КРИПТОГРАФИЧЕСКИЕ АЛГОРИТМЫ

ЗИЯ САРДАР (ZIA SARDAR), специалист по применению, компании Maxim Integrated

В статье рассматривается реализация наиболее распространенных криптографических алгоритмов. Мы начнем с фундаментальной функции XOR, затем обсудим более сложные симметричные и асимметричные алгоритмы. Статья завершится обзором асимметричного алгоритма для обмена общим закрытым ключом.

ФУНКЦИЯ XOR

XOR (исключающее ИЛИ) – критически важная логическая операция, используемая во многих, если не во всех, криптографических алгоритмах. На рисунке 1 показано, как работает эта базовая функция. Ее понимание необходимо перед анализом любого из алгоритмов.

ИСКЛЮЧАЮЩЕЕ ИЛИ – ОСНОВНОЙ ЭЛЕМЕНТ ШИФРОВАНИЯ БЕЗ ПОТЕРИ ДАННЫХ

Благодаря свойствам XOR один из входов может использоваться в качестве ключа для передачи данных на другой вход. Например, если A – одиночный бит ключа шифрования, XOR с битом данных из B «переключает» бит в другое состояние, если A – 1. Повторное применение побитовой операции XOR с ключом и зашифрованным сообщением расшифровывает его.

Рассмотрим пример. Наша цель – зашифровать слово Secret ключом с помощью XOR, а затем расшифровать его с помощью того же ключа и функции XOR. Это делается следующим образом.

1. Выбираем ключ. В его качестве мы выбираем букву k.
2. Преобразуем букву k в двоичный код, используя стандарт кодировки символов ASCII (American Standard Code for Information Interchange). Результат: 01101011.
3. Преобразуем слово Secret в двоичный код. Результат: 01010011 01100101 0110 0011 01110010 01100101 01110100.
4. Применяем операцию XOR к каждой букве слова Secret с ключом k. Получаем зашифрованное сообщение (см. табл. 1).
5. Теперь, чтобы расшифровать зашифрованное сообщение, мы применяем к нему XOR с кодом ключа k. Этот шаг возвращает первоначальное слово Secret (см. табл. 2).

БЕЗОПАСНЫЙ АЛГОРИТМ ХЭШИРОВАНИЯ SHA

Основное назначение функции SHA (Secure Hash Algorithm) состоит в сжатии данных переменного размера в битовую строку фиксированного размера. Эта операция называется хэшированием.

Функции SHA – это семейство алгоритмов хэширования, разрабатываемых в течение длительного времени под наблюдением американского Национального института стандартов и техно-

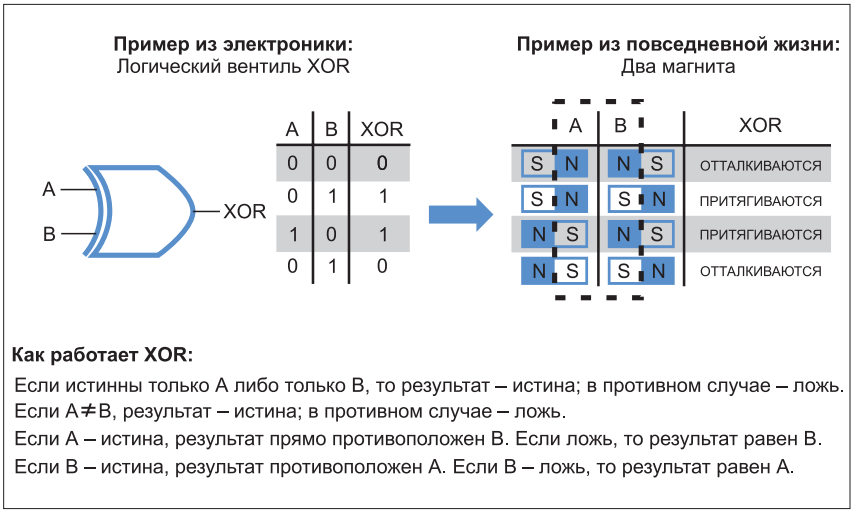

Рис. 1. **Схема работы функции XOR**

Таблица 1. **Зашифрованное сообщение после применения операции XOR с ключом k**

	S	e	c	r	e	t
	01010011	01100101	01100011	01110010	01100101	01110100
Результат применения XOR к «ключу»	01101011	01101011	01101011	01101011	01101011	01101011
Зашифрованное значение	00111000	00001110	00001000	00011001	00001110	00011111

Таблица 2. **Рашифрованное сообщение после применения операции XOR с кодом ключа k**

Зашифрованное значение	00111000	00001110	00001000	00011001	00001110	00011111
Результат применения XOR к «ключу»	01101011	01101011	01101011	01101011	01101011	01101011
Расшифрованное значение	01010011	01100101	01100011	01110010	01100101	01110100
	S	e	c	r	e	t

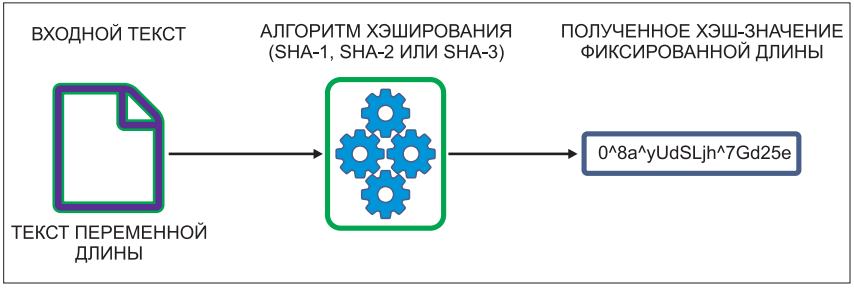

Рис. 2. **Структурная схема базовой концепции безопасной генерации хэша**

логий (NIST). Последним из них является функция SHA-3. На рисунке 2 показана основная концепция безопасной генерации хэша.

Функция SHA имеет следующие характеристики.

1. Переменная длина входного текста.
2. Фиксированная длина выходного значения.
3. Это необратимая функция. Используя алгоритм, описанный на рисунке 2, невозможно использовать полученное хэш-значение для регенерации входного текста, кроме как попробовать каждый возможный входной текст. Для достаточно больших входных данных это становится *вычислительно невозможным*.
4. Если одно и то же входное сообщение подается в функцию SHA, она всегда генерирует один и тот же результирующий хэш.
5. Невозможно сгенерировать одно и то же хэш-значение, используя два разных входных значения. Это называется «*устойчивостью к коллизиям*».
6. Небольшое изменение входного значения, даже одного бита, полностью меняет результирующее хэш-значение. Это называется «*лавинным эффектом*».

Если хэш-функция удовлетворяет всем перечисленным требованиям, она считается сильной хэш-функцией. Некоторые из используемых в настоящее время функций SHA – это SHA-1, SHA-2 и SHA-3.

Теперь давайте рассмотрим, как работают функции SHA. В этой статье мы рассмотрим только SHA-2 и SHA-3. SHA-1 постепенно выводится из применения и не рекомендуется для каких-либо новых устройств.

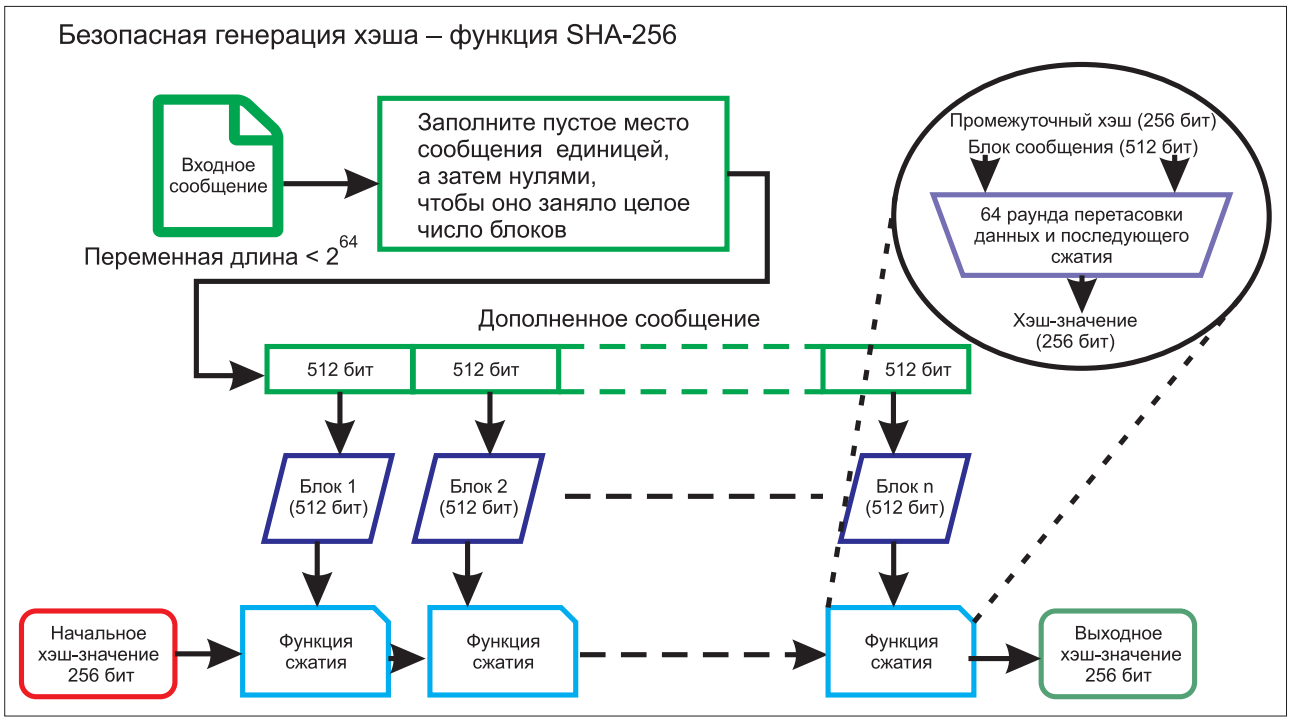

Рис. 3. **Безопасная генерация хэша с помощью функции SHA-256**

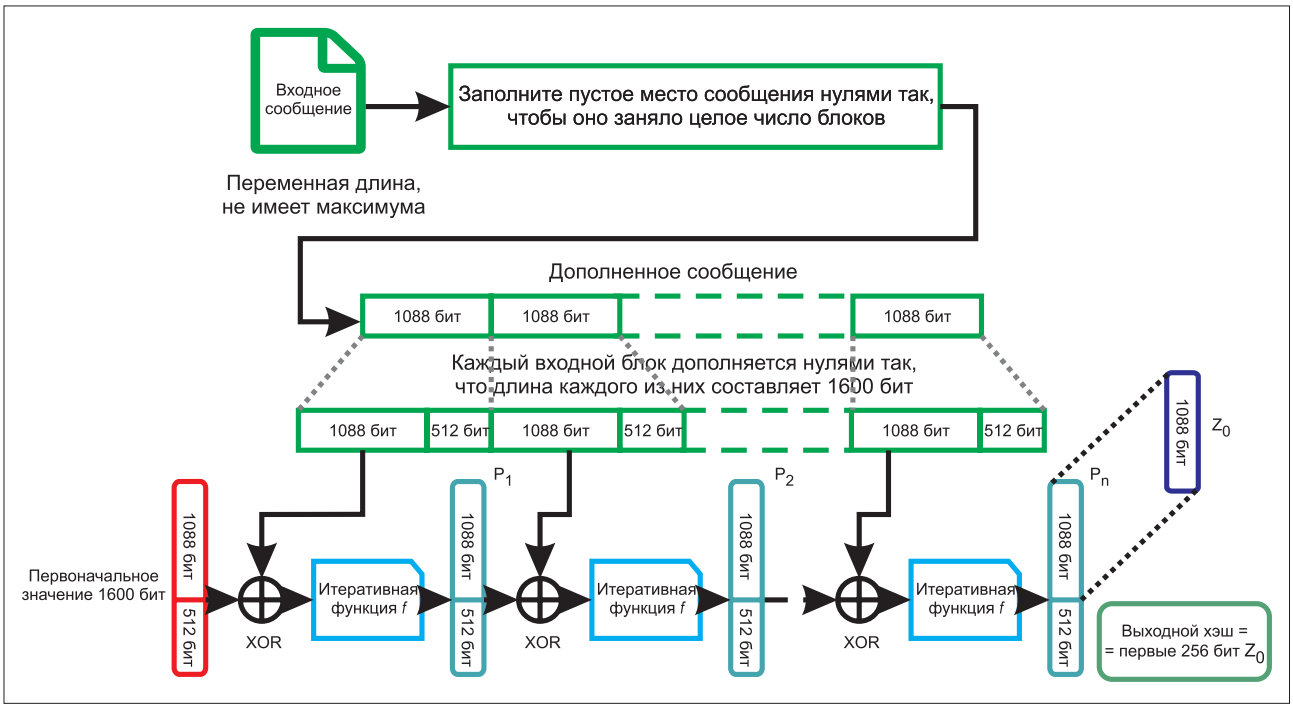

Рис. 4. **Безопасная генерация хэша с помощью функции SHA3-256**

КАК РАБОТАЕТ SHA-2?

Функция SHA-2 имеет четыре основных типа, различающихся длиной выходного значения:

1. SHA-224 – хэш имеет длину 224 бит.
2. SHA-256 – хэш имеет длину 256 бит.
3. SHA-384 – хэш имеет длину 384 бит.
4. SHA-512 – хэш имеет длину 512 бит.

В качестве примера рассмотрим SHA-256. На рисунке 3 показана структурная схема алгоритма SHA-256.

БЕЗОПАСНАЯ ГЕНЕРАЦИЯ ХЭША: ФУНКЦИЯ SHA-256

Сначала входное сообщение дополняется до целого числа 512-бит блоков. Затем первый 512-бит блок подается в функцию сжатия вместе с начальным 256-бит хэш-значением. По сути, функция сжатия перетасовывает сообщение 64 раза, сжимает его до 256 бит и отправляет в следующий блок сжатия или выдает его как окончательный хэш. Таким образом, изменяемое входное сообщение много раз перетасовывается, предотвращая его использование для получения исходного сообщения. После этого генерируется выходной хэш.

КАК РАБОТАЕТ SHA-3?

Функция SHA-3 не имеет предопределенной длины выходного значения. Длины входных и выходных сообщений также не имеют ограничения максимального размера. Для сравнения с SHA-2 мы определим четыре основных типа с разными длинами выходного значения:

1. SHA3–224 – хэш имеет длину 224 бит.
2. SHA3–256 – хэш имеет длину 256 бит.
3. SHA3–384 – хэш имеет длину 384 бит.
4. SHA3–512 – хэш имеет длину 512 бит.

Давайте рассмотрим в качестве примера SHA3–256. SHA-3 использует функцию криптографической губки Keccak. Аналогично губке, на первом шаге входное сообщение «впитывается» или «поглощается». На следующем этапе «выжимается» выходной хэш. На рисунке 4 представлена структурная схема функции SHA3–256.

БЕЗОПАСНАЯ ГЕНЕРАЦИЯ ХЭША: ФУНКЦИЯ SHA3–256

Итеративная функция, показанная на рисунке 4, берет 1600 бит данных и затем проводит их через 24 раунда перестановки, используя определенный алгоритм. После этого она переходит к следующему 1600-бит блоку. Это продолжается до тех пор, пока не завершится фаза поглощения.

По ее завершении последний 1600-бит блок передается для сжатия. В этом случае, поскольку длина выходного хэша SHA3–256 не превышает 1088 бит, фаза сжатия не нуждается в каких-либо итеративных функциях. Первые 256 бит с последнего этапа и есть выходной хэш.

Например, если требуемая длина хэша составляет 2500 бит, то чтобы получить хэш нужной длины, понадобились бы еще три вхождения итеративной функции.

AES (УСОВЕРШЕНСТВОВАННЫЙ СТАНДАРТ ШИФРОВАНИЯ)

Как и у более старых алгоритмов шифрования, например DES (стандарт шифрования данных) и 3DES (тройной стандарт шифрования данных), целью алгоритма AES является обратимое скремблирование и замена входных данных на основе значения входного ключа. Результат называется шифротекстом.

Алгоритм AES был разработан для замены уязвимых для атак алгоритмов DES и 3DES, появившихся ранее. На рисунке 5 описан алгоритм AES.

АЛГОРИТМ AES

Алгоритм AES – это алгоритм шифрования фиксированной ширины. Поэтому входное сообщение сначала дополняется так, чтобы оно заняло целое число 128-бит блоков.

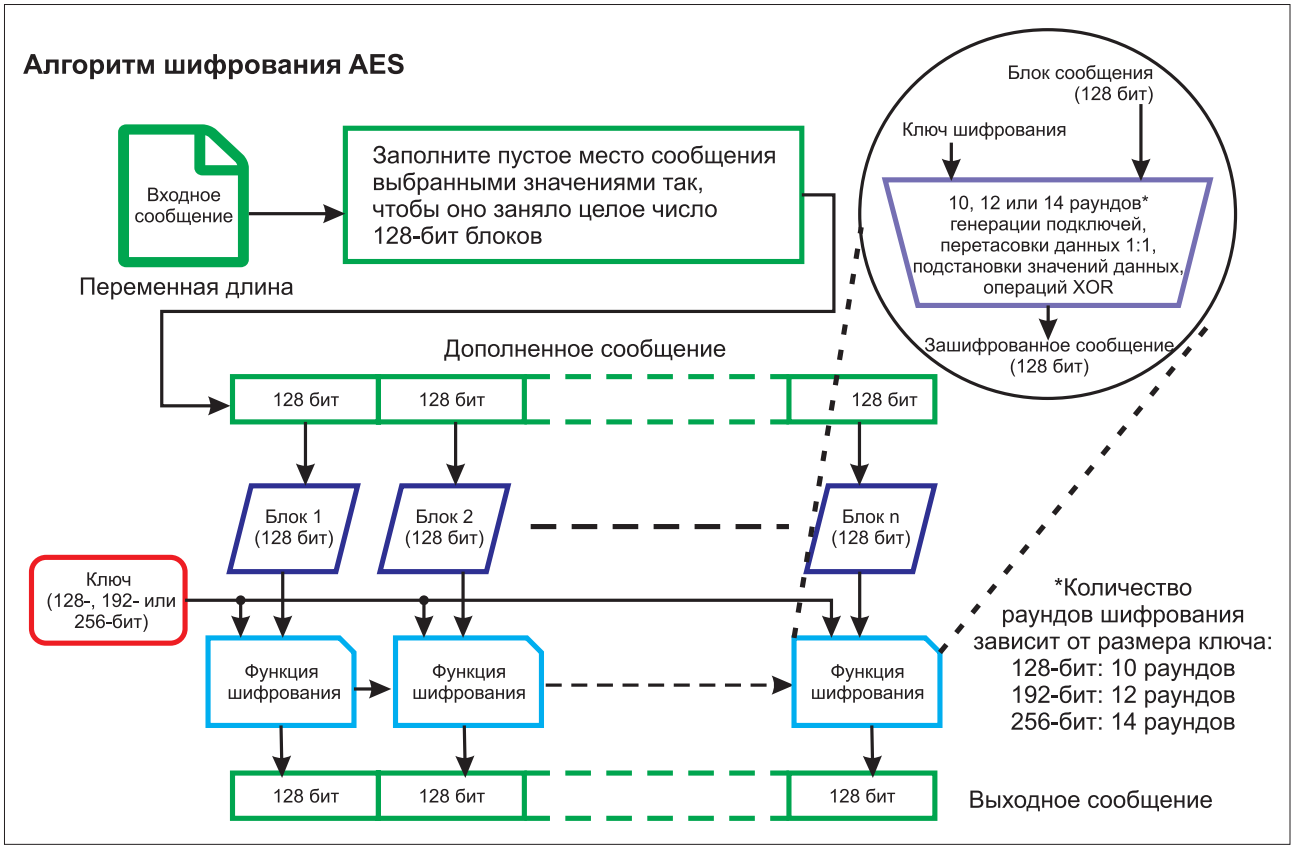

Рис. 5. **Краткий обзор алгоритма AES**

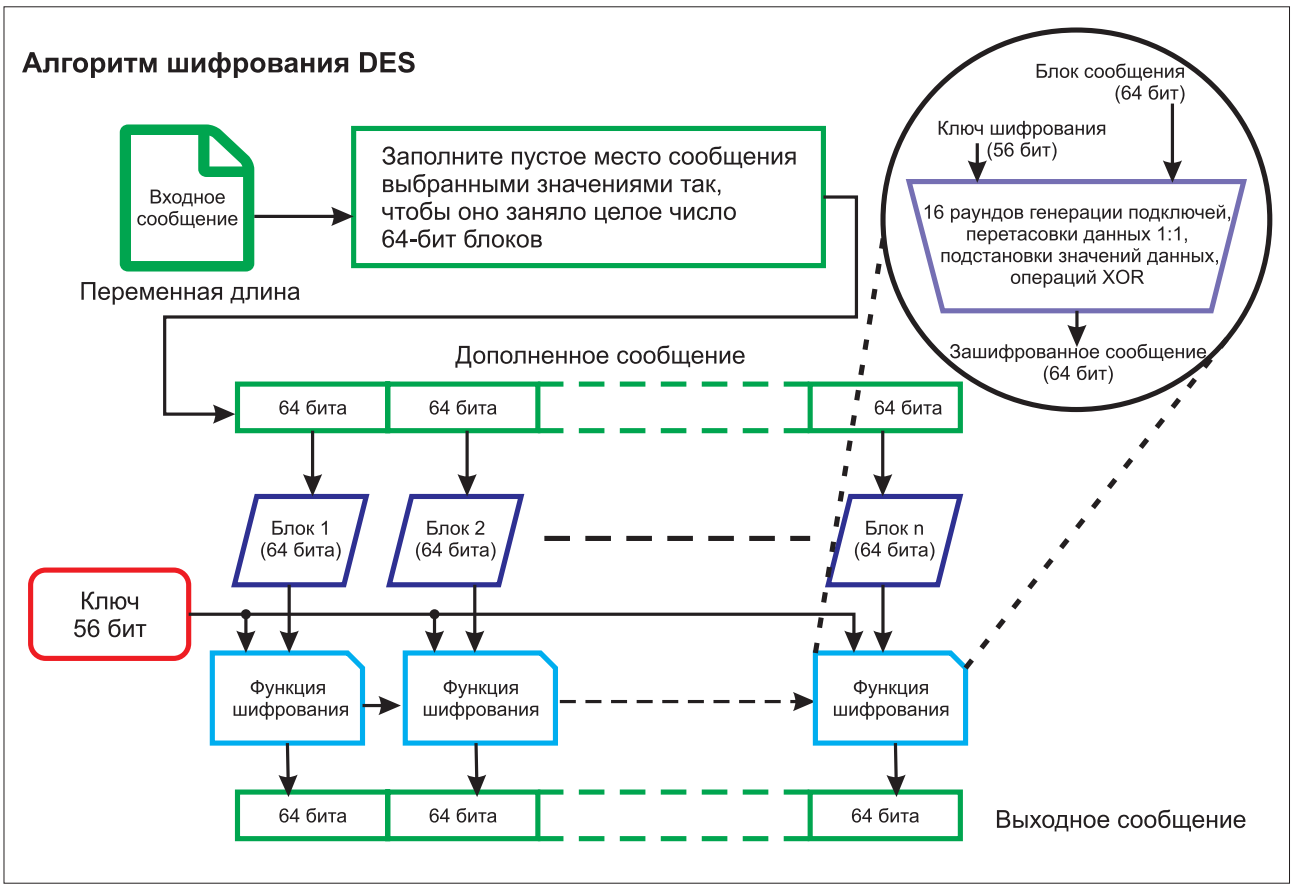

Рис. 6. **Обзор алгоритма DES**

Каждый 128-бит блок подается в алгоритм шифрования вместе с ключом шифрования. В зависимости от количества битов в ключе шифрования алгоритм AES выполняет определенное количество раундов сокрытия битов входного блока.

Сокрытие осуществляется с помощью перетасовки битов данных, взятия частей данных и замены их значениями из таблицы поиска (например, шифра Цезаря), а также выполнения операций XOR для переключения битов с 0 на 1 в соответствии со значениями битов в наборе «раундовых ключей», генерируемых из входного ключа шифрования. Раундовый ключ используется один раз для одного из раундов сокрытия и создается «расширением» части ключа шифрования с помощью копирования битов и вставки этих копий между другими битами.

Функция дешифрования AES выполняет обратные операции функции шифрования, используя для расшифровки исходных данных входного блока тот же ключ шифрования.

3DES (ТРОЙНОЙ СТАНДАРТ ШИФРОВАНИЯ ДАННЫХ)

Основная идея алгоритма 3DES заключается в обратимом скремблировании и замене входных данных на основе входного ключа. Результат называется шифротекстом.

Алгоритм 3DES основан на алгоритме DES, разработанном в 1970-х гг. Когда в 1990-х гг. DES был скомпрометирован, стала очевидна необходимость в более безопасном алгоритме. 3DES оказался скорейшим решением проблем с DES. Чтобы понять алгоритм 3DES, сначала следует рассмотреть описание оригинального DES, показанного на рисунке 6.

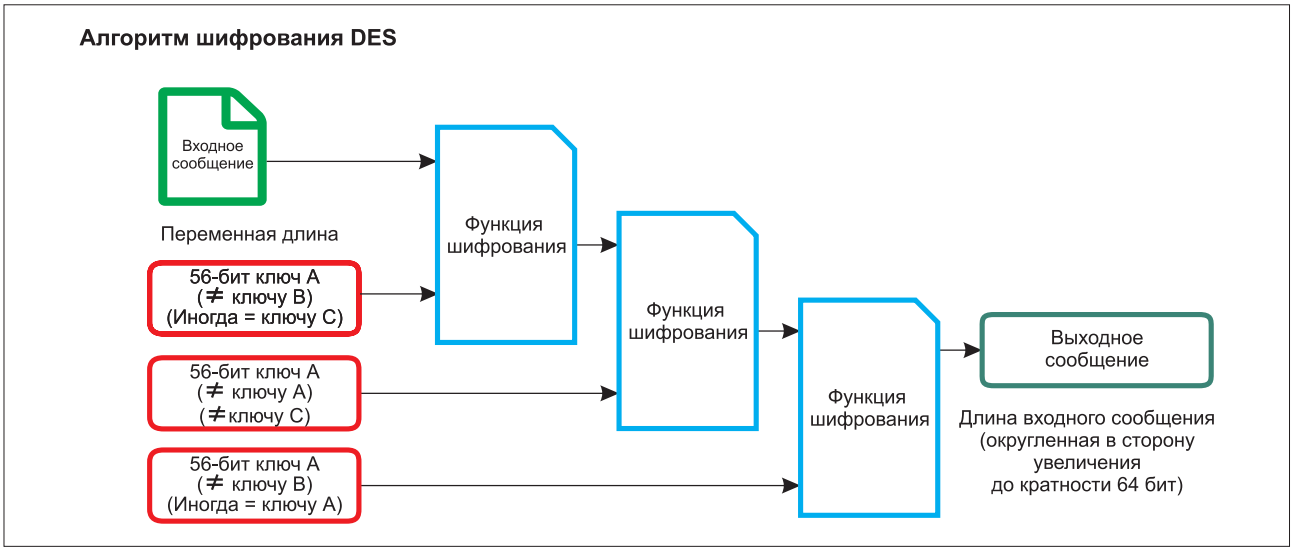

Рис. 7. **Для создания алгоритма 3DES используются три операции DES**

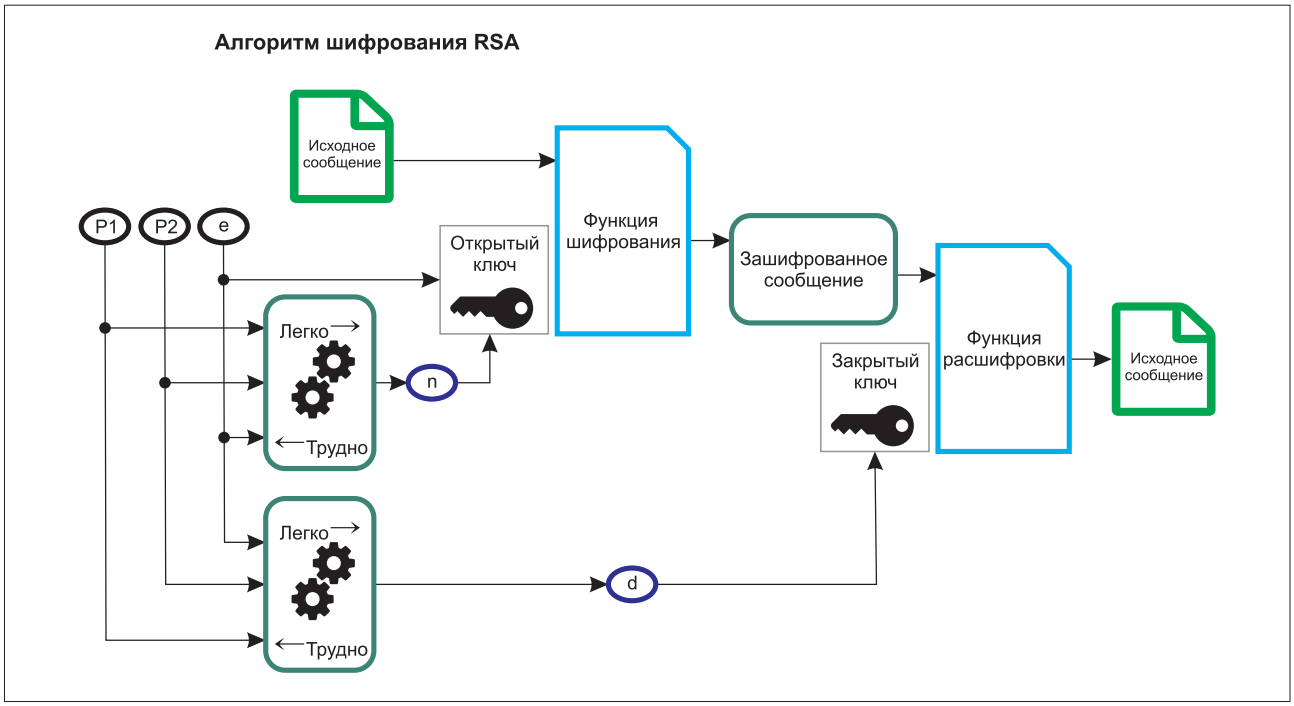

Рис. 8. **Общее представление о шифровании RSA**

АЛГОРИТМ DES

Поскольку алгоритм DES шифрования имеет фиксированную ширину, входное сообщение сначала дополняется, чтобы оно заняло целое число 64-бит блоков.

Каждый 64-бит блок подается в алгоритм шифрования вместе с 56-бит ключом шифрования (большинство версий алгоритма принимают 64-бит ключ, но 8 бит игнорируются). Функция шифрования использует входной ключ для генерации 16 «подключей», каждый из которых используется для 16 раундов сокрытия битов входного блока. Это сокрытие осуществляется с помощью перетасовки битов данных, взятия частей данных и замены их значениями из таблицы поиска (например, шифра Цезаря), а также выполнения операций XOR для переключения битов с 0 на 1 в соответствии со значениями битов в подключах.

Функция дешифрования DES выполняет обратные операции функции шифрования, используя для расшифровки исходных данных входного блока тот же ключ шифрования.

Посмотрим, как работает 3DES.

ПРЕВРАЩЕНИЕ АЛГОРИТМА DES В 3DES

После открытия уязвимости DES для атак методом «грубой силы» (циклического перебора всех возможных значений ключа до тех пор, пока не обнаружатся исходные блоки сообщений), был разработан простой метод эффективного увеличения размера ключа шифрования. На рисунке 7 представлено решение 3DES.

Алгоритм 3DES – это в прямом смысле три операции DES. Первая и последняя из них являются операциями шифрования, в то время как средняя представляет собой операцию дешифрования. Заметим, что «шифрование» и «дешифрование» – просто названия, присвоенные операциям скремблирования, которые являются обратными друг другу.

Для каждой операции DES, выполняемой в 3DES, используется соответствующий ключ. Часто для первой и третьей операций применяется один и тот же ключ. Использование одного и того же ключа для первой и третьей операций и другого ключа для средней операции эффективно удваивает общую длину ключа. Это делает атаку «грубой силы» намного сложнее и устраняет уязвимости алгоритма DES.

СИСТЕМА ШИФРОВАНИЯ С ОТКРЫТЫМ КЛЮЧОМ RSA

Система RSA, названная в честь своих создателей – Рона Ривеста (Ron Rivest), Ади Шамира (Adi Shamir) и Леонарда Адлемана (Leonard Adleman), является одной из первых асимметричных систем шифрования/дешифрования с открытым ключом. Этот алгоритм использует свойства арифметических операций над абсолютными значениями простых чисел для генерации открытого ключа для шифрования и закрытого ключа для дешифрования. Операции шифрования и дешифрования основаны на этом же принципе. Общее представление об RSA дает рисунок 8.

Операция генерации ключей и шифрования/дешифрования известна как односторонняя функция, или «лазейка». Это математические операции,

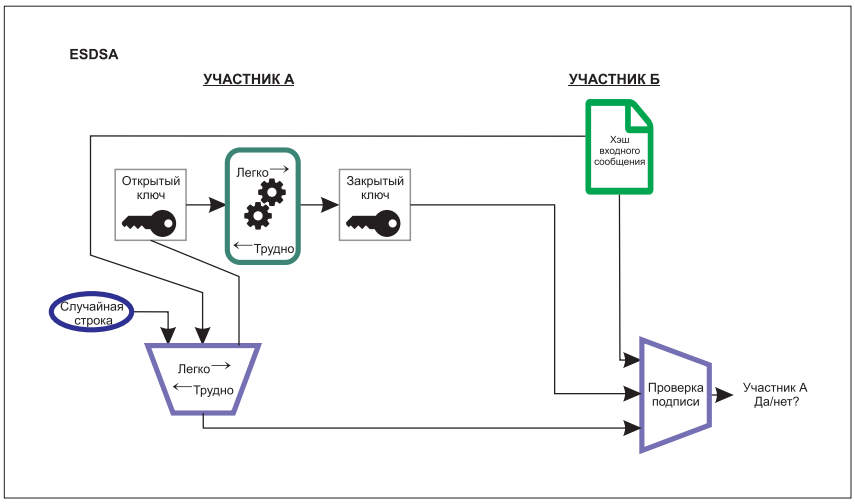

Рис. 9. **ECDSA позволяет проверять цифровые подписи**

которые относительно просто вычислить в одном направлении, но трудно в другом. Например, легко выполнить умножение на 2, но труднее вычислить квадратный корень из X.

В случае RSA два больших простых числа умножаются для создания части открытого и закрытого ключей. Умножение – простая операция; разложение на множители для нахождения секретных простых чисел – трудная.

Кроме того, гораздо проще зашифровать сообщение с помощью открытого ключа, чем пытаться действовать в обратном направлении, чтобы узнать сообщение, не имея закрытого ключа. Однако с помощью закрытого ключа можно легко расшифровать сообщение, и потому он никогда не должен стать доступным посторонним лицам. Закрытый ключ можно рассматривать как лазейку, открывающую кратчайший путь для обхода сложного лабиринта попыток взлома зашифрованного сообщения.

Безопасность RSA зависит от больших простых чисел и сложных операций. Даже легкий путь с его функцией «лазейки» с большими ключами для очень многих вычислительных систем является затруднительным. Поэтому RSA часто используется в качестве средства передачи общих ключей шифрования, которые можно использовать в более быстрых симметричных алгоритмах, например DES, 3DES и AES для отдельных транзакций.

АЛГОРИТМ ECDSA

Алгоритм цифровой подписи на основе эллиптических кривых (ECDSA) позволяет участнику обмена данными доказать подлинность, генерируя цифровую подпись для входного сообщения на основе скрытой части информации, известной как закрытый ключ. Этот ключ необходим для создания открытого ключа, который применяется другими участниками для проверки подлинности участника.

Цифровые подписи генерируются с помощью входного сообщения, закрытого ключа и случайного числа. Затем открытый ключ можно использовать для проверки того, что владелец подписи (или участник) владеет соответствующим закрытым ключом и потому является подлинным. Эту концепцию иллюстрирует рисунок 9.

Алгоритм цифровой подписи был впервые введен с модульной арифметикой, которая зависит от больших простых чисел и вычислений, требующих большого использования вычислительных мощностей. Криптография с эллиптическими кривыми использует математические свойства эллиптических функций для упрощения математических вычислений без ущерба для безопасности.

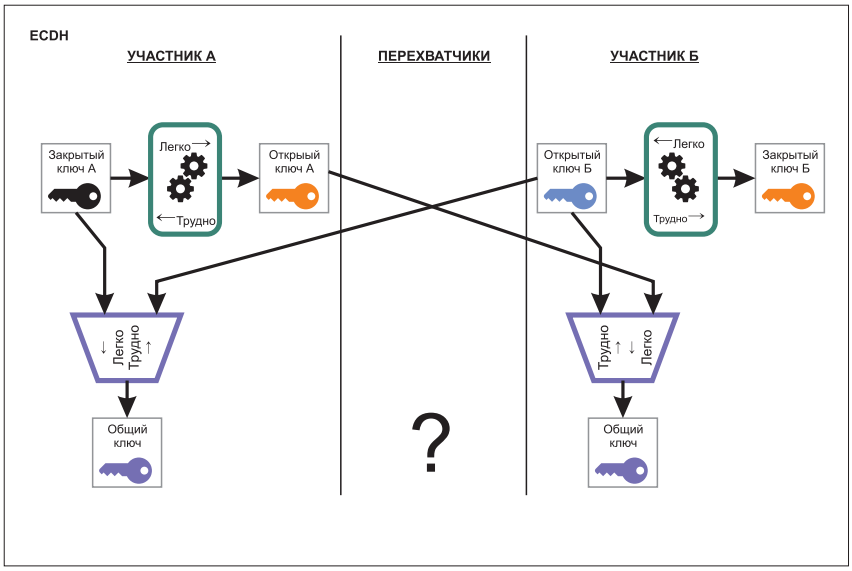

Рис. 10. **Протокол обмена ключами ECDH позволяет двум сторонам установить общий ключ для передачи данных**

Операции генерации и подписания ключей известны как односторонняя функция, или «лазейка». Как и операции RSA, эти вычисления эллиптической кривой относительно просты для вычисления в одном направлении, но для вычисления в другом направлении трудны. Закрытый ключ можно рассматривать как лазейку, открывающую кратчайший путь для обхода сложного лабиринта попыток для взлома генерации и подписания ключей.

ECDSA позволяет одному участнику подписывать сообщения, полученные от любого участника. Однако, чтобы доказать подлинность с помощью ECDSA, подписавший не должен заранее знать о том, что сообщение будет подписано. Это отсутствие контроля над сообщением позволяет другому участнику коммуникации «бросить вызов» подписавшему новой информацией, чтобы доказать владение закрытым ключом.

ПРОТОКОЛ ОБМЕНА КЛЮЧАМИ ECDH

Протокол обмена ключами Диффи-Хеллмана на эллиптических кривых (ECDH) позволяет двум участникам определить общий ключ для обмена данными; существует только одна часть скрытой информации, называемая закрытым ключом. Без этого ключа одной из вовлеченных сторон перехватчик не сможет легко определить общий ключ. Однако этот алгоритм позволяет объединить закрытый ключ одной стороны и открытый ключ другой стороны для получения результирующего ключа, который является одинаковым для обеих сторон. Эта концепция иллюстрируется рисунком 10.

ОБМЕН КЛЮЧАМИ ECDH

Обмен ключами Диффи-Хеллмана был впервые введен с модульной арифметикой, которая зависит от больших простых чисел и вычислений, требующих большой вычислительной мощности. Криптография с эллиптическими кривыми использует математические свойства эллиптических функций для упрощения математических вычислений без ущерба для безопасности.

Как и ECSDA, операции генерации и комбинации ключей известны как односторонняя функция, или «лазейка». Вычисления эллиптической кривой относительно просты для вычисления в одном направлении, но для вычисления в другом направлении требуют больших затрат. Закрытый ключ можно рассматривать как лазейку, открывающую кратчайший путь для обхода сложного лабиринта попыток для взлома генерации или комбинации ключей.

Алгоритм ECDH позволяет двум сторонам вместе определить ключ, но он не гарантирует, что любой из сторон можно доверять – для этого требуются дополнительные уровни аутентификации. Если открытый ключ получает сертификат, например подпись ECDSA, вычисленную с помощью закрытого ключа от доверенного владельца, то открытый ключ определяется путем проверки подлинности сертификата этого ключа с помощью открытого ключа доверенного владельца, выдавшего сертификат.

Используя открытые ключи с сертификатами от доверенного центра, участники ECDH могут быть уверены, что их коллега является подлинным участником.