Лабораторная работа № 5

Тема работы: Криптографические методы защиты информации.

Цель работы: Изучение основных криптографических алгоритмов (изучения алгоритмов симметричного шифрования и хеширования).

Теоретическая часть:

1. Симметричные алгоритмы шифрования

Симметричное шифрование — это способ шифрования данных, при котором один и тот же ключ используется и для кодирования, и для восстановления информации. До 1970-х годов, когда появились первые асимметричные шифры, оно было единственным криптографическим методом.

Принцип работы симметричных алгоритмов

В целом симметричным считается любой шифр, использующий один и тот же секретный ключ для шифрования и расшифровки. Например, если алгоритм предполагает замену букв числами, то и у отправителя сообщения, и у его получателя должна быть одна и та же таблица соответствия букв и чисел: первый с ее помощью шифрует сообщения, а второй — расшифровывает.

Однако такие простейшие шифры легко взломать — например, зная частотность разных букв в языке, можно соотносить самые часто встречающиеся буквы с самыми многочисленными числами или символами в коде, пока не удастся получить осмысленные слова. С использованием компьютерных технологий такая задача стала занимать настолько мало времени, что использование подобных алгоритмов утратило всякий смысл.

Поэтому современные симметричные алгоритмы считаются надежными, если отвечают следующим требованиям:

- Выходные данные не должны содержать статистических паттернов исходных данных (как в примере выше: наиболее частотные символы осмысленного текста не должны соответствовать наиболее частотным символам шифра).
- Шифр должен быть нелинейным (то есть в шифрованных данных не должно быть закономерностей, которые можно отследить, имея на руках несколько открытых текстов и шифров к ним).

Большинство актуальных симметричных шифров для достижения результатов, соответствующих этим требованиям, используют комбинацию операций подстановки

(замена фрагментов исходного сообщения, например букв, на другие данные, например цифры, по определенному правилу или с помощью таблицы соответствий) и перестановки (перемешивание частей исходного сообщения по определенному правилу), поочередно повторяя их. Один круг шифрования, состоящий из этих операций, называется *раундом*.

Виды алгоритмов симметричного шифрования

В зависимости от принципа работы алгоритмы симметричного шифрования делятся на два типа:

- блочные;
- потоковые.

Блочные алгоритмы шифруют данные блоками фиксированной длины (64, 128 или другое количество бит в зависимости от алгоритма). Если все сообщение или его финальная часть меньше размера блока, система дополняет его предусмотренными алгоритмом символами, которые так и называются дополнением.

К актуальным блочным алгоритмам относятся:

- DES
- Triple DES (3DES)
- RSA RC2
- AES
- ГОСТ 28147-89
- RSA RC5
- Blowfish
- Twofish

Потоковое шифрование данных предполагает обработку каждого бита информации с использованием *гаммирования*, то есть изменения этого бита с помощью соответствующего ему бита псевдослучайной секретной последовательности чисел, которая формируется на основе ключа и имеет ту же длину, что и шифруемое сообщение. Как правило, биты исходных данных сравниваются с битами секретной последовательности с помощью логической операции XOR (исключающее ИЛИ, на выходе дающее 0, если значения битов совпадают, и 1, если они различаются).

Потоковое шифрование в настоящее время используют следующие алгоритмы:

- RSA RC4
- Salsa20
- HC-256
- WAKE

Достоинства и недостатки симметричного шифрования

Симметричные алгоритмы требуют меньше ресурсов и демонстрируют большую скорость шифрования, чем асимметричные алгоритмы. Большинство симметричных шифров предположительно устойчиво к атакам с помощью квантовых компьютеров, которые в теории представляют угрозу для асимметричных алгоритмов.

Слабое место симметричного шифрования — обмен ключом. Поскольку для работы алгоритма ключ должен быть и у отправителя, и у получателя сообщения, его необходимо передать; однако при передаче по незащищенным каналам его могут перехватить и использовать посторонние. На практике во многих системах эта проблема решается шифрованием ключа с помощью асимметричного алгоритма.

Область применения симметричного шифрования

Симметричное шифрование используется для обмена данными во многих современных сервисах, часто в сочетании с асимметричным шифрованием. Например, мессенджеры защищают с помощью таких шифров переписку (при этом ключ для симметричного шифрования обычно доставляется в асимметрично зашифрованном виде), а сервисы для видеосвязи — потоки аудио и видео. В защищенном транспортном протоколе *TLS* симметричное шифрование используется для обеспечения конфиденциальности передаваемых данных.

Симметричные алгоритмы не могут применяться для формирования цифровых подписей и сертификатов, потому что секретный ключ при использовании этого метода должен быть известен всем, кто работает с шифром, что противоречит самой идее электронной подписи (возможности проверки ее подлинности без обращения к владельцу).

Описание рассматриваемых алгоритмов симметричного шифрования

Стандарт шифрования DES (Data Encryption Standard) был разработан в 1970-х годах, он базируется на алгоритме DEA. Исходные идеи алгоритма шифрования данных DEA (data encryption algorithm) были предложены компанией IBM еще в 1960-х годах и базировались на идеях, описанных Клодом Шенноном в 1940-х годах. Перво-

начально эта методика шифрования называлась lucifer (разработчик Хорст Фейштель, название dea она получила лишь в 1976 году. Lucifer был первым блочным алгоритмом шифрования, он использовал блоки размером 128 бит и 128-битовый ключ. По существу этот алгоритм являлся прототипом DEA. В 1986 в Японии (NIT) разработан алгоритм FEAL(Fast data Encipherment ALgorithm), предназначенный для использования в факсах, модемах и телефонах (длина ключа до 128 бит). Существует и ряд других разработок.

Triple DES - Симметричный блочный шифр, созданный Уитфилдом Диффи, Мартином Хеллманом и Уолтом Тачманном в 1978 году на основе алгоритма DES, с целью устранения главного недостатка последнего — малой длины ключа (56 бит), который может быть взломан методом полного перебора ключа. Скорость работы 3DES в 3 раза ниже, чем у DES, но криптостойкость намного выше — время, требуемое для криптоанализа 3DES, может быть в миллиард раз больше, чем время, нужное для вскрытия DES. 3DES используется чаще, чем DES, который легко ломается при помощий сегодняшних технологий (в 1998 году организация Electronic Frontier Foundation, используя специальный компьютер DES Cracker, разбила DES за 3 дня). 3DES является простым способом устранения недостатков DES. Алгоритм 3DES построен на основе DES, поэтому для его реализации возможно использовать программы, созданные для DES.

RSA RC2 и RSA RC4 алгоритмы шифрования с переменной длиной ключа для быстрого шифрования больших объемов информации. Разработаны одним из основателей компании RSA Data Security Inc. Р. Ривестом (R. Rivest). Работают быстрее DES и способны повышать степень защиты за счет применения более длинного ключа. RC2 - это блочный шифр и его можно использовать вместо DES. RC4 представляет собой потоковый шифр и работает почти в 10 раз быстрее DES.

2. Алгоритмы хеширования данных

В настоящее время практически ни одно приложение криптографии не обходится без использования хэширования.

Хэш-функции – это функции, предназначенные для «сжатия» произвольного сообщения или набора данных, записанных, как правило, в двоичном алфавите, в некоторую битовую комбинацию фиксированной длины, называемую сверткой. Хэш-функции имеют разнообразные применения при проведении статистических экспери-

ментов, при тестировании логических устройств, при построении алгоритмов быстрого поиска и проверки целостности записей в базах данных. Основным требованием к хэш-функциям является равномерность распределения их значений при случайном выборе значений аргумента.

Криптографические хеш-функции

Криптографической хеш-функцией называется всякая хеш-функция, являющаяся криптостойкой, то есть удовлетворяющая ряду требований специфичных для криптографических приложений. Криптографические хэш-функции распространены очень широко. Они используются для хранения паролей, аутентификации источника данных, построения систем контроля целостности данных при их передаче или хранении, для защиты данных в системах проверки файлов, для обнаружения вредоносного программного обеспечения, для кодирования информации в блокчейне (блок — основной примитив, обрабатываемый Биткойном и Эфириумом).

В наши дни существует много криптографических алгоритмов. Они бывают разные и отличаются по сложности, разрядности, криптографической надежности, особенностям работы. Алгоритмы хеширования — идея не новая. Они появилась более полувека назад, причем за много лет с принципиальной точки зрения мало что изменилось. Но в результате своего развития *хеширование данных* приобрело много новых свойств, поэтому его применение в сфере информационных технологий стало уже повсеместным.

Ключевые и бесключевые хэш-функции, их свойства и требования к ним

Выделяют два важных вида криптографических хэш-функций — ключевые и бесключевые. Ключевые хэш-функции называют кодами аутентификации сообщений. Они дают возможность без дополнительных средств гарантировать как правильность источника данных, так и целостность данных в системах с доверяющими друг другу пользователями.

Бесключевые хэш-функции называются кодами обнаружения ошибок. Они дают возможность с помощью дополнительных средств (шифрования, например) гарантировать целостность данных. Эти хэш-функции могут применяться в системах как с доверяющими, так и не доверяющими друг другу пользователями.

Хеш или хэш — это криптографическая функция хеширования (function), которую обычно называют просто хэшем. Хеш-функция представляет собой математиче-

ский алгоритм, который может *преобразовать произвольный массив данных в строку фиксированной длины*, состоящую из цифр и букв.

Основным требованием к хэш-функциям является равномерность распределения их значений при случайном выборе значений аргумента. Для криптографических хеш-функций также важно, чтобы при малейшем изменении аргумента значение функции сильно изменялось. Это называется лавинным эффектом.

К ключевым функциям хэширования предъявляются следующие требования:

- невозможность фабрикации,
- невозможность модификации.

Первое требование означает высокую сложность подбора сообщения с правильным значением свертки. Второе — высокую сложность подбора для заданного сообщения с известным значением свертки другого сообщения с правильным значением свертки.

К бесключевым функциям предъявляют требования:

- однонаправленность,
- устойчивость к коллизиям,
- устойчивость к нахождению второго прообраза.

Под однонаправленностью понимают высокую сложность нахождения сообщения по заданному значению свертки. Следует заметить что на данный момент нет используемых хэш-функций с доказанной однонаправленностью.

Под устойчивостью к коллизиям понимают сложность нахождения пары сообщений с одинаковыми значениями свертки. Обычно именно нахождение способа построения коллизий криптоаналитиками служит первым сигналом устаревания алгоритма и необходимости его скорой замены.

Под устойчивостью к нахождению второго прообраза понимают сложность нахождения второго сообщения с тем же значением свертки для заданного сообщения с известным значением свертки.

Популярные хэш-алгоритмы

Алгоритмы CRC16/32 — контрольная сумма (не криптографическое преобразование).

Алгоритмы MD2/4/5/6. Являются творением Рона Райвеста, одного из авторов алгоритма RSA. Алгоритм MD5 имел некогда большую популярность, но первые

предпосылки взлома появились еще в конце девяностых, и сейчас его популярность стремительно падает. Алгоритм MD5 — один из первых стандартов алгоритма, который применялся в целях проверки целостности файлов (контрольных сумм). Также с его помощью хранили пароли в базах данных web-приложений. Функциональность относительно проста — алгоритм выводит для каждого ввода данных фиксированную 128-битную строку, задействуя для вычисления детерминированного результата однонаправленные тривиальные операции в нескольких раундах. Особенность — простота операций и короткая выходная длина, в результате чего MD5 является относительно легким для взлома. А еще он обладает низкой степенью защиты к атаке типа «дня рождения».

Secure Hashing Algorithm (SHA1) — алгоритм, созданный Агентством национальной безопасности (NSA). Он создает 160-битные выходные данные фиксированной длины. На деле SHA1 лишь улучшил MD5 и увеличил длину вывода, а также увеличил число однонаправленных операций и их сложность. Однако каких-нибудь фундаментальных улучшений не произошло, особенно когда разговор шел о противодействии более мощным вычислительным машинам. Со временем появилась альтернатива — SHA2, а потом и SHA3. Алгоритмы линейки SHA широко распространенные сейчас алгоритмы.

Российские стандарты — ГОСТ 34.11-94, . ГОСТ Р 34.11-2012, ГОСТ 34.11-2018. Межгосударственный стандарт ГОСТ 34.11-2018 введен в действие в качестве национального стандарта Российской Федерации с 1 июня 2019 г.

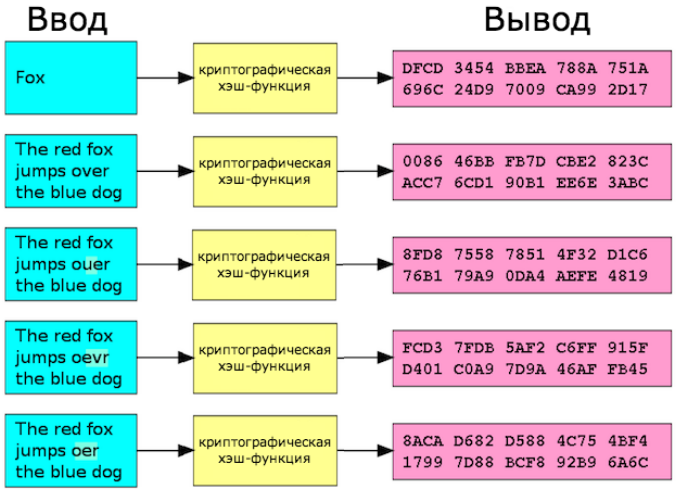
Пример детерминированности алгоритмов хеширования

Основная идея используемых в данном случае функций — применение *детерминированного алгоритма*. Речь идет об алгоритмическом процессе, выдающем уникальный и предопределенный результат при получении входных данных. То есть при приеме одних и тех же входных данных будет создаваться та же самая строка фиксированной длины (использование одинакового ввода каждый раз приводит к одинаковому результату). Детерминизм — важное свойство этого алгоритма. *И если во входных данных изменить хотя бы один символ, будет создан совершенно другой хэш*.

Убедиться в этом можно на любом онлайн-генераторе (например, **Crypto Demo**). Набрав слово «Otus» и воспользовавшись алгоритмом **SHA1** (Secure Hashing Algorithm), мы получим хеш **7576750f9d76fab50762b5987739c18d99d2aff7**. При изме-

нении любой буквы изменится и результат, причем изменится полностью. Мало того, если просто поменять регистр хотя бы одной буквы, итог тоже будет совершенно иным: если написать «otus», алгоритм хэш-функции отработает со следующим результатом: **1bbd70dc1b6fc84e5617ca8703c72c744b3b4fc1**. Хотя общие моменты все же есть: строка *всегда* состоит из сорока символов.

В примере речь шла о применении хэш-алгоритма для слова из 4 букв. Но с тем же успехом можно вставить слово из 1000 букв — все равно после обработки данных на выходе получится значение из 40 символов. Аналогичная ситуация будет и при обработке полного собрания сочинений Льва Толстого.



Криптостойкость функций хеширования

Говоря о криптостойкости, предполагают выполнение ряда требований. То есть хороший алгоритм обладает несколькими свойствами:

- при изменении одного бита во входных данных, должно наблюдаться изменение всего хэша;
- алгоритм должен быть устойчив к коллизиям;
- алгоритм должен быть устойчив к восстановлению хешируемых данных, то есть должна обеспечиваться высокая сложность нахождения прообраза, а вычисление хэша не должно быть простым.

Проблемы хэш-функций

Одна из проблем криптографических функций хеширования — *неизбежность коллизий*. Раз речь идет о строке фиксированной длины, значит, существует вероятность, что *для каждого ввода возможно наличие и других входов, способных привести*

к тому же самому хешу. В результате хакер может создать коллизию, позволяющую передать вредоносные данные под видом правильного хэша.

Цель хороших криптографических функций — максимально усложнить вероятность нахождения способов генерации входных данных, хешируемых с одинаковым значением. Как уже было сказано ранее, вычисление хэша не должно быть простым, а сам алгоритм должен быть устойчив к «атакам нахождения прообраза». Необходимо, чтобы на практике было чрезвычайно сложно (а лучше — невозможно) вычислить обратные детерминированные шаги, которые предприняты для воспроизведения созданного хешем значения.

Если S = hash (x), то, в идеале, нахождение x должно быть практически невозможным.

Задание 1. Изучение алгоритмов симметричного шифрования

1. Установите демонстрационный криптографический модуль **Crypto Demo**.
2. Запустите демонстрационный криптографический модуль **Crypto Demo**, выбрав последовательно **Пуск → Все программы → Cryptography Demonstration → CryptoDemo 1.0**.
3. Введите в поле **Key** окна программы значение ключа шифрования: **012345678901234567890123**4. Введите в поле **Data** окна программы текст шифруемого сообщения. Текст нужно обязательно ввести, т.к. при копировании возникают ошибки.
4. Зашифруйте набранный текст выбранным ключом, выбирая последовательно в поле **Encryption Algorithm** каждый из доступных алгоритмов симметричного шифрования и нажимая кнопку **Encrypt**.
5. Выпишите значение зашифрованного текста в кодировке **BASE64** из поля **Encrypted Data**.

Triple DES (3DES):

DES:

RSA RC4:

RSA RC2:

Различается ли длина зашифрованного текста при выборе различных алгоритмов шифрования? Почему? _____

6. Измените одну любую букву в открытом тексте. Зашифруйте изменённый текст выбранным ключом, выбирая последовательно в поле **Encryption Algorithm** каждый из доступных алгоритмов симметричного шифрования и нажимая кнопку **Encrypt**.

Насколько сильно изменилось значение зашифрованного текста в поле ***Encrypted Data*** *по сравнению с выписанным ранее?* _____

7. Закройте все открытые окна.

Задание 2. Изучение алгоритмов хеширования

1. Запустите демонстрационный криптографический модуль **Crypto Demo**, выбрав последовательно ***Пуск → Все программы → Cryptography Demonstration → CryptoDemo 1.0***.
2. Переключитесь на вкладку **Hashing**. Введите в поле данных окна программы изречение Кузьмы Пруткова из файла **C:\Temp\XOR\XOR_cmd\plain.txt**. Текст нужно обязательно ввести, т.к. при копировании возникают ошибки.
3. Выберите последовательно в поле **Hash Algorithm** каждый из доступных алгоритмов хеширования и нажмите кнопку **Get Hash**. Выпишите значение хэша для каждого из алгоритмов.

MD2:

MD4:

MD5:

SHA1:

*Различается ли длина хэша при выборе различных алгоритмов хеширования файла?*_____

4. Измените любую одну букву в исходном тексте. Посчитайте и выпишите хэш изменённого текста.

MD2: _____

MD4: _____

MD5: _____

SHA1: _____

Насколько сильно изменилось его значение по сравнению с выписанным ранее? _____

Почему? _____

5. Закройте все открытые окна.