A screenshot of a computer

Description automatically generated

**1. Một số thuật ngữ sử dụng**

* Encipher - Bộ mã hóa
* Decipher - Bộ giải mã
* Plaintext - Bản rõ là dữ liệu gốc chưa được mã hóa
* Ciphertext - Bản mã là dữ liệu đã dược mã hóa
* Key - khóa mã là một giá trị được sử dụng để mã hóa và giải mã
* Round key - khóa vòng là những giá trị trung gian được tạo ra từ khóa mã trong suốt quá trình mã hóa và giải mã

**2. Thuật toán mã hóa dữ liệu DES – Encipher**

**2.1 Lưu đồ thuật toán mã hóa**

Thuật toán DES được sử dụng để mã hóa và giải mã các block (khối) dữ liệu 64 bit dựa trên một key (khóa mã) 64 bit. Chú ý, các block được đánh số thứ tự bit từ trái sang phải và bắt đầu từ 1, bit đầu tiên bên trái là bit số 1 và bit cuối cùng bên phải là bit số 64. Quá trình giải mã và mã hóa sử dụng cùng một key nhưng thứ tự phân phối các giá trị các bit key của quá trình giải mã ngược với quá trình mã hóa.

Một block dữ liệu sẽ được hoán vị khởi tạo (Initial Permutation) IP trước khi thực hiện tính toán mã hóa với key. Cuối cùng, kết quả tính toán với key sẽ được hoán vị lần nữa để tạo ra , đây là hoán vị đảo của hoán vị khởi tạo gọi là (Inverse Initial Permutation) IP-1. Việc tính toán dựa trên key được định nghĩa đơn giản trong một hàm f, gọi là hàm mã hóa, và một hàm KS, gọi là hàm phân phối key (key schedule). Hàm KS là hàm tạo ra các khóa vòng (round key) cho các lần lặp mã hóa. Có tất cả 16 khóa vòng từ K1 đến K16.

\

1. Симметричные блочные и потоковые шифры.
2. Синхронные и асинхронные потоковые шифры.
3. Сферы применения потоковых шифров.
4. Одноразовый шифр-блокнот
5. Регистры сдвига с линейной обратной связью
6. GSM Протокол A5/2
7. Необходимость вектора инициализации в потоковых шифрах.
8. Шифр Trivium.
9. Шифр Rabbit
10. Свойства конфузии и диффузии в блочных шифрах.
11. Сеть Фейстеля
12. Алгоритм DES. Процедуры шифрования/дешифрования. Расширение ключа.
13. Техника отбеливания ключа
14. 3DES
15. ГОСТ 28147 или блочный шифр с длиной блока 64 бита из ГОСТ 34.12.
16. Режимы шифрования ECB, CBC, CFB, OFB, CTR, GCM. Устойчивость к атакам вставки и удаления блоков. Лавинный эффект. Возможность параллельной обработки блоков.
17. Код аутентификации сообщения
18. Схемы хэширования на основе блочных симметричных криптосистем.
19. **Симметричные блочные и потоковые шифры.**

Симметричные блочные и потоковые шифры - это два основных типа симметричного шифрования, которые используются для защиты конфиденциальности данных.

1. Симметричные блочные шифры:  
   Симметричные блочные шифры преобразуют данные блоками фиксированного размера. Они принимают на вход блок данных и ключ шифрования, и возвращают зашифрованный блок данных того же размера. Примеры блочных шифров включают DES (Data Encryption Standard), AES (Advanced Encryption Standard) и Blowfish.

Основные характеристики симметричных блочных шифров:

* Фиксированный размер блока данных.
* Один и тот же ключ используется для шифрования и расшифрования.
* Высокая степень безопасности при использовании достаточно длинных ключей.

1. Симметричные потоковые шифры:  
   Симметричные потоковые шифры шифруют данные побитово или по байтам. Они генерируют последовательность псевдослучайных битов, называемых потоком ключей, который затем комбинируется с исходными данными для получения зашифрованных данных. Примеры потоковых шифров включают RC4 и Salsa20.

Основные характеристики симметричных потоковых шифров:

* Последовательное шифрование данных.
* Использование потока ключей для комбинирования с исходными данными.
* Высокая скорость работы и возможность шифрования и передачи данных в реальном времени.

Оба типа шифрования имеют свои преимущества и недостатки, и выбор между ними зависит от конкретных требований безопасности и контекста применения.

1. Синхронные и асинхронные потоковые шифры.

Синхронные и асинхронные потоковые шифры - это два подтипа потоковых шифров, которые отличаются способом генерации потока ключей.

1. Синхронные потоковые шифры:  
   Синхронные потоковые шифры генерируют поток ключей, который зависит только от ключа шифрования. Этот поток ключей затем комбинируется с исходными данными для получения зашифрованных данных. Генерация потока ключей осуществляется независимо от входных данных. Примером синхронного потокового шифра является RC4.

Основные характеристики синхронных потоковых шифров:

* Генерация потока ключей зависит только от ключа шифрования.
* Использование независимых от входных данных алгоритмов генерации ключевого потока.
* Простота реализации и высокая скорость работы.

1. Асинхронные потоковые шифры:  
   Асинхронные потоковые шифры генерируют поток ключей, который зависит как от ключа шифрования, так и от входных данных. Генерация потока ключей связана с самими данными, которые шифруются. Примером асинхронного потокового шифра является шифр RSA.

Основные характеристики асинхронных потоковых шифров:

* Генерация потока ключей зависит как от ключа шифрования, так и от входных данных.
* Использование зависимых от входных данных алгоритмов генерации ключевого потока.
* Высокий уровень безопасности и возможность эффективного использования для защиты данных.

Выбор между синхронными и асинхронными потоковыми шифрами зависит от требований безопасности, характеристик применяемого алгоритма и контекста использования.

1. Сферы применения потоковых шифров.

Потоковые шифры находят применение в различных сферах, где требуется защита конфиденциальности данных и обеспечение безопасной передачи информации. Некоторые сферы применения потоковых шифров включают:

1. Компьютерные сети и интернет: Потоковые шифры широко используются для защиты передачи данных в компьютерных сетях, включая передачу через интернет. Они обеспечивают конфиденциальность и целостность данных, а также защиту от несанкционированного доступа.
2. Беспроводные связи: В беспроводных коммуникационных системах, таких как Wi-Fi и мобильные сети, потоковые шифры применяются для обеспечения безопасности передачи данных между устройствами. Они защищают информацию от перехвата и несанкционированного доступа.
3. Криптография голоса и видео: Потоковые шифры используются для защиты голосовых и видео данных, передаваемых по сети. Они обеспечивают конфиденциальность разговоров и видеоконференций, а также защищают от возможности прослушивания или просмотра неавторизованными лицами.
4. Хранение данных: Потоковые шифры могут быть использованы для защиты данных, хранящихся на различных носителях, таких как жесткие диски, флеш-накопители или облачные хранилища. Это помогает предотвратить несанкционированный доступ к информации, если физическое устройство попадает в чужие руки.
5. Криптографические протоколы: Потоковые шифры могут быть включены в различные криптографические протоколы, такие как SSL/TLS для защищенного соединения веб-сайтов или протоколы VPN (виртуальной частной сети) для безопасного удаленного доступа к сетям.

В целом, потоковые шифры находят применение во множестве областей, где требуется защита данных в режиме реального времени или передача данных через небезопасные каналы связи.

1. Одноразовый шифр-блокнот

Одноразовый шифр-блокнот (One-Time Pad, OTP) - это метод шифрования, который использует случайно сгенерированные и одноразовые ключи, имеющие такую же длину, как и сами данные. Он является одним из наиболее безопасных методов шифрования, при условии правильного использования.

Принцип работы одноразового шифра-блокнота следующий:

1. Генерация ключей: Для каждого блока данных генерируется случайный ключ такой же длины. Ключи могут состоять из случайных битов или случайных символов.
2. Шифрование: Каждый блок данных комбинируется с соответствующим ключом путем побитового или символьного исключающего ИЛИ (XOR). Результатом является зашифрованный блок данных.
3. Дешифрование: Зашифрованный блок данных комбинируется с использованным ключом снова путем операции XOR. Результатом является исходный блок данных.

Основные характеристики одноразового шифра-блокнота:

* Абсолютная безусловная безопасность: При правильной реализации и использовании одноразовый шифр-блокнот обеспечивает абсолютную безусловную безопасность. Это означает, что при условии использования ключей один раз и их генерации с истинной случайностью, расшифровка данных без знания ключа становится практически невозможной.
* Необходимость ключей одноразового использования: Каждый ключ должен быть использован только один раз. Если ключи повторно используются, безопасность шифра нарушается, и данные могут быть подвержены взлому.
* Сложность генерации и распределения ключей: Генерация ключей должна быть выполнена с истинной случайностью, и ключи должны быть переданы получателю безопасным способом.
* Размер ключей и данных: Длина ключей должна быть такой же, как и длина данных, которые требуется зашифровать. Это означает, что для больших объемов данных требуются большие ключи, что может быть затратно в плане ресурсов.

Хотя одноразовый шифр-блокнот обеспечивает высокий уровень безопасности, его практическое использование ограничено из-за необходимости генерации и распределения больших объемов случайных ключей и ограничений на их одноразовое использование. Он находит применение в некоторых сферах, требующих высокой степени безопасности, таких как государственная криптография или военные коммуникации.

1. **Регистры сдвига с линейной обратной связью**

Регистры сдвига с линейной обратной связью (Linear Feedback Shift Registers, LFSR) - это тип регистров сдвига, который использует линейную обратную связь для генерации псевдослучайной последовательности битов. Они широко применяются в различных областях, включая криптографию, генерацию случайных чисел и кодирование.

Принцип работы регистров сдвига с линейной обратной связью следующий:

1. Структура регистра: LFSR состоит из набора битов, которые хранятся в регистре. Регистр может быть реализован в виде сдвигового регистра, где биты сдвигаются вправо при каждом такте.
2. Обратная связь: Некоторые биты регистра соединяются с помощью операции XOR, обратной связью, которая определяет новое значение крайнего правого бита.
3. Генерация псевдослучайной последовательности: При каждом такте регистр сдвигается вправо, при этом новое значение крайнего правого бита вычисляется с использованием обратной связи и текущих значений других битов регистра. Полученная последовательность битов называется псевдослучайной последовательностью.

Основные характеристики регистров сдвига с линейной обратной связью:

* Длина регистра: Длина регистра определяет период генерируемой псевдослучайной последовательности. Максимальный период достигается, когда регистр имеет максимальную длину.
* Обратная связь: Обратная связь определяет линейную комбинацию битов регистра, которая используется для определения нового значения крайнего правого бита при каждом такте. Обратная связь должна быть выбрана таким образом, чтобы обеспечить равномерное распределение псевдослучайной последовательности.
* Использование в криптографии: LFSR может использоваться в различных криптографических алгоритмах, включая генерацию ключей и шифрование. Однако, для достижения высокой безопасности, требуется более сложные конструкции, такие как комбинирование нескольких LFSR или применение нелинейных операций.
* Применение в кодировании: LFSR также используются в кодировании и декодировании данных. Они могут быть использованы для обнаружения ошибок и исправления ошибок в передаваемых данных.

LFSR являются простым и эффективным средством генерации псевдослучайных последовательностей, однако они обладают некоторыми ограничениями в области криптографической безопасности и требуют заботливого выбора параметров для достижения желаемых свойств.

1. GSM Протокол A5/2

Протокол A5/2 был одним из алгоритмов шифрования, используемых в старых версиях сети GSM (Global System for Mobile Communications). Он был разработан как слабый алгоритм шифрования для использования в странах, где существовали ограничения на экспорт криптографического программного обеспечения.

Протокол A5/2 использовал 64-битный ключ и был основан на сдвиговом регистре с линейной обратной связью (LFSR). Однако, поскольку алгоритм был слабым и уязвимым к атакам, он был заменен более безопасным алгоритмом шифрования A5/1, который был введен позже.

Уязвимости протокола A5/2 были широко известными, и существовали методы для его взлома. Атаки на A5/2 могли быть осуществлены с использованием слабо защищенного текста или известного потока шифротекста. Это позволяло злоумышленникам предсказывать ключ и дешифровать зашифрованные данные.

В настоящее время протокол A5/2 считается устаревшим и не безопасным для использования. Сети GSM перешли на более современные и безопасные алгоритмы шифрования, такие как A5/1 и более новые версии, такие как A5/3 (также известный как KASUMI), который обеспечивает более высокий уровень безопасности для передачи данных по GSM-сети.

Важно отметить, что информация, предоставленная выше, относится к состоянию на момент моего последнего обновления в сентябре 2021 года. Технологии и протоколы могут изменяться со временем, и рекомендуется обратиться к актуальным источникам для получения последней информации о протоколах GSM.

1. Необходимость вектора инициализации в потоковых шифрах.

Вектор инициализации (Initialization Vector, IV) является неотъемлемой частью потоковых шифров. Вот несколько причин, почему вектор инициализации необходим:

1. Уникальность: Вектор инициализации гарантирует уникальность шифрованного потока данных, даже если один и тот же ключ шифрования используется для разных сообщений. Это важно для предотвращения анализа трафика и обнаружения повторяющихся шаблонов или связей между различными сообщениями.
2. Безопасность: Использование случайного и уникального вектора инициализации усиливает безопасность потокового шифрования. Если вектор инициализации предсказуем или повторно используется, это может привести к серьезным уязвимостям, таким как возможность восстановления ключа или расшифровки данных.
3. Разнообразие: Использование разных векторов инициализации для каждой операции шифрования обеспечивает разнообразие в шифрованном потоке данных. Даже если два сообщения содержат одни и те же данные, шифрованные потоки будут отличаться. Это erschwert атакующим обнаружение шаблонов и получение информации о содержимом сообщений.
4. Синхронизация: Вектор инициализации также служит для синхронизации приемника и передатчика. Получатель должен знать вектор инициализации, чтобы правильно расшифровать данные, которые были зашифрованы с использованием соответствующего вектора инициализации.

Важно отметить, что вектор инициализации должен быть случайным, уникальным и надежным. Уязвимости могут возникнуть, если вектор инициализации предсказуем или не сгенерирован должным образом. Поэтому важно использовать надежные методы генерации случайных векторов инициализации в соответствии с рекомендациями криптографических стандартов.

В итоге, вектор инициализации является важным элементом в потоковых шифрах, обеспечивающим уникальность, безопасность и разнообразие шифрования данных. Его правильное использование является критическим для обеспечения защиты и безопасности информации.

1. Шифр Trivium.

Trivium - это симметричный потоковый шифр, разработанный Кристофером Де Канделером (Christophe De Cannière) и Бартом Презентом (Bart Preneel) в 2005 году. Он был предложен в качестве кандидата на конкурсе eSTREAM, проводимом European Network of Excellence for Cryptology.

Основные характеристики шифра Trivium:

1. Размер ключа и вектора инициализации: Trivium принимает ключ длиной 80 бит и вектор инициализации (IV) длиной 80 бит.
2. Алгоритм: Trivium использует комбинацию трех линейных обратных связей со сдвиговыми регистрами (LFSR) и операций xor для генерации потокового шифротекста. Он состоит из 288 битовых регистров A, B и C, а также из нелинейной функции обратной связи.
3. Инициализация: Перед использованием Trivium необходимо инициализировать регистры A, B и C с использованием ключа и вектора инициализации. Инициализация выполняется путем заполнения регистров определенными значениями на основе ключа и IV.
4. Генерация шифротекста: После инициализации Trivium генерирует потоковый ключ, который может быть использован для шифрования или расшифрования данных. Генерация ключа основана на сдвиге значений в регистрах A, B и C и применении операции xor.
5. Безопасность: Trivium был подвергнут обширному криптографическому анализу, и на данный момент нет известных практических атак, которые могут эффективно взломать шифр. Однако следует отметить, что безопасность Trivium основана на математических предположениях и его безопасность может быть угрожена, если эти предположения будут опровергнуты в будущем.

Trivium является одним из многих потоковых шифров, предназначенных для обеспечения конфиденциальности и защиты данных. При использовании шифра Trivium или любого другого криптографического алгоритма важно следовать рекомендациям по его правильной реализации и использованию для обеспечения максимальной безопасности.

1. Шифр Rabbit

Rabbit (Rapid Encryption/Decryption Routine) — это симметричный потоковый шифр, разработанный Мэттью Д. Грином (Mattew D. Green) и Йеном А. Голдбергом (Ian A. Goldberg) в 2003 году. Он предназначен для обеспечения высокой скорости шифрования и хорошей безопасности.

Основные характеристики шифра Rabbit:

1. Размер ключа и вектора инициализации: Rabbit принимает ключ переменной длины от 128 до 1024 бит и вектор инициализации (IV) длиной 64 бита.
2. Алгоритм: Rabbit основан на комбинации нелинейной функции обратной связи сдвигового регистра (NLFSR) и генератора задержки (DFG). Он использует операции xor и добавления по модулю 2 для генерации потокового шифротекста.
3. Инициализация: Перед использованием Rabbit необходимо инициализировать внутреннее состояние с использованием ключа и вектора инициализации. Инициализация выполняется путем применения функции расширения ключа и генерации начального состояния.
4. Генерация шифротекста: Rabbit генерирует потоковый ключ, который может быть использован для шифрования или расшифрования данных. Генерация ключа основана на операциях xor и добавлении по модулю 2.
5. Безопасность: Rabbit является одним из шифров, рекомендованных для использования в программе eSTREAM, проводимой European Network of Excellence for Cryptology. Он был подвергнут обширному криптографическому анализу, и на данный момент нет известных практических атак, которые могут эффективно взломать шифр.

Rabbit обеспечивает высокую скорость шифрования и может быть использован в различных приложениях, требующих быстрого и безопасного потокового шифрования данных. При использовании Rabbit или любого другого криптографического алгоритма важно следовать рекомендациям по его правильной реализации и использованию для обеспечения максимальной безопасности.

**-----------------------------------------------**

1. Свойства конфузии и диффузии в блочных шифрах.

Конфузия и диффузия являются двумя важными криптографическими свойствами блочных шифров. Они способствуют обеспечению безопасности и стойкости шифра путем усложнения связей между входными данными, ключом и выходными данными. Вот более подробное описание каждого из этих свойств:

1. Конфузия (Confusion): Конфузия относится к созданию сложных и нелинейных связей между ключом и шифротекстом. Цель конфузии - сделать статистические связи между входными и выходными данными шифра нелинейными и сложными для анализа. Это достигается путем использования нелинейных замен (S-блоков) или подобных механизмов, которые затрудняют обратное вычисление ключа из шифротекста.
2. Диффузия (Diffusion): Диффузия относится к распространению входной информации по всему блоку данных шифра. Цель диффузии - распределить входные биты по всем выходным битам шифра, чтобы изменение одного бита входа приводило к изменению множества выходных битов. Диффузия обеспечивает равномерное распределение статистических свойств данных в шифротексте и увеличивает изменчивость выхода шифра относительно изменений входных данных.

Конфузия и диффузия взаимодополняющие свойства и обеспечивают стойкость и безопасность блочных шифров. Конфузия усложняет анализ шифрования путем введения нелинейности и неоднозначности связей между входными и выходными данными, а диффузия распространяет изменения входных данных по всему шифруемому блоку, обеспечивая равномерность и изменчивость шифротекста.

Комбинация конфузии и диффузии позволяет блочным шифрам достигать свойства шифровального оракула, где небольшое изменение входных данных или ключа приводит к значительным изменениям в выходных данных, делая шифр устойчивым к различным атакам, включая атаки перебором и статистические атаки.

Важно отметить, что эти свойства должны быть применены в правильной мере и сбалансированно в дизайне блочных шифров, чтобы обеспечить эффективную защиту и безопасность.

1. Сеть Фейстеля

Сеть Фейстеля (Feistel network) - это структура, используемая в блочных шифрах для выполнения операций шифрования и расшифрования. Она была предложена Гордоном Фейстелем (Horst Feistel) в 1973 году и стала широко применяемой в различных криптографических алгоритмах, таких как DES (Data Encryption Standard).

Основная идея сети Фейстеля заключается в разделении блока данных на две половины и последовательном применении раундовых функций к этим половинам с использованием ключей раунда. Структура сети Фейстеля состоит из следующих шагов:

1. Разделение: Входной блок данных разделяется на две половины одинакового размера. Обозначим эти половины как L0 и R0.
2. Раунды: Процесс выполняется в несколько раундов. Каждый раунд состоит из следующих шагов:
   * Правая половина (R) подается на вход раундовой функции, которая преобразует ее с использованием ключа раунда.
   * Результат преобразования раундовой функции (F) складывается с левой половиной (L) с помощью операции xor.
   * Правая половина (R) становится новой левой половиной (L), а результат операции xor становится новой правой половиной (R).
   * Процесс повторяется для заданного числа раундов.
3. Обратное объединение: После завершения последнего раунда левая и правая половины объединяются в обратном порядке. То есть, финальный блок данных будет иметь структуру RnLn, где n - количество раундов.

Преимущество сети Фейстеля заключается в ее простоте и возможности обеспечения хорошей конфузии и диффузии, важных криптографических свойств. Она также позволяет использовать те же операции и ключи как для шифрования, так и для расшифрования, что делает ее вычислительно эффективной.

Сеть Фейстеля широко применяется в различных блочных шифрах, включая DES, AES (Advanced Encryption Standard) и Blowfish. Однако существуют и другие конструктивные подходы к блочным шифрам, и выбор конкретной структуры зависит от требований безопасности, производительности и других факторов.

1. Алгоритм DES. Процедуры шифрования/дешифрования. Расширение ключа.

DES (Data Encryption Standard) - это симметричный блочный шифр, который был разработан в 1970-х годах и является одним из наиболее широко используемых и известных криптографических алгоритмов. Он использует сеть Фейстеля и ключ длиной 56 бит.

Процедуры шифрования и дешифрования в алгоритме DES следующие:

1. Расширение ключа (Key Expansion):
   * Исходный 56-битный ключ подвергается перестановке и сжатию, чтобы получить 48-битный ключ раунда для каждого из 16 раундов шифрования.
   * В процессе расширения ключа также генерируются раундовые ключи для использования в каждом раунде шифрования.
2. Шифрование:
   * Входной блок данных размером 64 бита разделяется на левую и правую половины (L0 и R0) по 32 бита каждая.
   * Процесс выполняется в 16 раундах, в каждом из которых применяются следующие шаги:
     + Правая половина (R) подается на вход раундовой функции, которая преобразует ее с использованием 48-битного ключа раунда.
     + Результат преобразования раундовой функции складывается с левой половиной (L) с помощью операции xor.
     + Правая половина (R) становится новой левой половиной (L), а результат операции xor становится новой правой половиной (R).
   * После завершения 16 раундов левая и правая половины объединяются в обратном порядке для получения шифрованного блока данных размером 64 бита.
3. Дешифрование:
   * Процедура дешифрования DES идентична процедуре шифрования, но раундовые ключи используются в обратном порядке.
   * То есть, в процессе дешифрования первый раунд использует последний раундовый ключ, второй раунд - предпоследний раундовый ключ и так далее.

Алгоритм DES обеспечивает хорошую безопасность на момент его разработки, но с течением времени выявлены некоторые ограничения. В настоящее время рекомендуется использовать более современные алгоритмы, такие как AES (Advanced Encryption Standard), которые обладают более высокой стойкостью к атакам и поддерживают более длинные ключи.

1. Техника отбеливания ключа

Техника отбеливания ключа (key whitening) - это метод, применяемый в криптографии для усиления безопасности симметричных ключей шифрования. Он основан на комбинировании ключа с псевдослучайными битами (отбеливателем), что обеспечивает дополнительную сложность и стойкость к различным атакам.

Процесс отбеливания ключа включает следующие шаги:

1. Генерация отбеливателя: Создается псевдослучайная последовательность бит, которая называется отбеливателем. Эта последовательность должна быть достаточно длинной и криптографически стойкой.
2. Комбинирование ключа и отбеливателя: Сгенерированный ключ шифрования и отбеливатель объединяются путем применения операции xor (исключающее ИЛИ) для каждого бита. То есть каждый бит ключа xor'ится с соответствующим битом отбеливателя.
3. Использование отбеленного ключа: Полученный отбеленный ключ используется для выполнения операций шифрования или расшифрования данных.

Преимущества использования техники отбеливания ключа включают следующее:

1. Усложнение атак: Комбинирование ключа с отбеливателем усложняет задачу злоумышленника, так как для успешной атаки ему потребуется знание как самого ключа, так и отбеливателя.
2. Защита от атак с известным открытым текстом: Если злоумышленник обладает информацией о паре открытый текст-шифротекст, отбеливание ключа делает невозможным использование известного открытого текста для восстановления ключа.
3. Увеличение стойкости: Применение отбеливания ключа увеличивает сложность атаки методами анализа разностей, линейных или дифференциальных криптоанализа.

Техника отбеливания ключа может быть использована в различных криптографических применениях, включая блочные шифры, поточные шифры и хэш-функции, с целью повышения уровня безопасности и стойкости к атакам.

1. 3DES

3DES (Triple Data Encryption Standard), также известный как TDEA (Triple Data Encryption Algorithm), является улучшенной версией оригинального алгоритма DES. 3DES использует блочный шифр DES для выполнения тройного шифрования данных, обеспечивая более высокий уровень безопасности по сравнению с оригинальным DES.

Процесс шифрования с использованием 3DES включает следующие шаги:

1. Генерация ключей: Генерируются три ключа DES, каждый длиной 56 бит. Обозначим эти ключи как K1, K2 и K3.
2. Шифрование:
   * Исходные данные разделяются на блоки фиксированного размера (обычно 64 бита).
   * Каждый блок данных проходит через три этапа шифрования.
   * На первом этапе блок данных шифруется с использованием ключа K1.
   * На втором этапе зашифрованный блок данных расшифровывается с использованием ключа K2.
   * На третьем этапе результат второго этапа шифруется с использованием ключа K3.
   * Итоговый зашифрованный блок данных получается в результате трех этапов шифрования.

Процесс дешифрования 3DES аналогичен процессу шифрования, но ключи K1, K2 и K3 используются в обратном порядке.

3DES обеспечивает более высокую стойкость к атакам по сравнению с оригинальным DES, так как он использует три ключа и выполняет три этапа шифрования. Он также обратно совместим с DES, что означает, что существующие системы, использующие DES, могут легко обновиться до 3DES без необходимости изменения структуры данных или протоколов.

Однако, стандарт 3DES считается устаревшим в настоящее время, поскольку ключи длиной 56 бит считаются недостаточно безопасными. Для современных криптографических задач рекомендуется использовать более современные алгоритмы, такие как AES (Advanced Encryption Standard), который обеспечивает более высокий уровень безопасности и производительности.

1. ГОСТ 28147 или блочный шифр с длиной блока 64 бита из ГОСТ 34.12.

ГОСТ 28147-89 (также известный как Магма) - это блочный шифр, используемый в криптографии и стандарте информационной безопасности России. Он был разработан в СССР и описан в ГОСТ 28147-89. Шифр ГОСТ 28147-89 является частью стандарта ГОСТ 34.12, который также включает в себя алгоритмы блочного шифрования с длиной блока 128 бит (ГОСТ 34.12-2015).

Основные характеристики шифра ГОСТ 28147-89:

* Длина блока: 64 бита.
* Длина ключа: 256 бит (32 байта).
* Режимы работы: шифрование простой замены (ECB), шифрование с обратной связью по шифротексту (CFB), режим сцепления блоков шифротекста (CBC) и другие.

Процесс шифрования с использованием ГОСТ 28147-89 включает следующие шаги:

1. Генерация раундовых ключей: 256-битный ключ разделяется на 8 32-битных раундовых ключей.
2. Разделение исходных данных на блоки: Исходные данные разбиваются на блоки размером 64 бита. Если последний блок данных меньше 64 бит, он дополняется до нужного размера.
3. Раунды шифрования: Каждый блок данных проходит через несколько раундов шифрования, включающих следующие этапы:
   * Блок данных проходит через несколько итераций замены и перемешивания (Substitution-Permutation Network, SPN).
   * В каждом раунде блок данных комбинируется с раундовым ключом с помощью операции xor.
   * Результат операции xor становится входом для следующего раунда.
4. Финальная перестановка: После завершения раундов шифрования выполняется финальная перестановка битов в зашифрованном блоке данных.

Процесс дешифрования ГОСТ 28147-89 аналогичен процессу шифрования, но раундовые ключи используются в обратном порядке.

Помимо режима ECB, ГОСТ 28147-89 поддерживает другие режимы работы, такие как CFB и CBC, которые позволяют шифровать и расшифровывать данные большего объема и обеспечивают внутреннюю обратную связь между блоками данных.

Важно отметить, что ГОСТ 28147-89 является устаревшим стандартом, и в настоящее время рекомендуется использовать более современные алгоритмы, такие как AES, для обеспечения более высокого уровня безопасности и эффективности.

1. Режимы шифрования ECB, CBC, CFB, OFB, CTR, GCM. Устойчивость к атакам вставки и удаления блоков. Лавинный эффект. Возможность параллельной обработки блоков.

**Режимы шифрования ECB, CBC, CFB, OFB, CTR, GCM**

Режимы шифрования ECB, CBC, CFB, OFB, CTR и GCM являются различными способами применения алгоритма блочного шифрования для защиты данных. Каждый из этих режимов имеет свои особенности и применяется в различных ситуациях.

**ECB (Electronic Codebook)** - это самый простой режим шифрования, в котором каждый блок данных шифруется независимо от других блоков. Однако этот режим не обеспечивает конфиденциальность, так как одинаковые блоки данных будут шифроваться в одинаковые шифротексты.

**CBC (Cipher Block Chaining)** - в этом режиме каждый блок данных перед шифрованием комбинируется с предыдущим шифротекстом. Это обеспечивает конфиденциальность и предотвращает возможность атаки вставки и удаления блоков.

**CFB (Cipher Feedback)** - в этом режиме шифрования предыдущий шифротекст используется для генерации ключевого потока, который затем комбинируется с открытым текстом. Это позволяет обрабатывать данные меньшими блоками и обеспечивает защиту от атак вставки и удаления блоков.

**OFB (Output Feedback)** - в этом режиме шифрования предыдущий шифротекст используется для генерации ключевого потока, который затем комбинируется с открытым текстом. Это похоже на режим CFB, но не требует обратной связи с открытым текстом. Он также обеспечивает защиту от атак вставки и удаления блоков.

**CTR (Counter)** - в этом режиме шифрования каждый блок данных комбинируется с уникальным счетчиком, который затем шифруется. Это позволяет параллельную обработку блоков и обеспечивает защиту от атак вставки и удаления блоков.

**GCM (Galois/Counter Mode)** - это режим шифрования, который комбинирует режим CTR с аутентификацией сообщений. Он обеспечивает конфиденциальность, целостность и аутентичность данных.

**Устойчивость к атакам вставки и удаления блоков**

Режимы шифрования CBC, CFB, OFB, CTR и GCM обеспечивают защиту от атак вставки и удаления блоков. Это достигается путем комбинирования блоков данных с предыдущими шифротекстами или счетчиками, что делает невозможным изменение данных без знания ключа шифрования.

**CBC** использует предыдущий шифротекст для комбинирования с текущим блоком данных, что делает невозможным изменение данных без изменения всего последующего шифротекста.

**CFB** и **OFB** используют предыдущий шифротекст для генерации ключевого потока, который затем комбинируется с открытым текстом. Это означает, что изменение блока данных приведет к изменению всего последующего шифротекста.

**CTR** использует уникальный счетчик для комбинирования с каждым блоком данных. Это означает, что изменение блока данных приведет к изменению только этого блока шифротекста.

**GCM** комбинирует режим CTR с аутентификацией сообщений, что обеспечивает защиту от атак вставки и удаления блоков, а также целостность и аутентичность данных.

**Лавинный эффект**

Лавинный эффект - это свойство шифрования, при котором небольшие изменения в открытом тексте приводят к значительным изменениям в шифротексте. Лавинный эффект является желательным свойством шифрования, так как он усложняет анализ шифротекста и делает его более устойчивым к атакам.

В режимах шифрования CBC, CFB, OFB, CTR и GCM обеспечивается лавинный эффект. Это достигается путем комбинирования блоков данных с предыдущими шифротекстами или счетчиками. Даже небольшие изменения в открытом тексте приведут к изменениям во всем последующем шифротексте, что затрудняет анализ и восстановление исходных данных.

**Возможность параллельной обработки блоков**

Режимы шифрования CTR и GCM обеспечивают возможность параллельной обработки блоков данных. В режиме CTR каждый блок данных может быть обработан независимо от других блоков, что позволяет использовать параллельные вычисления для ускорения процесса шифрования или расшифрования. Режим GCM также поддерживает параллельную обработку блоков данных и обеспечивает аутентификацию сообщений.

Остальные режимы шифрования (ECB, CBC, CFB, OFB) не поддерживают прямую параллельную обработку блоков данных, так как каждый блок зависит от предыдущего блока. Однако, в некоторых случаях, можно использовать параллельные вычисления для обработки нескольких блоков одновременно, но это требует дополнительных мер предосторожности для обеспечения правильного порядка обработки блоков.

1. Код аутентификации сообщения

#### Код аутентификации сообщения

Код аутентификации сообщения (Message Authentication Code, MAC) - это код, который используется для аутентификации и целостности сообщения. MAC генерируется с использованием секретного ключа и исходного сообщения, и он может быть проверен с использованием того же ключа для подтверждения целостности сообщения.

MAC обеспечивает аутентификацию сообщения, так как любые изменения в сообщении или ключе приведут к изменению MAC. Он также обеспечивает целостность сообщения, так как получатель может проверить MAC, чтобы убедиться, что сообщение не было изменено в пути.

MAC может быть реализован с использованием различных алгоритмов, таких как HMAC (Hash-based Message Authentication Code) и CBC-MAC (Cipher Block Chaining Message Authentication Code). HMAC использует хэш-функцию в сочетании с ключом для генерации MAC **1**, а CBC-MAC использует блочный шифр в режиме CBC для генерации MAC.

MAC является важным инструментом для обеспечения безопасности данных и защиты от подделки и изменения сообщений. Он широко используется в различных протоколах и системах, включая компьютерную безопасность, сетевую безопасность и криптографию.

1. Схемы хэширования на основе блочных симметричных криптосистем.

#### 18. Схемы хэширования на основе блочных симметричных криптосистем

Схемы хэширования на основе блочных симметричных криптосистем - это методы, которые используют блочные шифры для создания хэш-функций. Хэш-функции принимают на вход произвольное сообщение и генерируют фиксированную длину хэш-значения, которое является уникальным представлением исходного сообщения.

Существует несколько схем хэширования на основе блочных симметричных криптосистем, включая:

1. **MD5** (Message Digest Algorithm 5) - это одна из самых известных схем хэширования. Она использует 128-битный блочный шифр и генерирует 128-битное хэш-значение. Однако, MD5 считается устаревшей и небезопасной схемой из-за своих уязвимостей к коллизиям.
2. **SHA-1** (Secure Hash Algorithm 1) - это другая известная схема хэширования. Она использует 160-битный блочный шифр и генерирует 160-битное хэш-значение. Однако, SHA-1 также считается устаревшей и небезопасной схемой из-за своих уязвимостей к коллизиям.
3. **SHA-256** (Secure Hash Algorithm 256) - это более современная и безопасная схема хэширования. Она использует 256-битный блочный шифр и генерирует 256-битное хэш-значение. SHA-256 является частью семейства SHA-2, которое также включает SHA-224, SHA-384 и SHA-512.
4. **HMAC** (Hash-based Message Authentication Code) - это схема хэширования, которая комбинирует хэш-функцию с секретным ключом для создания MAC (Message Authentication Code). HMAC может использовать различные хэш-функции, включая MD5, SHA-1 и SHA-256.

Схемы хэширования на основе блочных симметричных криптосистем широко используются для обеспечения целостности данных, аутентификации и защиты от подделки. Они применяются в различных областях, включая компьютерную безопасность, сетевую безопасность и криптографию.