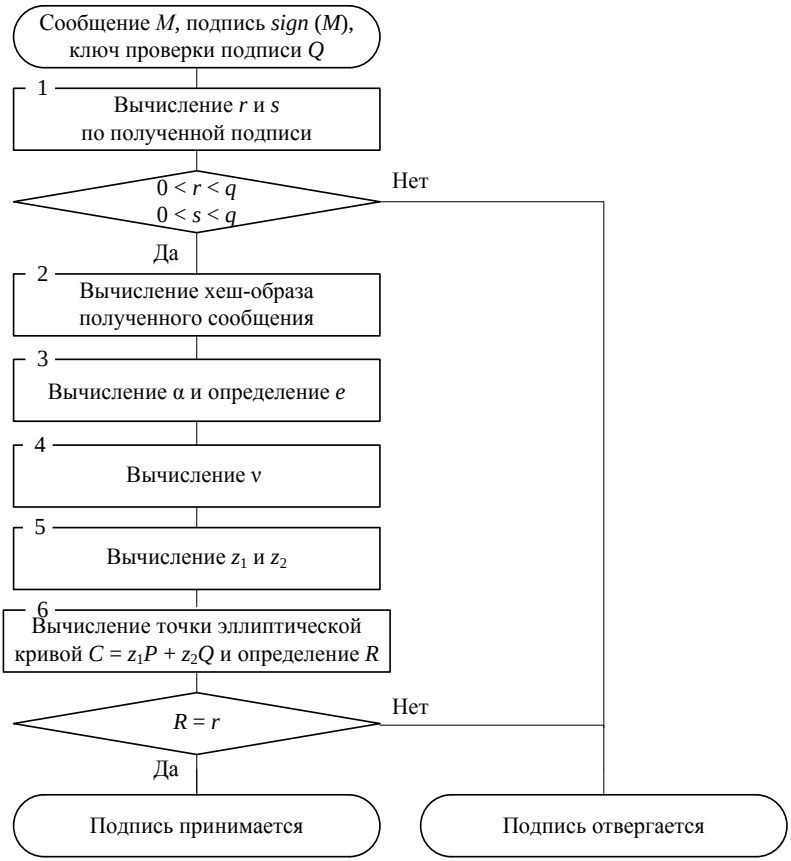


Рис. 12.4. ГОСТ Р 34.10-2001. Схема проверки цифровой подписи

Последовательность формирования цифровой подписи сообщения $M \in V_\infty$:

1) вычислить хеш-образ $h(M)$ сообщения M;
2) вычислить целое число α, двоичным представлением которого является вектор h и определить $e \equiv \alpha \bmod q$; если $e = 0$, то определить $e = 1$;
3) сформировать случайное число k, удовлетворяющее условию $0 < k < q$;
4) вычислить точку эллиптической кривой $C = kP$ и определить $r \equiv x_C \bmod q$, где x_C – координата точки C; если $r = 0$, то вернуться к шагу 3;
5) вычислить значение $s \equiv (rd + ke) \bmod q$; если $s = 0$, то вернуться к шагу 3;
6) вычислить двоичные векторы r и s и определить цифровую подпись $sign\ (M)$ как конкатенацию двух двоичных векторов (r, s).

Последовательность проверки цифровой подписи:

1) по полученной подписи $sign\ (M)$ вычислить целые числа r и s; если выполнены неравенства
$$0 < r < q,\ 0 < s < q,$$
то перейти к следующему шагу, в противном случае подпись отвергается;
2) вычислить хеш-образ $h(M)$ полученного сообщения M;
3) вычислить целое число α, двоичным представлением которого является вектор h, и определить $e \equiv \alpha \bmod q$; если $e = 0$, то определить $e = 1$;
4) вычислить значение $\nu \equiv e^{-1} \bmod q$;
5) вычислить значения $z_1 \equiv s\nu \bmod q$, $z_2 \equiv -r\nu \bmod q$;
6) вычислить точку эллиптической кривой $C = z_1P + z_2Q$ и определить $R \equiv x_C \bmod q$, где x_C – координата точки C;
7) если выполняется равенство $R = r$, то подпись принимается, в противном случае – отвергается.

Контрольные вопросы

1. Приведите пример построения криптосистемы с открытым ключом, основанной на свойствах эллиптических кривых.
2. Приведите примеры эллиптических алгоритмов формирования и проверки ЭЦП.
3. Приведите пример построения гибридной криптосистемы на основе ECCS.

ГЛАВА 13. АТАКИ НА ECDLP ОБЩЕГО ВИДА

Необходимое условие безопасности всех стохастических алгоритмов ОБИ, основанных на свойствах эллиптических кривых, – сложность решения ECDLP. Рассмотрим алгоритмы решения ECDLP и методы, позволяющие избежать появления предпосылок для проведения атак на практике [46].

При построении схемы преобразования сначала выбираются несекретные параметры E, $GF(q)$, точка P порядка N на E. Затем случайным образом выбирается $d \in [1, N-1]$ и вычисляется $Q = kP$, где $k < N$. Открытым ключом объявляется Q, а секретным – k. Очевидно, что если решение ECDLP является простым, противник может получить k из Q. Таким образом, сложность решения ECDLP является критической для безопасности всех схем, основанных на свойствах эллиптических кривых.

Пусть E – эллиптическая кривая над конечным полем $GF(q)$ и $P \in E(GF(q))$ – точка порядка N. Заданы параметры эллиптической кривой E, $GF(q)$, P, N, а также точка $Q \in E(GF(q))$. Задача состоит в нахождении уникального целого $k \in [1, N-1]$ такого, что $Q = kP$.

13.1. Полный перебор (Exhaustive Search)

При полном переборе последовательно вычисляются значения P, $2P$, $3P$, $4P$, … до получения Q. Если рассматривать сложение, как операцию на эллиптической кривой, метод в наихудшем случае требует $O(N)$ операций на построение таблицы и $O(N)$ памяти для ее хранения. Чтобы противодействовать этой атаке, следует выбирать параметры кривой так, чтобы N было достаточно велико.

13.2. Атака Полига–Хеллмана (Pohlig–Hellman Attack)

Атака Полига–Хеллмана использует разложение на простые множители числа N – порядка точки P. В результате проблема получения k сводится к проблеме получения k по модулю каждого из простых сомножителей N; после чего требуемое число k

можно получить путем применения китайской теоремы об остатках.

Пусть P и Q – элементы группы G, требуется найти целое число k такое, что $Q = kP$. Допустим, известно разложение порядка N точки P на простые сомножители: $N = \prod_i q_i^{e_i}$. Идея атаки Полига–Хеллмана состоит в том, чтобы найти k по модулю $q_i^{e_i}$ для каждого i, после чего, используя китайскую теорему об остатках, получить $k \bmod N$.

Пусть q – простое, q^e – степень q, делящая N. Запишем k в следующем виде:

$$k = k_0 + k_1 q + k_2 q^2 + \cdots,\ 0 \le k_i < q.$$

Будем вычислять $k \bmod q^e$, последовательно определяя $k_0, k_1, \ldots, k_{e-1}$. Алгоритм Полига–Хеллмана имеет вид:

1) вычисляем $T = \left\{ j\left(\frac{N}{q}\right)Q \,\middle|\, 0 \le j \le q-1 \right\}$;
2) вычисляем $\frac{N}{q}Q$; Это будет элемент $k_0\left(\frac{N}{q}P\right)$ из T;
3) если $e = 1$, останов; в противном случае продолжаем;
4) пусть $Q_1 = Q - k_0 P$;
5) вычисляем $\frac{N}{q_2}Q_1$; это будет элемент $k_1\left(\frac{N}{q}P\right)$ из T;
6) полагаем, что вычислили $k_0, k_1, \ldots, k_{r-1}$ и $Q_1, \ldots, Q_{r-1}$;
7) пусть $Q_r = Q_{r-1} - k_{r-1} q^{r-1} P$;
8) определяем k_r такое, что $\frac{N}{q^{r+1}}Q_r = k_r\left(\frac{N}{q}P\right)$;
9) если $r = e - 1$, останов; в противном случае возвращаемся к шагу 7;
10) получаем k, используя китайскую теорему об остатках.

Пример. Пусть $G = E(GF(599))$, где кривая E задана уравнением $y^2 = x^3 + 1$, $P = (60, 19)$ и $Q = (277, 239)$. P имеет порядок $N = 600$. Хотим решить $Q = kP$ относительно k. Разложение на простые множители $N = 600$ имеет вид $600 = 2^3 \cdot 3 \cdot 5^2$.

Будем вычислять k mod 8, k mod 3, и k mod 25, затем получим k mod 600 при помощи китайской теоремы об остатках.

Найдем k mod 8. Вычисляем $T = \{O, (598, 0)\}$. Так как

$$(N/2)Q = O = 0\cdot(N/2)P,$$

имеем $k_0 = 0$. Поэтому $Q_1 = Q - 0P = Q$. Так как

$$(N/4)Q_1 = 150Q_1 = (598, 0) = 1\cdot(N/2)P,$$

имеем $k_1 = 1$. Поэтому $Q_2 = Q_1 - 1\cdot 2\cdot P = (35, 243)$. Так как

$$(N/8)Q_2 = 75Q_2 = O = 0\cdot(N/2)P,$$

имеем $k_2 = 0$. Поэтому $k = 0 + 1\cdot 2 + 0\cdot 4 \equiv 2 \bmod 8$.

Найдем k mod 3. $T = \{O, (0, 1), (0, 598)\}$. Так как

$$(N/3)Q = (0, 598) = 2\cdot(N/3)P,$$

имеем $k_0 = 2$. Поэтому $k \equiv 2 \bmod 3$.

Найдем k mod 25. $T = \{O, (84, 179), (491, 134), (491, 465), (84, 420)\}$. Так как $(N/5)Q = (84, 179)$, имеем $k_0 = 1$. Тогда

$$Q_1 = Q - 1\cdot P = (130, 129).$$

Так как $(N/25)Q_1 = (491, 465)$, имеем $k_1 = 3$. Поэтому

$$k = 1 + 3\cdot 5 \equiv 16 \bmod 25.$$

Таким образом, справедливо:

$$\begin{cases} x \equiv 2 \pmod 8, & (13.1) \\ x \equiv 2 \pmod 3, & (13.2) \\ x \equiv 16 \pmod{25}. & (13.3) \end{cases}$$

Запишем сравнение (13.1) так:

$$x = 8q_1 + 2. \quad (13.4)$$

Запишем сравнение (13.2) следующим образом:

$$2 = 8q_1 + 2 \bmod 3,$$
$$8q_1 = 0 \bmod 3,$$
$$q_1 = 0 \bmod 3, \text{ или } q_1 = 3q_2 \bmod 3.$$

Запишем сравнение (13.4) так:

$$x = 8(3q_2) + 2. \quad (13.5)$$

Запишем сравнение (13.3) следующим образом:

$$16 = 24q_2 + 2 \bmod 25,$$
$$12q_2 = 7 \bmod 25,$$
$$q_2 = 11.$$

Запишем сравнение (13.5) так:

$$x = 8(3\cdot 11) + 2 = 266.$$

В результате получаем $k \equiv 266 \bmod 600$, т.е. $k = 266$.

Метод Полига–Хеллмана хорошо работает, если все простые делители N малы. Однако если q – большое простое число, делящее N, то трудно составить список T, который содержит q элементов. Чтобы воспрепятствовать этой атаке и построить наиболее трудный экземпляр ECDLP, следует выбирать эллиптическую кривую, порядок которой кратен большому простому числу N (более 30 цифр). Желательно чтобы данный порядок был простым или почти простым (большим простым N, умноженным на небольшое целое h (2 или 4)).

13.3. Алгоритм маленьких и больших шагов Шэнкса (Baby-step Giant-step)

Этот метод, требующий выполнения до $O\left(N^{1/2}\log N\right)$ операций и затрат памяти до $O\left(N^{1/2}\right)$, хорошо работает только для N небольшого размера.

Алгоритм «Baby-step Giant-step» имеет следующий вид:

1) фиксируем целое число $m \geq \sqrt{N}$ и вычисляем mP;
2) вычисляем iP и сохраняем список iP, $0 < i \leq m$;
3) вычисляем точки $Q - jmP$ для $j = 0, 1, \ldots, m - 1$, пока не получим точку, соответствующую элементу из сохраненного списка;
4) если $iP = Q - jmP$, имеем $Q = kP$, где $k \equiv i + jm \pmod N$.

Точка iP вычисляется путем прибавления P («baby step») к $(i - 1)P$. Точка $Q - jmP$ вычисляется путем прибавления $-mP$ («giant step») к $Q - (j - 1)mP$.

Заметим, что нет необходимости знать точный порядок N группы G. Требуется только верхняя граница значений N. Поэтому для эллиптических кривых над $GF(q)$ можно использовать теорему Хассе ($m^2 \geq q+1+2\sqrt{N}$) для нахождения верхней границы значений N.

Пример. Пусть $G = E(GF(23))$, где кривая E задана уравнением $y^2 = x^3 + x + 1$. Пусть $P = (0, 1)$ и $Q = (11, 20)$. Согласно теореме Хассе знаем, что порядок G – самое большее 32, поэтому пусть $m = 6$.

Точки iP для $0 < i \leq 5$ имеют вид

(0, 1), (6, 19), (3, 13), (13, 16), (18, 3).

Вычисляем $Q - jmP$ для $j = 0, 1, 2, 3$ и получаем

(11, 20), (9, 7), (19, 18), (3, 13).

Останавливаемся, так как четвертая точка соответствует $3P$.

Поскольку $j = 3$, имеем

$$(11, 20) = (5+3\cdot 6)P = 21P.$$

Поэтому $k = 21$.

Недостатком метода «Baby-step Giant-step» является то, что для его реализации требуются значительные затраты памяти. ρ- и λ-методы Полларда выполняются приблизительно за то же самое время, что и метод «Baby-step Giant-step», но требуют в отличии от него небольших затрат памяти.

13.4. ρ-метод Полларда

ρ-алгоритм Полларда на данный момент считается лучшим из известных алгоритмов для вычисления дискретного логарифма на эллиптической кривой и является рандомизированной версией алгоритма «baby-step giant-step». Сложность алгоритма оценивается как $O\left(N^{1/2}\right)$, где N порядок генератора.

Пусть Q – точка с неизвестным индексом и P – генератор группы точек на эллиптической кривой порядка N. Полагаем, что $Q = xP$. Требуется найти x.

ρ-алгоритм Полларда имеет следующий вид:

1) разбиваем группу на три множества S_0, S_1 и S_2 приблизительно одинаковой мощности;
2) случайным образом выбираем $1 \le a_0, b_0 \le N - 1$;
3) устанавливаем $X_0 = a_0P + b_0Q$ – начальную точку для случайного блуждания;
4) определяем последовательность $\{X_i\}$ следующим образом:

$$X_i = \begin{cases} P + X_{i-1}, \text{ если } X_{i-1} \in S_0; \\ 2X_{i-1}, \text{ если } X_{i-1} \in S_1; \\ Q + X_{i-1}, \text{ если } X_{i-1} \in S_2; \end{cases}$$

это дополнительно определяет две последовательности $\{a_i\}$, $\{b_i\}$:

$$a_i = \begin{cases} 1 + a_{i-1}, \text{ если } X_{i-1} \in S_0; \\ 2a_{i-1}, \text{ если } X_{i-1} \in S_1; \\ a_{i-1}, \text{ если } X_{i-1} \in S_2. \end{cases}$$

$$b_i = \begin{cases} b_{i-1}, \text{ если } X_{i-1} \in S_0; \\ 2b_{i-1}, \text{ если } X_{i-1} \in S_1; \\ 1 + b_{i-1}, \text{ если } X_{i-1} \in S_2; \end{cases}$$

5) допустим, что для некоторого $i \neq j$ справедливо $X_i = X_j$, тогда получаем

$a_iP + b_iQ = a_jP + b_j Q$;
$(a_i - a_j)P = (b_j - b_i)Q$;
$Q = (a_i - a_j)(b_j - b_i)^{-1}P$;

6) вычисляем $k \equiv (a_i - a_j)(b_j - b_i)^{-1} (\mathrm{mod}\, N)$, если $(b_j - b_i)$ имеет обратный элемент по модулю N, то решение найдено.

Форма процесса вычисления точек похожа на греческую букву ρ (рис. 13.1). Наиболее эффективный метод для нахождения $X_i = X_j$ поиск $X_i = X_{2i}$. Поиск в этом случае требует сохранять только шесть значений: X_i, a_i, b_i, X_{2i}, a_{2i}, b_{2i}. Такой алгоритм может не обнаружить первую пару $X_i = X_j$, но зато исключает необходимость хранения большого списка точек и поиска их на каждой итерации.

Пример. Пусть $G = E(GF(41))$, где E – кривая, соответствующая уравнению $y^2 = x^3 + 2x + 1$. Пусть $s = 3$. Пусть $P = (0, 1)$ и $Q = (13, 25)$. P имеет порядок 39. Необходимо найти такое k, что $kP = Q$.

Стартуем со случайного элемента X_0 и вычисляем итерации $X_{i+1} = f(X_i)$, где $f(X_i)$ – функция для случайного блуждания.

Вычисляем $X_0 = 3P + 5Q$, $X_0 = (40, 30)$.

Последовательность $\{X_i\}$ определяется следующим образом:

$$X_i = f(X_{i-1}) = \begin{cases} P + X_{i-1}, \text{ если } x \equiv 0; \\ 2X_{i-1}, \text{ если } x \equiv 1; \\ Q + X_{i-1}, \text{ если } x \equiv 2. \end{cases}$$

Здесь x – целое число, $0 \le x < 41$, взятое по модулю 3. Например,

$$f(X_0) = P + X_0 = (4, 14),$$

потому что $X_0 = (40, 30)$ и $i \equiv 1 \bmod 3$.

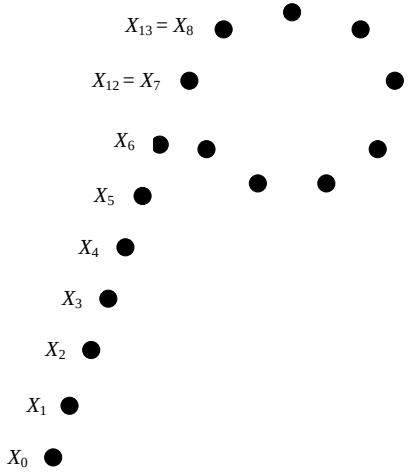


Рис. 13.1. Схема процесса вычисления точек

При вычислении X_0, $X_1 = f(X_0)$, $X_2 = f(X_1)$, … получаем

$X_0 = (40, 30)$, $X_1 = (4, 14)$, $X_2 = (0, 40)$,
$X_3 = O$, $X_4 = (0, 1)$, $X_5 = (1, 39)$,
$X_6 = (38, 38)$, $X_7 = (11, 40)$, $X_8 = (22, 22)$,
$X_9 = (28, 22)$, $X_{10} = (8, 23)$, $X_{11} = (30, 1)$,
$X_{12} = (11, 40)$, $X_{13} = (22, 22)$, … .

Таким образом, последовательность начинает повторяться при $X_7 = X_{12}$. Поэтому получаем

$$30P + 41Q = 121P + 169Q;$$
$$(30 - 121)P = (169 - 41)Q;$$
$$Q = -91 \times 128^{-1} \times P.$$

Так как P имеет порядок 39, вычисляем

$$k = -91 \times 128^{-1} \equiv 13 \bmod 39.$$

В результате $Q = 13P$, а $k = 13$.

Предположим, что распараллеленная версия ρ-алгоритма Полларда останется лучшим алгоритмом для решения ECDLP. Существует экспертная оценка, что если параметр N удовлетворяет условию $N > 2^{160}$, то аппаратные атаки становятся неосуществимы, даже принимая во внимание непрерывное совершенствование аппаратных средств, в том числе снижение их

стоимости. В [46] дается оценка, что 163-битная и 191-битная кривая обеспечат соответственно в 2021 и 2040 гг. тот же уровень защиты, который обеспечивал в 1982 г. стандарт DES (который считался в то время безопасным для банковских приложений).

13.5. λ-метод Полларда

λ-метод Полларда, как и ρ-метод, использует функцию f случайного блуждания, но при этом берутся несколько случайных начальных точек. При двух случайных начальных точках имеются две случайные траектории блуждания. Как только эти две последовательности пересеклись, дальнейшее движение осуществляется по одному пути. Форма процесса вычисления точек похожа на греческую букву λ (рис. 13.2).

Этот метод в наихудшем случае требует самое большее $\sqrt{N}$ шагов.

Пусть f – функция, X_0 – случайная точка в G и $\{X_i\}$, $i \in N$, – последовательность случайного блуждания. Для некоторой положительной константы t вычисляем $X_t = f^t(X_0)$. t должно быть выбрано достаточно большим, чтобы имелась некоторая вероятность того, что X_t находится на уникальном цикле. Пусть Y_0 – другая случайная точка в G, а $\{Y_j\}$, $j \in N$, – другая последовательность случайного блуждания, где $Y_j = f(Y_{j-1}) = f^j(Y_0)$ для всех j. На j-м шаге вычисляем Y_j и проверяем, равен ли он X_t. Если $Y_j = X_t$, останов. Иначе, продолжаем.

Чтобы предотвратить эту атаку, секретный ключ в системах, основанных на свойствах эллиптических кривых, должен быть равномерно распределен на интервале $[1, N-1]$.

13.6. Дискретное логарифмирование методом индексного исчисления

Метод индексного исчисления, имеющий субэкспоненциальную сложность, является наиболее эффективным алгоритмом вычисления дискретного логарифма в группах G, удовлетворяющих следующим условиям:

$$\log_g(a \times b) = \log_g a + \log_g b, \quad (13.6)$$

$$\log_g(a^e) = e \log_g a. \quad (13.7)$$

В частности, он применим для дискретного логарифмирования в полях $GF(p)$ и $GF(2^m)$.

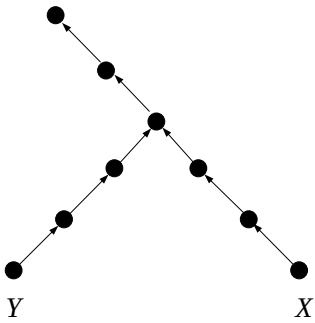


Рис. 13.2. Схема процесса вычисления точек

Данный алгоритм требует предварительного определения так называемой *факторной базы*, т.е. относительно небольшой совокупности S элементов группы, таких, что значительная часть элементов может быть представлена как их произведение.

В качестве примера рассмотрим метод индексного исчисления применительно к полю $GF(p)$. В этом случае факторную базу обычно задают как число –1 и последовательность простых чисел, начиная с 2. С точки зрения эффективности алгоритма выгодно выбрать небольшую факторную базу, однако, для нахождения дискретного логарифма в большой группе может потребоваться большая факторная база. Таким образом, при реализации этого метода приходится идти на компромисс.

Итак, задача алгоритма – найти такое число $x \in GF(p)$, что $g^x = h$, при условии, что известны g, h и p, где g – генератор элементов группы G, а h на нее элемент этой группы. На входе алгоритма имеется факторная база, состоящая из $r + 1$ элементов $\{-1, 2, 3, 5, \ldots, q\}$, g, h, p, где q – последнее простое число из используемой факторной базы.

Метод предполагает выполнение двух этапов. Первый включает в себя сбор информации о соотношениях между факторной базой и степенями генератора поля g. Фактически, это стадия

предварительных вычислений. Возможный алгоритм первого этапа выглядит следующим образом:

1) $k := 0,\ l := 0.$
2) $k := k + 1;$
3) вычисляем g^k;
4) раскладываем g^k на множители по факторной базе, т.е. находим такую последовательность $k_0, k_1, \ldots, k_r$, что
$$g^k = (-1)^{k_0} 2^{k_1} 3^{k_2} 5^{k_3} \cdot \ldots \cdot q^{k_r};$$
5) если факторизация невозможна, возвращаемся к шагу 2;
6) запоминаем полученное соотношение в виде вектора
$$v_l = (k_0, k_1, k_2, k_3, \ldots, k_r, k);$$
7) если v_l не является линейно независимым относительно уже сохраненных соотношений, возвращаемся к шагу 2;
8) $l := l + 1$; если $l < r$, переходим к шагу 2, в противном случае работа алгоритма завершается, так как найдено достаточное количество соотношений.

Иначе говоря, мы получили совокупность линейно независимых разложений первых $r + 1$ элементов группы на множители по факторной базе.

Теперь появляется возможность построить из этих соотношений систему линейных уравнений и после ее решения относительно k получить дискретный логарифм каждого элемента факторной базы.

Второй этап включает в себя разложение h на множители по заданной факторной базе. После этого можно легко вычислить x по соотношениям (13.6) и (13.7).

Контрольные вопросы

1) Приведите пример атаки на ECDLP: а) Полига–Хеллмана; б) с использованием алгоритма маленьких и больших шагов Шэнкса; в) с использованием ρ-метода Полларда; г) с использованием λ-метода Полларда; д) методом индексного исчисления.

Заключение

Авторы попытались продемонстрировать безграничные возможности стохастических методов ОБИ, в первую очередь криптографических, с помощью которых можно решить практически любую задачу защиты информации. В настоящее время стохастические методы защиты используются в информационных системах любой степени сложности и назначения. Этими методами защищается государственная тайна, обеспечивается законность электронного документооборота, предотвращаются попытки мошенничества в системах электронной торговли. Без криптографии обеспечить требуемую степень безопасности в современном компьютеризированном мире уже не представляется возможным. Со временем ее роль и значение обещают стать еще больше.

Следует акцентировать внимание на часто игнорируемом факте, суть которого в том, что стохастические методы – технологии двойного назначения, которые могут использоваться (и активно используются!) не только для защиты, но и нападения, в частности для создания РПВ. Первым следствием этого факта является то, что любое решение, связанное с защитой информации, необходимо анализировать как с позиции разработчика, так и злоумышленника, для чего нужны знания основных видов атак на программные системы. В качестве второго следствия можно отметить: в настоящее время не уделяется достаточно внимания технологиям комплексного анализа защищенности программных систем и совершенствованию методов и средств защиты. Разработчики систем ОБИ часто не учитывают, что эффективная система защиты – это не фиксированная совокупность методов и средств, а непрерывный процесс периодического анализа защищенности системы на всех ее уровнях и опережающего совершенствования методов и средств защиты.

Рассмотрим еще раз основные проблемы, с которыми приходится сталкиваться разработчикам систем ОБИ [26, 30, 49].

Современная наука предоставляет все необходимые алгоритмы, методы и средства, позволяющие построить систему защиты, затраты на взлом которой таковы, что у противника с

ограниченными финансовыми и техническими возможностями для получения интересующей его информации остаются только две возможности – использование либо человеческого фактора, либо особенностей конкретной реализации криптоалгоритмов и криптопротоколов, которая чаще всего оставляет желать лучшего. Именно такой вывод можно сделать, анализируя примеры реальных успешных атак на криптосистемы. Различных примеров взломов реальных систем очень много, в то же время известны лишь единичные случаи взлома с использованием исключительно математических методов.

Система защиты в целом не может быть надежнее отдельных ее компонентов. Иными словами, для того чтобы преодолеть систему защиты, достаточно взломать или использовать для взлома самый ненадежный из ее компонентов. Самое ненадежное звено системы – человек. Типичные ошибки пользователей, нарушающие безопасность всей системы защиты:

предоставление своего секретного пароля коллегам по работе для решения неотложных задач во время отсутствия владельца пароля;

повторное использование секретных паролей в несекретных системах;

генерация паролей самими пользователями, выбор паролей по критерию удобства запоминания;

несвоевременное информирование о компрометации ключевой информации (например, об утере смарт-карт).

Получают распространение атаки типа отказ в обслуживании (denial of service), провоцирующие пользователя отключать «заедающую» систему защиты при решении неотложных задач.

Можно выделить следующие причины ненадежности криптосистем, связанные с особенностями их реализации:

использование нестойких криптоалгоритмов;

неправильное применение криптоалгоритмов;

ошибки в реализации криптоалгоритмов.

В некоторых случаях, особенно в системах реального времени, применение стойких алгоритмов принципиально невозможно в силу их низкого быстродействия, и поэтому вынужденно используются менее стойкие, но быстрые криптоалгоритмы.

Многие качественные криптографические средства подпадают под действие экспортных ограничений, искусственно снижающих качество этих средств. Например, в США запрещен экспорт криптоалгоритмов с длиной ключа более 56 бит. Все программные средства, произведенные в США и легально экспортируемые за рубеж, обеспечивают ослабленную криптографическую защиту. Аналогичная ситуация имеет место и в Европе. Так, например, существуют две версии алгоритма поточного шифрования A5: надежная A5/1 и существенно менее стойкая A5/2 для поставок в развивающиеся страны.

Многие разработчики ПО включает в свои продукты собственные криптоалгоритмы, самонадеянно считая себя специалистами, забывая, что современная криптография основана на глубоких исследованиях в таких разделах математики, как высшая алгебра, теория чисел, теория информации, теория сложности вычислений и др. Если разработчики делают ставку на собственные методы, шансы взломщиков, даже в случае полного отсутствия на начальном этапе информации об использованном алгоритме, многократно возрастают.

Основные ошибки при применении криптоалгоритмов:

- недостаточная длина ключа или недостаточный объем ключевого пространства;
- некачественная процедуры генерации, хранения или распределения ключей;
- некачественный генератор псевдослучайных чисел или неправильная его инициализации;
- применение криптоалгоритмов не по назначению;
- использование на практике модели доверительных отношений, отличной от той, в расчете на которую проектировалась система.

Тема ошибок в реализации криптоалгоритмов в силу своей нетривиальности и многообразия требует отдельного рассмотрения, поэтому ограничимся лишь кратким перечислением основных проблем, возникающих при реализации криптоалгоритмов.

Надежная система защиты должна быть способна оперативно обнаруживать несанкционированные действия для минимизации возможного ущерба. В случае обнаружения повреждений

в системе должны включаться эффективные процедуры восстановления разрушенных элементов. Система не должна потерять живучесть даже в случае проведения на нее успешной атаки.

Причины наличия большинства «дыр» (или люков) в ПО, т.е. не описанных в документации возможностей работы с ним, очевидны: забывчивость разработчиков, которые в процессе отладки продукта создают временные механизмы, облегчающие ее проведение (например, за счет прямого доступа к отлаживаемым частям программы). По окончании отладки часть «дыр» убирается, а о части разработчики благополучно забывают либо оставляют их сознательно, особенно в ранних версиях продукта, когда в будущем весьма вероятна его доработка.

«Дыры» могут являться следствием применения технологии разработки программ «сверху вниз», когда программист сразу приступает к написанию управляющей программы, заменяя предполагаемые в будущем подпрограммы «заглушками», имитирующими реальные подпрограммы или просто обозначающими место их будущего подсоединения. Очень часто эти «заглушки» остаются в конечной версии программы. Либо опять же по причине забывчивости, либо в расчете на будущую модификацию продукта, либо, например, если в процессе разработки выясняется, что какая-то подпрограмма не нужна, а удалить «заглушку» не представляется возможным. В случае обнаружения такой «заглушки», злоумышленник может воспользоваться ей для подключению к программе своей подпрограммы, работающей отнюдь не в интересах законного пользователя.

Третий источник «дыр» – неправильная обработка (или ее отсутствие) каких-либо нестандартных ситуаций, которые могут иметь место при работе программы: неопределенный ввод, ошибки пользователей, сбои и т.п. В этом случае противник может искусственно вызвать в системе появление такой нестандартной ситуации, чтобы выполнить нужные ему действия. Например, он может вызвать аварийное завершение программы, работающей в привилегированном режиме, чтобы, перехватив управление, остаться в этом привилегированном режиме.

Наконец, известны случаи, когда люк в ПО или аппаратуре – первый шаг к атаке системы безопасности. Разработчик умыш-

ленно оставляет его в конечном продукте, чтобы в будущем, например, иметь возможность модифицировать информацию незаметно для законного пользователя, расшифровывать ее, не зная ключа, и т.п.

Существуют программы, изначально предназначенные для разрушительных действий. Они получили обобщенное название разрушающих программных воздействий (РПВ). Пути внедрения РПВ чрезвычайно разнообразны. Можно выделить следующие средства, предназначенные для борьбы с РПВ и без которых любая программная реализация криптоалгоритма практически беззащитна:

средства, препятствующие внедрению РПВ;

средства выявления РПВ до использования программных продуктов по назначению;

средства, обеспечивающие оперативное обнаружение РПВ в процессе реального функционирования ПО, изначально свободного от них;

средства удаления РПВ;

средства определения факта наличия или отсутствия РПВ в добавляемом в систему ПО.

Аппаратуру легче физически защитить от проникновения извне. Криптомодули могут помещаться в особые контейнеры, которые делают невозможным изменение алгоритма функционирования. Интегральные схемы могут покрываться специальным химическим составом, при этом любая попытка преодоления защитного слоя приводит к самоуничтожению их внутренней логической структуры. Тем не менее известны случаи выявления и аппаратных закладок.

Кроме того, возникает проблема защиты от экзотических атак, применимых к реализациям в смарт-картах, – временного анализа и анализа потребляемой мощности. Эти атаки основаны на использовании того факта, что различные операции, выполняемые на микропроцессоре, требуют разного времени, а также приводят к разному потреблению мощности. Общая идея этих атак в том, что, анализируя временные характеристики алгоритма (время ответа) или потребление мощности, мы можем составить картину выполнения различных операций и даже приблизительно вычислить их аргументы. Приблизительный

анализ уязвимости различных операций с точки зрения временных характеристик дает следующие результаты:

поиск по таблицам – неуязвим для временных атак;

фиксированные сдвиги – неуязвимы для временных атак;

булевы операции – неуязвимы для временных атак;

сложение/вычитание – трудно защитить от временных атак;

умножение/деление – наиболее уязвимые для временных атак операции.

Стойкость к такого рода атакам, направленным не на криптоалгоритм, а на его реализацию, также надо учитывать. Защищенность по отношению к временному анализу можно повысить путем введения дополнительных задержек. Более сложна проблема защиты от анализа мощности, но ее можно решить несколькими путями. Это, во-первых, балансировка алгоритма (равномерное распределение различных операций по коду), во-вторых – введение специальных «шумовых» операций, или, наконец, просто выбор другого микропроцессора.

С недавних пор получили распространение атаки на аппаратуру криптосистем, основанные на анализе электромагнитного излучения и других побочных источников информации.

Получают распространение, по сути, «биологические» методы взлома, рассматривающие криптосистемы как сложные объекты, определенным образом реагирующие на внешние раздражители. Атаки подобного рода основаны на анализе поведения системы после случайных или преднамеренных сбоев в работе и последующим использованием выявленных некачественных процедур восстановления после сбоев.

Универсальным путем противодействия атакам на криптосистемы с использованием побочных каналов (side channel attacks) является применение запутывающих преобразований и рандомизации при реализации криптоалгоритмов.

Как справедливо отмечается в [30], сама по себе криптография довольно бесполезна. Она является всего лишь частью более крупной системы безопасности. Очень часто бывает так, что место криптографии в системе безопасности можно сравнить с толстой банковской дверью из закаленной стали в туристической палатке. Разработчики любят обсуждать длину клю-

ча в криптосистемах, а вот устранять переполнение буфера или состязания при доступе к ресурсу в программных системах им нравится куда меньше. Криптография полезна тогда, когда оставшаяся часть системы также безопасна. Тем не менее правильная реализация криптографии важна и в системах, имеющих слабые стороны. Потому что атаки на криптосистемы проходят быстро и незаметно, позволяя взломщику возвращаться вновь и вновь.

Криптография – это не панацея. Во многих случаях она обеспечивает не реальную безопасность, а только видимость безопасности – польза такая же, как от амулета, который носят на шее. Она может стать как частью решения, так и частью самой проблемы, ослабляя безопасность. Например, есть секретный файл – можно защитить файловую систему от неавторизованного доступа, а можно дополнительно зашифровать файл и защитить ключ. Некоторые программы сохраняют ключи на диске. В результате вместо одной «дыры» появляются две: возможны «старые» атаки только уже на ключевую информацию; возможны атаки на алгоритм шифрования [30].

Сложная система ОБИ может создать ложное впечатление полной безопасности. Излишнее усердие при разработке системы защиты может привести к новым проблемам, поскольку более изощренная система ОБИ повышает сложность системы в целом. Как следствие, более сложная система затрудняет анализ стойкости. Кроме того, в ней потенциально возможно большее количество ошибок реализации [30].

Итак, несмотря на успехи современной криптографии, задача построения надежной системы защиты комплексная, она значительно сложнее, чем кажется на первый взгляд. Надежная система защиты может быть построена только с учетом всех перечисленных факторов.

Список использованных источников информации

1. Аршинов М.Н., Садовский Л.Е. Коды и математика (рассказы о кодировании). М.: Наука, 1983.
2. Блочные криптосистемы. Основные свойства и методы анализа стойкости / А.А. Варфоломеев, А.Е. Жуков, А.Б. Мельников, Д.Д. Устюжанин. М.: МИФИ, 1998.
3. Брассар Ж. Современная криптология: Пер. с англ. М.: ПОЛИМЕД, 1999.
4. Введение в криптографию / Под общ. ред. В.В. Ященко. М.: МЦНМО, «ЧеРо», 1998.
5. Винокуров А.Ю. ГОСТ не прост ... , а очень прост! // Монитор, 1995, № 1, с. 60–73.
6. Винокуров А.Ю. Алгоритм шифрования ГОСТ 28147-89, его использование и реализация для компьютеров платформы Intel x86. 1997. http://www.ssl.stu.neva.ru/psw/crypto.html.
7. Винокуров А.Ю. Как устроен блочный шифр? 1995. http://www.ssl.stu.neva.ru/psw/crypto.html.
8. Винокуров А.Ю. Проблема аутетификации данных и блочные шифры. 1998. http://www.ssl.stu.neva.ru/psw/crypto.html.
9. Галатенко В.А. Основы информационной безопасности: учебное пособие. Под ред. В.Б. Бетелина. – 4-е изд. М.: Интернет-университет информационных технологий; БИНОМ. Лаборатория знаний, 2008.
10. Гарднер М. От мозаик Пенроуза к надежным шифрам: Пер. с англ. М.: Мир, 1993.
11. Герасименко В.А., Малюк А.А. Основы защиты информации. М.: МИФИ, 1997.
12. ГОСТ 28147-89. Система обработки информации. Защита криптографическая. Алгоритм криптографического преобразования.
13. ГОСТ Р 34.10 – 2001. Криптографическая защита информации. Процессы формирования и проверки электронной цифровой подписи. М.: Госстандарт России, 2001.

14. Зензин О.С., Иванов М.А. Стандарт криптографической защиты AES. Конечные поля / Под ред. М.А. Иванова. Серия СКБ (специалисту по компьютерной безопасности). Кн. 1. М.: Кудиц-Образ, 2002.
15. Зиммерман Ф.Р. PGP: концепция безопасности и уязвимые места: Пер. с англ. // Компьютерра, 1997, № 48, с. 36–40, 42–51.
16. Иванов М.А., Чугунков И.В. Теория, применение и оценка качества генераторов псевдослучайных последовательностей. Серия СКБ (специалисту по компьютерной безопасности). Кн. 2. М.: Кудиц-Образ, 2003.
17. Коблиц Н. Курс теории чисел и криптографии. М.: ТВП, 2001.
18. Кнут Д. Искусство программирования, т. 2. Получисленные алгоритмы, 3-е изд.: Пер. с англ.: Учебное пособие. М.: ИД «Вильямс», 2000.
19. Корн Г., Корн Т. Справочник по математике для научных работников и инженеров: Пер. с англ. / под ред. И.Г. Арамановича. М.: Наука, 1973.
20. Макуильямс Ф.Дж., Слоан Н.Дж.А. Псевдослучайные последовательности и таблицы. // ТИИЭР, 1976, № 12, с. 80–95.
21. Мао В. Современная криптография: теория и практика: Пер. с англ. М.: ИД «Вильямс», 2005.
22. Мельников В.В. Защита информации в компьютерных системах. М.: Финансы и статистика; Электронинформ, 1997.
23. Можно ли взломать 512-разрядный RSA? Compute-Review, 22 сентября 1999 г., с. 14.
24. Поточные шифры / А.А. Асосков, М.А. Иванов, А.Н. Тютвин и др. Серия СКБ (специалисту по компьютерной безопасности). Кн. 3. М.: Кудиц-Образ, 2003.
25. Романец Ю.В., Тимофеев П.А., Шаньгин В.Ф. Защита информации в компьютерных системах и сетях / Под ред. В.Ф. Шаньгина. М.: Радио и связь, 1999.
26. Семьянов П.В. Почему криптосистемы ненадежны? // Проблемы информационной безопасности. Компьютерные системы. № 1, 1999, с. 70–82.

27. Слоан Н.Дж.А. Коды, исправляющие ошибки и криптография. В кн.: Математический цветник/ Сост. и ред. Д.А. Кларнер; Пер. с англ. Данилова Ю.А.; Под ред. И.М. Яглома. М.: Мир, 1983.
28. Столингс В. Криптография и защита сетей: принципы и практика. 2-е изд.: Пер. с англ. М.: ИД «Вильямс», 2001.
29. Стохастические методы и средства защиты информации в компьютерных системах и сетях / М.А. Иванов, Д.М. Михайлов, И.В. Чугунков и др.; Под ред. И.Ю. Жукова. М.: Кудиц-Пресс, 2009.
30. Фергюсон Н., Шнайер Б. Практическая криптография: Пер. с англ. М.: ИД «Вильямс», 2005.
31. Шеннон К. Математическая теория связи. Работы по теории информации и кибернетике. М., 1963.
32. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.: Триумф, 2002.
33. A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST Special Publications 800-22. May 15, 2001.
34. Bajalcaliev K. Stream Cipher design postulates/SQ model. http://eon.pmf.ukim.edu.mk/~kbajalc
35. Certicom Research, Standards for efficient cryptography, SEC 1: Elliptic Curve Cryptography, Version 1.0, 2000.
36. Digital signature standard (DSS). FIPS 186-2. Federal information processing standards publication. U.S. department of commerce/National Institute of Standards and Technology. 2000.
37. Escott A., Sager J., Selkirk A., Tsapakidis D.. Attacking Elliptic Curve Cryptosystems Using the Parallel Pollard rho Method. CryptoBytes 2:2, 1999.
38. Gong G., Berson T., Stinson D. Elliptic Curve Pseudorandom Sequence Generators. University of Waterloo, Canada, 1998. http://www.anagram.com/berson/ecpsg99.pdf
39. Gong G., Lam C. Linear Recursive Sequences over Elliptic Curves. – Proceedings of Sequences and Their Applications-SETA'01. DMTCS series. Berlin: Spring-Verlag, 2001, pp.182–196.

40. Hallgren S. Linear Congruental Generators Over Elliptic Curves. Technical Report CS-94-143, Cornegie Mellon University, 1994.
41. Jelly A. Криптографический стандарт в новом тысячелетии. – BYTE, Россия, 6.06.1999.
42. Jurisic A, Menezes A. Elliptic Curves and Cryptography. – Dr. Dobb's Journal, April 1997.
43. Koblitz N. Elliptic curve cryptosystems. Mathematics of Computation 48, 1987, pp. 203–209.
44. Marsaglia G. DIEHARD Statistical Tests. http://stat.fsu.edu/~geo/diehard.html.
45. Menezes A. Elliptic Curve Public Key Cryptosystem. Kluwer Academic Publishers, 1993.
46. Menezes A. Evaluation of Security Level of Cryptography: The Elliptic Curve Discrete Logarithm Problem (ECDLP). University of Waterloo. 2001. http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1028_ecdlp.pdf
47. Miller V. Use of elliptic curves in cryptography. CRYPTO 85, 1985.
48. NESSIE: New europian schemes for signatures, integrity and encryption. http://www.cryptonessie.org
49. Schneier B. Why Cryptography Is Harder Than It Looks. http://www.schneier.com/essay-037.pdf
50. Sherif M.H. Protocols for Secure Electronic Commerce. CRC Press, 2000.

Контрольно-измерительные материалы по курсу «Методы и средства защиты компьютерной информации»

Примечание. На каждый вопрос может быть дано от одного до четырех правильных ответов. Необходимо найти все правильные ответы.

Заключительная проверка

1. Укажите ложное утверждение:

А. В качественной хеш-функции не должно быть коллизий.
Б. В случае использования качественной хеш-функции минимальное изменение на ее входе должно приводить в среднем к изменению 50 % бит хеш-образа.
В. В случае использования качественной хеш-функции любое изменение на ее входе должно приводить в среднем к изменению 50 % бит хеш-образа.
Г. При использовании качественной хеш-функции задача нахождения коллизий вычислительно неразрешима.

2. Укажите, какие из перечисленных действий пользователя снижают защищенность системы:

А. Использование секретных паролей в несекретных системах.
Б. Формирование паролей по принципу удобства запоминания.
В. Информирование администратора об утерянной или скомпрометированной ключевой информации.
Г. Хранение паролей в зашифрованном виде.

3. Укажите ложные утверждения:

А. Система обеспечения безопасности информации (ОБИ) надежна настолько, насколько надежно ее самое слабое звено.

Б. Система ОБИ надежна настолько, насколько надежно ее самое сильное звено.
В. Прочность любого звена системы ОБИ зависит от навыков противника и имеющихся у него ресурсов.
Г. Укрепление любого звена системы ОБИ, кроме самого слабого, – пустая трата времени.

4. Укажите области предпочтительного использования режима блочного шифрования CBC:

А. Шифрование ключевой информации симметричных блочных криптоалгоритмов.
Б. Шифрование баз данных с произвольным доступом к отдельным записям.
В. Формирование кода MAC.
Г. Шифрование сообщений большой длины в асимметричных криптосистемах.

5. Укажите соображения, которые должны учитываться при определении времени жизни ключевой информации:

А. Чем дольше используется ключ, тем больше вероятность его компрометации.
Б. Чем дольше используется ключ, тем больший потенциальный ущерб может нанести его компрометация.
В. Чем больший объем информации, зашифрованной на одном ключе, перехватывает противник, тем легче проводить атаку на ключ.
Г. При длительном использовании ключа у противника появляется дополнительный стимул потратить на его вскрытие значительные ресурсы, так как выгода в случае успеха оправдает все затраты.

6. Укажите области предпочтительного использования режима блочного шифрования ECB:

А. Шифрование ключевой информации симметричных блочных криптоалгоритмов.
Б. Шифрование баз данных с произвольным доступом к отдельным записям.
В. Формирование кода MAC.
Г. Шифрование сообщений большой длины в асимметричных криптосистемах.

7. Укажите утверждения, которые строго математически доказаны:

А. Непредсказуемый влево генератор псевдослучайных чисел является криптографически сильным.
Б. Непредсказуемый влево генератор псевдослучайных чисел существует тогда и только тогда, когда существуют односторонние функции.
В. Сложность взлома криптосистемы RSA эквивалентна сложности разложения большого целого числа на простые сомножители.
Г. Схема однократного использования является абсолютно стойкой.

8. Какие режимы симметричного блочного шифрования требуют наличия и функции зашифрования E_{AB}, и функции расшифрования D_{AB}?

А. ECB.
Б. CBC.
В. OFB.
Г. CFB.
Д. Counter Mode.

9. Какие из перечисленных схем шифрования являются схемами гаммирования с обратной связью?

А. AES в режиме CFB.
Б. RC4.
В. ECCS.
Г. RSA.

10. Укажите решения, предполагающие использование генераторов псевдослучайных чисел для внесения неопределенности в результат работы криптоалгоритмов:

А. Гаммирование.
Б. Вероятностное шифрование.
В. Технология OAEP.
Г. Несепарабельные режимы симметричного шифрования.

11. Укажите особенности синхронного поточного шифрования:

А. У противника, не знающего ключа, всегда есть возможность вносить предсказуемые изменения в зашифрованный текст.
Б. При повторном использовании ключа противник получает возможность использовать для взлома шифра частотный анализ.
В. Результат шифрования каждого элемента сообщения зависит от всех предшествующих элементов сообщения.
Г. Если при передаче информации по каналу связи происходит «выпадение» или «вставка», на принимающей стороне происходит необратимая потеря информации.

Тема «Симметричные криптосистемы»

1. Укажите криптоалгоритмы, в которых используется один фиксированный *S*-блок:

А. DES.
Б. RIJNDAEL.
В. RC4.
Г. ГОСТ 28147-89.

2. Укажите криптоалгоритмы, в которых в качестве ключевой информации используется таблица замен *S*-блока, изменяющаяся в процессе шифрования:

А. DES.
Б. RIJNDAEL.
В. RC4.
Г. ГОСТ 28147-89.

3. Укажите разрядность ключевой информации в криптоалгоритме ГОСТ 28147-89:

А. 256 бит.
Б. 768 бит.
В. 56 бит.
Г. Другое.

4. В совершенно секретных криптосистемах после анализа шифротекста противником:

А. Апостериорные вероятности некоторых открытых текстов возрастают относительно их априорных вероятностей.
Б. Апостериорные вероятности всех возможных открытых текстов не меняются относительно их априорных вероятностей.

В. Апостериорные вероятности некоторых открытых текстов снижаются до нуля.
Г. Правильного ответа нет.

5. Укажите криптоалгоритмы, архитектура которых называется сетью Фейстеля:

А. DES.
Б. ГОСТ 28147-89.
В. AES.
Г. MARS.

6. Какие из перечисленных схем шифрования являются схемами гаммирования?

А. AES в режиме OFB.
Б. RC4.
В. A5.
Г. ГОСТ 28147-89 в режиме простой замены.

7. Укажите ложные утверждения:

А. Схема абсолютно стойкого шифра суть схема гаммирования с обратной связью.
Б. При использовании абсолютно стойкого шифра у противника, не знающего ключа, всегда есть возможность вносить предсказуемые изменения в зашифрованный текст.
В. При использовании для шифрования схемы гаммирования наложение псевдослучайной последовательности на последовательность открытых данных может осуществляться только с использованием функции XOR.
Г. Утверждение Б является истинным.

8. На основе какого блочного шифра разработан стандарт криптографической защиты AES?

А. RC6.
Б. MARS.
В. SERPENT.
Г. TWOFISH.
Д. Другое.

9. Укажите ложные утверждения:

А. При синхронном поточном шифровании результат шифрования каждого элемента открытого текста зависит от позиции этого элемента во входном потоке данных.
Б. При синхронном поточном шифровании результат шифрования каждого элемента открытого текста зависит от всех предшествующих элементов во входном потоке данных.
В. Синхронное поточное шифрование суть наложение псевдослучайной последовательности на входную информационную последовательность.
Г. При синхронном поточном шифровании у противника, не знающего ключа, всегда есть возможность вносить предсказуемые изменения в зашифрованный текст.

10. Укажите ложные утверждения:

А. При самосинхронизирующемся поточном шифровании результат шифрования каждого элемента открытого текста зависит от позиции этого элемента во входном потоке данных.
Б. При самосинхронизирующемся поточном шифровании результат шифрования каждого элемента открытого текста зависит от всех предшествующих элементов во входном потоке данных.

В. Самосинхронизирующееся поточное шифрование суть наложение псевдослучайной последовательности на входную информационную последовательность.
Г. При самосинхронизирующемся поточном шифровании у противника, не знающего ключа, всегда есть возможность вносить предсказуемые изменения в зашифрованный текст.

Тема «Асимметричные криптосистемы»

1. Стойкость какой криптосистемы основывается на сложности решения задачи об укладке рюкзака?

А. ECCS.
Б. RSA.
В. AES.
Г. Другое.

2. Укажите криптосистему, на основе которой разработаны государственные стандарты России и США на ЭЦП:

А. AES.
Б. ECCS.
В. RSA.
Г. Криптосистема Эль-Гамаля.

3. Укажите ложные утверждения:

А. В схеме слепой ЭЦП трижды применяется операция шифрования, при этом дважды используется открытый ключ и один раз – закрытый ключ подписывающего.
Б. В схеме слепой ЭЦП трижды используется операция шифрования: для преобразования затемняющего множителя, формирования ЭЦП и проверки ЭЦП.
В. Шифрование затемняющего множителя в схеме слепой ЭЦП необходимо для обеспечения его секретности.

Г. Шифрование затемняющего множителя в схеме слепой ЭЦП необходимо для обеспечения возможности его снятия после формирования ЭЦП.

4. Какие режимы симметричного блочного шифрования могут использоваться при асимметричном шифровании?

А. Counter Mode.
Б. CBC.
В. OFB.
Г. CFB.

5. Укажите ложные утверждения:

А. Режим простой замены не может использоваться при асимметричном шифровании.
Б. Режим гаммирования не может использоваться при асимметричном шифровании.
В. Режим гаммирования с обратной связью не может использоваться при асимметричном шифровании.
Г. Шифрование предназначено только для обеспечения секретности информации.

6. На чем основана стойкость криптосистем с открытым ключом?

А. На секретности ключа зашифрования.
Б. На секретности ключа расшифрования.
В. На сложности решения некой математической задачи.
Г. На секретности алгоритма расшифрования.

7. На чем основана стойкость протокола выработки общего секретного ключа?

А. На сложности решения задачи дискретного логарифмирования.
Б. На сложности решения задачи разложения большого целого числа на простые сомножители.
В. На качестве используемых генераторов псевдослучайных чисел.
Г. На сложности решения задачи об укладке рюкзака.

8. Что такое гибридное шифрование?

А. Симметричное шифрование сообщения на сеансовом ключе, который в дальнейшем шифруется с использованием открытого ключа получателя.
Б. Выработка двумя удаленными абонентами общего секретного ключа, который в дальнейшем используется для симметричного шифрования.
В. Последовательное использование симметричного, а затем асимметричного шифрования.
Г. Последовательное использование асимметричного, а затем симметричного шифрования.

9. Для защиты интересов владельца цифровой купюры в централизованной платежной системе используются:

А. Хеширование прекурсора для получения серийного номера цифровой купюры.
Б. Слепая ЭЦП банка-эмитента на серийном номере цифровой купюры.
В. Ведение списка серийных номеров ранее использованных цифровых купюр.
Г. Правильного ответа нет.

10. Для защиты от повторного использования цифровой купюры в централизованной платежной системе применяют:

А. Хеширование прекурсора для получения серийного номера цифровой купюры.
Б. Слепую ЭЦП банка-эмитента на серийном номере цифровой купюры.
В. Ведение списка серийных номеров ранее использованных цифровых купюр.
Г. Правильного ответа нет.

11. Для обеспечения анонимности и неотслеживаемости платежей в централизованной платежной системе используются:

А. Хеширование прекурсора для получения серийного номера цифровой купюры.
Б. Слепая ЭЦП банка-эмитента на серийном номере цифровой купюры.
В. Ведение списка серийных номеров ранее использованных цифровых купюр.
Г. Правильного ответа нет.

12. Для защиты от подделки номинала цифровой купюры в централизованной платежной системе используются:

А. Хеширование прекурсора для получения серийного номера цифровой купюры.
Б. Слепая ЭЦП банка-эмитента на серийном номере цифровой купюры.
В. Ведение списка серийных номеров ранее использованных цифровых купюр.
Г. Правильного ответа нет.

13. В двухключевой криптосистеме для обеспечения секретности информации, пересылаемой от абонента *A* к абоненту *B*, применяют:

А. Открытый ключ *A*.
Б. Открытый ключ *B*.
В. Закрытый ключ *A*.
Г. Открытый ключ *B*.

14. Для формирования классической ЭЦП на сообщении, пересылаемом от абонента *A* к абоненту *B*, используется:

А. Открытый ключ *A*.
Б. Открытый ключ *B*.
В. Закрытый ключ *A*.
Г. Открытый ключ *B*.

15. Для проверки классической ЭЦП на сообщении, пересылаемом от абонента *A* к абоненту *B*, используется:

А. Открытый ключ *A*.
Б. Открытый ключ *B*.
В. Закрытый ключ *A*.
Г. Открытый ключ *B*.

16. Укажите ложные утверждения:

А. Недостаток двухключевых криптосистем – отсутствие юридической значимости пересылаемых электронных документов.
Б. Недостаток двухключевых криптосистем – низкое быстродействие.
В. Недостаток двухключевых криптосистем – возможность подмены открытых ключей.
Г. Для построения двухключевой критптосистемы используется односторонняя функция.

17. Укажите свойства классической ЭЦП:

А. Документ, подписанный ЭЦП, можно копировать сколь угодно много раз.
Б. ЭЦП невозможно подделать.
В. ЭЦП можно хранить и пересылать отдельно от документа.
Г. Правильного ответа нет.

Тема «Программно-аппаратные методы защиты информации»

1. Укажите истинные утверждения:

А. Одним из требований к безопасному ПО является недостаточность функциональности.
Б. Тестирование ПО может показать только наличие ошибок, а не их отсутствие.
В. Перезаписывание данных необязательно приводит к их удалению.
Г. Истинных утверждений нет.

2. Укажите методы обнаружения неизвестных компьютерных вирусов:

А. Сигнатурный анализ.
Б. Эвристический анализ.
В. Мониторинг и блокировка потенциально опасных действий.
Г. Обнаружение несанкционированных изменений файлов и системных областей.

3. Что такое ошибка 1-го рода при функционировании антивирусных средств?

А. Необнаружение компьютерного вируса.

Б. Ложное срабатывание («обнаружение» компьютерного вируса в незараженном файле).
В. Обнаружение «не того» компьютерного вируса.
Г. Другое.

4. Что такое ошибка 2-го рода при функционировании антивирусных средств?

А. Необнаружение компьютерного вируса.
Б. Ложное срабатывание («обнаружение» компьютерного вируса в незараженном файле).
В. Обнаружение «не того» компьютерного вируса.
Г. Другое.

5. Суть технологии Honeypot – это:

А. Внесение неопределенности в результат работы алгоритма защиты информации.
Б. Внесение неопределенности в последовательность выполнения шагов алгоритма защиты информации.
В. Внесение неопределенности в длительность выполнения шагов алгоритма защиты.
Г. Правильного ответа нет.

6. Что такое преобразованный канал связи в теории стохастических кодов?

А. Совокупность кодера, блоков прямого и обратного стохастического преобразований и декодера.
Б. Совокупность дискретного канала, блоков прямого и обратного стохастического преобразований.
В. Совокупность блока обратного стохастического преобразования и декодера.
Г. Правильного ответа нет.

7. Какой блок в схеме стохастического кодирования определяет свойства преобразованного канала связи?

А. Кодер.
Б. Блок прямого стохастического преобразования.
В. Блок обратного стохастического преобразования.
Г. Декодер.

8. Что такое ошибка 3-го рода при функционировании антивирусных средств?

А. Необнаружение компьютерного вируса.
Б. Ложное срабатывание («обнаружение» компьютерного вируса в незараженном файле).
В. Обнаружение «не того» компьютерного вируса.
Г. Другое.

9. Укажите код, который способен обнаруживать как случайные, так и умышленные искажения в каналах связи:

А. Стохастический код Осмоловского.
Б. Код Рида–Соломона.
В. Сверточный код.
Г. Код Хэмминга.

10. Укажите свойства кода с минимальным кодовым расстоянием 5:

А. Обнаружение и исправление всех ошибок кратности не больше 2.
Б. Обнаружение всех ошибок кратности не больше 3.
В. Обнаружение и исправление всех ошибок кратности не больше 3.
Г. Обнаружение всех ошибок кратности не больше 5.

11. Укажите ложные утверждения:

А. Компьютерный вирус – это разрушающее программное воздействие (РПВ), которое не может активизироваться без участия пользователя.
Б. Сетевой червь – это РПВ, которое не может активизироваться без участия пользователя.
В. Эксплойт (exploit) – это РПВ, которое в состоянии выполнить произвольный код на атакованной машине.
Г. Перезаписывающие компьютерные вирусы – это РПВ, которые необратимо искажают поражаемые информационные объекты.

12. Укажите причины появления уязвимостей программного кода типа «buffer overflow», «race condition» и т.п.:

А. Отсутствие обработки нестандартных ситуаций (неопределенный ввод, ошибки пользователей и пр.), возникающих в процессе работы программ.
Б. Неправильная обработка нестандартных ситуаций (неопределенный ввод, ошибки пользователей и пр.), возникающих в процессе работы программ.
В. Некачественные процедуры отладки и тестирования программ.
Г. Расчет при написании программы на «хорошего» пользователя, который будет работать с программой именно так, как предполагает программист.

13. Укажите методы обнаружения компьютерных вирусов до момента их активизации:

А. Сигнатурный анализ.
Б. Эвристический анализ.
В. Мониторинг и блокировка потенциально опасных действий.
Г. Обнаружение несанкционированных изменений файлов и системных областей.

14. Укажите, какие из перечисленных методов защиты информации являются организационными:

А. Обязательное сканирование всей входящей информации.
Б. Ограничение доступа.
В. Разделение доступа.
Г. Мониторинг и блокировка потенциально опасных действий.

15. Укажите технологии, использование которых предотвращает возможность запуска произвольного кода на атакуемом компьютере:

А. Honeypot.
Б. Технология безопасного программирования.
В. Honeynet.
Г. Рандомизация среды исполнения программы.

Тема «Математические основы»

1. Чему равно произведение элементов 3 и 8 конечного поля *GF*(23)?

А. 1.
Б. 24.
В. 11.
Г. Правильного ответа нет.

2. Чему равна сумма элементов 16 и 15 конечного поля *GF*(23)?

А. 1.
Б. 31.
В. 240.
Г. Правильного ответа нет.

3. Задано конечное поле $GF(23)$. Найдите значение 8^{-1}.

А. 4.
Б. 3.
В. 5.
Г. Правильного ответа нет.

4. Задано конечное поле $GF(23)$. Найдите значение $(6^{-1} – 3) : 8$.

А. 4.
Б. 3.
В. 5.
Г. Правильного ответа нет.

5. Задано конечное поле $GF(23)$. Найдите значение $(3 - 6^{-1}) : 11$.

А. 22.
Б. 3.
В. 2.
Г. Правильного ответа нет.

6. Задано конечное поле $GF(2^3) = \{0, 1, \omega, \omega^2, \omega^3, \omega^4, \omega^5, \omega^6\}$. Найдите значение $\omega^3 \times \omega^4$.

А. 000.
Б. 111.
В. 001.
Г. Правильного ответа нет.

7. Характеристический многочлен CRC-генератора имеет вид $x^8 + x^7 + x^5 + x^3 + 1$. Укажите вектора необнаруживаемых ошибок.

А. 01101010010.
Б. 10101001000.

В. 10111110110.
Г. 11010100100.

8. Чему равно произведение двух элементов 00001111 и 00000101 конечного поля $GF(2^8)$?

А. 00111100.
Б. 00111001.
В. 00110011.
Г. Правильного ответа нет.

9. Один из перечисленных ниже двоичных многочленов не является неприводимым. Укажите его.

А. $x^{65} + x^{18} + 1$.
Б. $x^{16} + x^{12} + x^9 + x^7 + 1$.
В. $x^{63} + x^{32}$.
Г. $x^{3217} + x^{3150} + 1$.

10. Укажите ложные утверждения:

А. Примитивный многочлен всегда является неприводимым.
Б. Неприводимый многочлен всегда является примитивным.
В. Неприводимый двоичный многочлен делится нацело только на самого себя и единицу.
Г. Число делителей многочлена $x^2 + x + 1$, примитивного над $GF(3)$, равно 4.

11. Число делителей многочлена, примитивного над GF(p), равно:

А. p^2.
Б. $2(p - 1)$.
В. $p(p - 1)$.
Г. Правильного ответа нет.

Тема «Стохастические методы»

1. Укажите истинное утверждение:

А. Энтропия 128-разрядного ключа, сформированного из 10-символьного пароля (состоящего из ASCII-символов), меньше 80 бит.
Б. Энтропия 128-разрядного ключа, сформированного из 10-символьного пароля (состоящего из ASCII-символов), равна 80 бит.
В. Энтропия 128-разрядного ключа, сформированного из 10-символьного пароля (состоящего из ASCII-символов), равна 128 бит.
Г. Правильного ответа нет.

2. Укажите, какие задачи обеспечения безопасности информации решаются стохастическими методами:

А. Обеспечение секретности информации.
Б. Обеспечение аутентичности субъектов информационного взаимодействия.
В. Защита прав владельцев информации.
Г. Оперативный контроль работоспособности компонентов системы.

3. Укажите, какие задачи обеспечения безопасности информации решаются стохастическими методами:

А. Неотслеживаемость информационных потоков в системе.
Б. Обеспечение аутентичности объектов информационного взаимодействия.
В. Оперативный контроль за процессами управления, обработки и передачи информации.
Г. Защита от несанкционированного доступа.

4. Укажите функции генераторов псевдослучайных чисел в системах обеспечения безопасности информации:

А. Генерация ключей и паролей пользователей.
Б. Формирование запросов при аутентификация пользователей по принципу «запрос – ответ».
В. Формирование контрольных кодов целостности информации.
Г. Внесение неопределенности в работу средств и объектов защиты.

5. Укажите, какие из перечисленных методов защиты информации – стохастические:

А. Разграничение доступа.
Б. Ограничение доступа.
В. Разделение доступа.
Г. Мониторинг и блокировка потенциально опасных действий.

6. Укажите, какие из перечисленных методов защиты информации – стохастические:

А. Межсетевое экранирование.
Б. Обнаружение вторжений.
В. Эвристический анализ.
Г. Контроль хода выполнения программ.

7. Укажите, какие протоколы удаленного взаимодействия абонентов являются стохастическими:

А. Протокол ЭЦП.
Б. Протокол доказательства с нулевым разглашением знаний.
В. Симметричный протокол аутентификации Нидхэма–Шредера.
Г. Асимметричный протокол аутентификации Диффи–Хеллмана.

8. Укажите, какие протоколы удаленного взаимодействия абонентов являются стохастическими:

А. Протокол выработки общего секретного ключа.
Б. Протокол разделения секрета.
В. Протокол привязки к биту.
Г. Протокол подбрасывания монеты.

9. Укажите свойства, присущие стохастическому коду Осмоловского:

А. Возможность обеспечения любой наперед заданной вероятности правильного приема информации.
Б. Возможность обеспечения секретности информации.
В. Возможность обеспечения аутентичности информации.
Г. Правильного ответа нет.

Приложение 1

Неприводимые многочлены над $GF(p)$, p – простое

Примечания:

1. Многочлены заданы набором их коэффициентов

$$a_N\ a_{N-1} \ldots\ a_i\ \ldots\ a_2\ a_1\ a_0$$

(например, набор 1021 соответствует многочлену $x^3 + 2x + 1$).

2. В скобках указан показатель многочлена, т.е. наименьшее положительное число e, при котором $x^e - 1$ делится на данный многочлен без остатка.

3. Многочлены $f^*(x)$, для которых справедливо соотношение

$$f^*(x) = \alpha x^N f(x^{-1}),$$

где $\alpha \in GF(p)$, $\alpha \neq 0$, $f(x)$ – многочлен степени N, уже имеющийся в списке, не приводятся.

Неприводимые многочлены над $GF(2)$:

$N = 1$

11 (1)

$N = 2$

111 (3)

$N = 3$

1011 (7)

$N = 4$

10011 (15)
11111 (5)

$N = 5$

100101 (31)
101111 (31)
110111 (31)

$N = 6$

1000011 (63)
1001001 (9)
1010111 (21)
1011011 (63)
1100111 (63)

$N = 7$

10000011 (127)
10001001 (127)
10001111 (127)
10011101 (127)
10100111 (127)
10101011 (127)
10111111 (127)
11001011 (127)
11101111 (127)

$N = 8$

100011011 (51)
100011101 (255)
100101011 (255)
100101101 (255)
100111001 (17)
100111111 (85)
101001101 (255)
101011111 (255)
101100011 (255)
101110111 (85)
101111011 (85)
110000111 (255)
110001011 (85)
110011111 (51)
111001111 (255)
111010111 (17)

Неприводимые многочлены над $GF(3)$:

$N = 1$

11 (2)
12 (1)

$N = 2$

101 (4)
112 (8)

$N = 3$

1021 (26)
1022 (13)
1112 (13)
1121 (26)

$N = 4$

10012 (80)
10022 (80)
10102 (16)
10111 (40)
10121 (40)
11021 (20)
11111 (5)
11122 (80)
12112 (80)
12121 (10)

$N = 5$

100021 (242)
100022 (121)
100112 (121)
100211 (242)
101011 (242)
101012 (121)
101102 (121)
101122 (121)
101201 (242)
101221 (242)
102112 (121)
102122 (11)
102211 (242)
102221 (22)
110012 (121)
110021 (242)
110111 (242)
110122 (121)
111121 (242)
111211 (242)
111212 (121)
120212 (121)
120221 (242)
121112 (121)

Неприводимые многочлены над $GF(5)$:

$N = 1$

11 (2)
12 (4)
14 (1)

$N = 2$

102 (8)
111 (3)
112 (24)
123 (24)
124 (12)
141 (6)

$N = 3$

1011 (62)
1014 (31)
1021 (62)
1024 (31)
1032 (124)
1033 (124)
1042 (124)
1043 (124)
1113 (124)
1114 (31)
1131 (62)
1134 (31)
1141 (62)
1143 (124)
1213 (124)
1214 (31)
1223 (124)
1312 (124)
1323 (124)
1341 (62)

Неприводимые многочлены над $GF(7)$:

$N = 1$

11 (2)
12 (6)
13 (3)
16 (1)

$N = 2$

101 (4)
102 (12)
113 (48)
114 (24)
116 (16)
123 (48)
125 (48)
131 (8)
136 (16)
141 (8)
145 (48)
152 (24)

$N = 3$

1002 (18)
1003 (9)
1011 (114)
1016 (57)
1021 (38)
1026 (19)
1032 (342)
1035 (171)
1041 (114)
1046 (57)
1052 (342)
1055 (171)
1062 (342)
1065 (171)
1112 (342)
1115 (171)
1124 (342)
1126 (57)
1131 (38)
1135 (171)
1143 (171)
1146 (57)
1151 (114)
1152 (342)
1153 (171)
1154 (342)
1163 (171)
1165 (171)
1214 (342)
1216 (57)
1223 (171)
1226 (57)
1235 (171)
1245 (171)
1251 (114)
1261 (114)
1262 (342)
1264 (342)
1325 (171)
1334 (342)
1336 (19)
1341 (38)
1343 (171)
1352 (342)
1354 (342)
1413 (171)
1416 (19)
1425 (171)
1432 (342)
1434 (342)
1453 (171)
1461 (114)
1513 (171)
1532 (342)
1534 (342)
1552 (342)

Неприводимые многочлены над $GF(11)$:

$N = 1$

11 (2)
12 (5)
13 (10)
15 (10)
17 (5)
1A (1)

$N = 2$

101 (4)
103 (20)
105 (20)
111 (3)
114 (60)
116 (40)
117 (120)
118 (120)
124 (30)
125 (60)
126 (120)
12A (24)
136 (120)
138 (120)
139 (15)
13A (8)
147 (120)
148 (40)
149 (60)
151 (12)
152 (120)
153 (15)
157 (40)
15A (24)
161 (12)
172 (120)
175 (30)
183 (60)
192 (40)
1A1 (6)

Приложение 2

Примитивные многочлены над *GF*(2)

Примечание: многочлены заданы индексами *N*, *i*, 0 своих ненулевых коэффициентов, например, строка 89 38 0 соответствует многочлену $x^{89} + x^{38} + 1$.

2 1 0
3 1 0
4 1 0
5 2 0
6 1 0
7 1 0
8 6 5 1 0
9 4 0
10 3 0
11 2 0
12 7 4 3 0
13 4 3 1 0
13 4 3 1 0
14 12 11 1 0
15 1 0
16 5 3 2 0
17 3 0
18 7 0
19 6 5 1 0
20 3 0
21 2 0
22 1 0
23 5 0
24 4 3 1 0
25 3 0
26 8 7 1 0
27 8 7 1 0
28 3 0
29 2 0
30 16 15 1 0
31 3 0
34 15 14 1 0
35 2 0
36 11 0
37 12 10 2 0
38 6 5 1 0
39 4 0
40 21 19 2 0
41 3 0
42 23 22 1 0
43 6 5 1 0
44 27 26 1 0
45 4 3 1 0
46 21 20 1 0
47 5 0
48 28 27 1 0
49 9 0
50 27 26 1 0
51 16 15 1 0
52 3 0
53 16 15 1 0
54 37 36 1 0
55 24 0
56 22 21 1 0
57 7 0
58 19 0
59 22 21 1 0
60 1 0
61 16 15 1 0
62 57 56 1 0
63 1 0
64 4 3 1 0
67 10 9 1 0
68 9 0
69 29 27 2 0
70 16 15 1 0
71 6 0
72 53 47 6 0
73 25 0
74 16 15 1 0
75 11 10 1 0
76 36 35 1 0
77 31 30 1 0
78 20 19 1 0
79 9 0
80 38 37 1 0
81 4 0
82 38 35 3 0
83 46 45 1 0
84 13 0
85 28 27 1 0
86 13 12 1 0
87 13 0
88 72 71 1 0
89 38 0
90 19 18 1 0
91 84 83 1 0
92 13 12 1 0
93 2 0
94 21 0
95 11 0
96 49 47 2 0
97 6 0

32 28 27 1 0	65 18 0	98 11 0
33 13 0	66 10 9 1 0	

Примитивные многочлены вида $x^N + x^i + 1$, где N – число Мерсенна:

17 11 0	127 90 0	607 460 0
17 12 0	127 97 0	607 502 0
17 14 0	127 112 0	1279 861 0
31 18 0	127 120 0	1279 1063 0
31 24 0	521 353 0	2281 1252 0
31 25 0	521 363 0	2281 1366 0
31 28 0	521 473 0	2281 1566 0
89 38 0	521 489 0	3217 2641 0
127 64 0	607 334 0	3217 3150 0

Примитивные многочлены вида $x^N + x^i + 1$, где i = 8, 16, 32, 64, 128:

15 8 0	65 32 0	105 16 0	135 16 0	177 8 0
39 8 0	81 16 0	119 8 0	159 128 0	225 32 0
63 32 0	97 64 0	127 64 0	175 16 0	521 320

Примитивные многочлены вида $x^N + x^i + 1$, где $(i, 2^N – 1) = 1$:

15 7 0	73 31 0	135 22 0	207 43 0	378 43 0
18 7 0	79 9 0	140 29 0	212 105 0	378 107 0
20 3 0	79 19 0	142 21 0	218 11 0	380 47 0
23 5 0	81 35 0	145 52 0	218 15 0	390 89 0
23 9 0	84 13 0	145 69 0	218 71 0	396 25 0
25 3 0	87 13 0	148 27 0	218 83 0	396 109 0
25 7 0	94 21 0	150 53 0	225 74 0	396 169 0
28 3 0	95 11 0	151 3 0	225 88 0	396 175 0
28 9 0	95 17 0	151 9 0	225 97 0	404 189 0
28 13 0	97 6 0	151 15 0	225 109 0	412 147 0
33 13 0	97 12 0	151 31 0	231 26 0	428 105 0
36 11 0	97 33 0	151 39 0	231 34 0	436 165 0
39 14 0	97 34 0	151 43 0	234 31 0	460 61 0
41 3 0	98 11 0	151 45 0	234 103 0	462 73 0

41 20 0
47 5 0
47 14 0
47 20 0
47 21 0
49 9 0
49 12 0
49 15 0
49 22 0
52 19 0
52 21 0
55 24 0
57 7 0
57 22 0
58 19 0
60 11 0
63 5 0
63 31 0
65 18 0
68 9 0
68 33 0
71 6 0
71 9 0
71 18 0
71 20 0
71 35 0
73 25 0
73 28 0
98 27 0
100 37 0
103 9 0
103 13 0
103 30 0
103 31 0
105 17 0
105 37 0
105 43 0
105 52 0
106 15 0
108 31 0
111 10 0
111 49 0
113 9 0
113 15 0
113 30 0
118 33 0
118 45 0
119 38 0
121 18 0
129 5 0
129 31 0
129 46 0
130 8 0
132 29 0
134 57 0
135 11 0
151 51 0
151 63 0
151 66 0
151 67 0
151 70 0
159 31 0
159 34 0
159 40 0
161 18 0
161 39 0
161 60 0
170 23 0
172 7 0
174 13 0
175 6 0
175 18 0
175 57 0
177 22 0
177 88 0
178 87 0
183 56 0
194 87 0
198 65 0
201 14 0
201 17 0
201 59 0
201 79 0
202 55 0
236 5 0
250 103 0
252 67 0
255 52 0
255 55 0
255 82 0
258 83 0
266 47 0
268 25 0
268 61 0
270 53 0
270 133 0
282 35 0
282 43 0
284 119 0
286 69 0
286 73 0
292 97 0
294 61 0
300 7 0
300 73 0
300 91 0
316 135 0
322 67 0
332 123 0
350 53 0
364 67 0
366 29 0
475 15 0
476 141 0
484 105 0
508 109 0
524 167 0
532 37 0
540 179 0
540 211 0
564 163 0
588 151 0
588 253 0
708 278 0
708 301 0
756 119 0
756 349 0
780 299 0
804 295 0
828 205 0

Приложение 3

Криптоалгоримы с архитектурой «Куб»

Трехмерный блок замены

Блоки замены (*S*-блоки) являются важнейшим элементом криптографических примитивов. Именно от качества используемых *S*-блоков зависят непредсказуемость формируемых псевдослучайных последовательностей и криптостойкость алгоритмов хеширования, блочного и поточного шифрования. Рассматриваются два новых способа построения *S*-блоков, блоки замены могут быть названы 2*D* и 3*D S*-блоками соответственно.

Алгоритм функционирования 2D S-блока. Представим входные и выходные блоки данных, а также все промежуточные результаты преобразований в виде квадратного массива битов размерностью $N \times N$, где N – разрядность используемых узлов замены. Таким образом, объем ключевой информации, однозначно определяющей логику работы каждого узла замены, равен $N \times 2^N$.

Последовательность выполнения операции $A = SubSquare[A]$ (или, кратко, $A = S_{sq}[A]$), замены квадратного массива битов A размерностью $N \times N$, имеет следующий вид:

1) разбиение входного массива A на N строк R_i длины N,
$$i = 0, 1, ..., (N - 1);$$
2) замена строк *SubRows*, т.е. преобразование каждого i-го N-разрядного двоичного набора R_i с использованием соответствующего узла замены S_i: $R_i = S_i[R_i]$;
3) разбиение получившегося массива $A = SubRows[A]$ на N столбцов C_i длины N
4) замена столбцов *SubColumns*, т.е. преобразование каждого i-го N-разрядного двоичного набора C_i с использованием соответствующего узла замены S_{i+N}: $C_i = S_{i+N}[C_i]$;
5) результатом замены является $A = SubColumns[A]$.

В частном случае, когда используется только одна таблица замен, т.е. $S_i = S$, получаем следующий алгоритм:

1) разбиение входного массива A на N строк R_i длины N;

2) замена строк *SubRows*, т.е. преобразование каждого i-го N-разрядного двоичного набора R_i:

$$R_i = S[R_i],\ i = 0, 1, ..., (N - 1);$$

3) разбиение получившегося массива $A = SubRows[A]$ на N столбцов C_i длины N;
4) замена столбцов *SubColumns*, т.е. преобразование каждого i-го N-разрядного двоичного набора C_i: $C_i = S[C_i]$;
5) результатом замены является $A = SubColumns[A]$.

Алгоритм функционирования 3D S-блока. Представим входные и выходные блоки данных, а также все промежуточные результаты преобразований в виде кубического массива битов размерностью $N \times N \times N$, где N – разрядность используемых узлов замены. Таким образом, объем ключевой информации, однозначно определяющей логику работы каждого узла замены, равен $N \times 2^N$.

Последовательность выполнения операции замены кубического массива битов $A = SubCube[A]$ (или кратко, $A = S_{cu}[A]$) размерностью $N \times N \times N$ имеет следующий вид:

1) разбиение входного массива A на N слоев L_{xi} размерностью $N \times N$ вдоль оси x, $i = 0, 1, \ldots, (N - 1)$;
2) замена слоев вдоль оси x *SubLayersX*, т.е. выполнение преобразования $L_{xi} = SubSquare[L_{xi}]$ каждого i-го слоя L_{xi} с использованием соответствующих узлов замены;
3) разбиение получившегося массива $A = SubLayersX[A]$ на N слоев L_{yi} размерностью $N \times N$ вдоль оси y;
4) замена слоев вдоль оси y *SubLayersY*, т.е. выполнение преобразования $L_{yi} = SubSquare[L_{yi}]$ каждого i-го слоя L_{yi} с использованием соответствующих узлов замены;
5) разбиение получившегося массива $A = SubLayersY[A]$ на N слоев L_{zi} размерностью $N \times N$ вдоль оси z;
6) замена слоев вдоль оси z *SubLayersZ*, т.е. выполнение преобразования $L_{zi} = SubSquare[L_{zi}]$ каждого i-го слоя L_{zi} с использованием соответствующих узлов замены;
7) результатом замены является $A = SubLayersZ\ [A]$.

Наиболее очевидное назначение предлагаемых алгоритмов – преобразование N^2- и N^3-разрядных блоков данных с использованием таблицы замен размерности $N \times 2^N$.

Трехмерный итеративный криптоалгоритм DOZEN

Рассмотрим $3D$ преобразование данных с архитектурой «Куб», названное авторами DOZEN (dozen – англ. дюжина; название обусловлено тем, что алгоритм имеет двенадцать основных шагов, как будет раскрыто далее).

Основные принципы проекта:

представление входных и выходных блоков данных, всех промежуточных результатов преобразований и раундовых ключей (RoundKeys) K_0, K_1, K_2, K_3 в виде кубического массива байтов $4 \times 4 \times 4$;

определение понятия слоя (Layer) − квадратного массива байтов 4×4;

представление i-го раундового ключа в виде четырех подключей (RoundSubKeys) K_{i0}, K_{i1}, K_{i2}, K_{i3}, $i = 1, 2, 3$, каждый из которых суть квадратный массив байтов 4×4;

трехмерное преобразование блока данных по слоям вдоль осей x, y, z;

включение в состав операции преобразования слоя (T_Layer) четырех шагов – замены байтов (SubBytes), перемешивания строк (MixRows), перемешивания столбцов (MixColumns), сложения (XOR) с раундовым подключом (AddRoundSubKey);

использование при выполнении преобразований MixRow и MixColumn операции, использующейся в криптоалгоритме Rijndael при реализации преобразования MixColumn.

Последовательность преобразования блока данных размером 512 бит ($4 \times 4 \times 4 \times 8$):

разбиение блока данных на слои (Layers) L_{x0}, L_{x1}, L_{x2}, L_{x3} вдоль оси x;

первый раунд: стохастическое преобразование слоев (T_Layer) L_{x0}, L_{x1}, L_{x2}, L_{x3} путем выполнения для каждого

слоя L_{xk} шестнадцати (по числу байтов) операций SubByte, четырех (по числу строк) операций MixRow, четырех (по числу столбцов) операций MixColumn и операции AddRoundSubKey;

разбиение блока данных на слои L_{y0}, L_{y1}, L_{y2}, L_{y3} вдоль оси y;

второй раунд: стохастическое преобразование слоев L_{y0}, L_{y1}, L_{y2}, L_{y3} путем выполнения для каждого слоя L_{yk} шестнадцати (по числу байтов) операций SubByte, четырех (по числу строк) операций MixRow, четырех (по числу столбцов) операций MixColumn и операции AddRoundKey;

разбиение блока данных на слои L_{z0}, L_{z1}, L_{z2}, L_{z3} вдоль оси z;

третий раунд: стохастическое преобразование слоев L_{z0}, L_{z1}, L_{z2}, L_{z3} путем выполнения для каждого слоя L_{zk} шестнадцати (по числу байтов) операций SubByte, четырех (по числу строк) операций MixRow, четырех (по числу столбцов) операций MixColumn и операции AddRoundKey.

Окончательно, получаем следующую последовательность 3D-преобразований:

Шаг 0. Сложение с раундовым ключом K_0 (AddRoundKey K_0).

Первый раунд:

Шаг 1. Преобразование слоя L_{x0} (T_Layer L_{x0}).

Шаг 2. Преобразование слоя L_{x1} (T_Layer L_{x1}).

Шаг 3. Преобразование слоя L_{x2} (T_Layer L_{x2}).

Шаг 4. Преобразование слоя L_{x3} (T_Layer L_{x3}).

Второй раунд:

Шаг 5. Преобразование слоя L_{y0} (T_Layer L_{y0}).

Шаг 6. Преобразование слоя L_{y1} (T_Layer L_{y1}).

Шаг 7. Преобразование слоя L_{y2} (T_Layer L_{y2}).

Шаг 8. Преобразование слоя L_{y3} (T_Layer L_{y3}).

Третий раунд:

Шаг 9. Преобразование слоя L_{z0} (T_Layer L_{z0}).

Шаг 10. Преобразование слоя L_{z1} (T_Layer L_{z1}).

Шаг 11. Преобразование слоя L_{z2} (T_Layer L_{z2}).

Шаг 12. Преобразование слоя L_{z3} (T_Layer L_{z3}).

Особенностями предложенного преобразования являются байт-ориентированная структура и высокая степень параллелизма на уровне инструкций, позволяющая достичь высокой производительности работы алгоритма для гибридных вычислительных систем CPU/GPU.

Михаил Александрович Иванов
Илья Владимирович Чугунков

Криптографические методы защиты информации в компьютерных системах и сетях

Учебное пособие

Под редакцией М.А. Иванова

Редактор Е.Г. Станкевич
Оригинал-макет подготовлен М.А. Ивановым

Подписано в печать 15.11.2011. Формат 60×84 1/16.
Печ. л. 25,0. Уч.-изд. л. 20,0. Тираж 130 экз.
Изд. № 1/37 Заказ № 3

Национальный исследовательский ядерный университет «МИФИ».
115409, Москва, Каширское ш., 31

ООО «Полиграфический комплекс «Курчатовский».
144000, Московская область, г. Электросталь, ул. Красная, 42