5076

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ЗАЩИТЫ ИНФОРМАЦИИ В МТКС

Методические указания к лабораторным работам

Рязань 2016

УДК 004.056.55

Исследование алгоритмов защиты информации в МТКС: методические указания к лабораторным работам / Рязан. гос. радиотехн. ун-т; сост.: С.Н. Кириллов, В.Т. Дмитриев, Д.С. Семин, А.Д. Суздальцев. Рязань, 2016. 24 с.

Содержат материал для выполнения лабораторных работ, посвященных изучению алгоритмов, обеспечивающих защиту информации в МТКС. В первой лабораторной работе рассмотрены современные симметричные алгоритмы защиты информации. Во второй лабораторной работе приведены методические указания по алгоритмам защиты информации с открытым ключом и создание электронной цифровой подписи.

Предназначены для обучения бакалавров и магистров по направлениям подготовки «Инфокоммуникационные технологии и системы связи» и специалистов по специальности «Радиоэлектронные системы и комплексы».

Табл. 6. Ил. 2. Библиогр.: 6 назв.

Защита информации, алгоритмы шифрования, электронная цифровая подпись

Печатается по решению редакционно-издательского совета Рязанского государственного радиотехнического университета.

Рецензент: кафедра радиоуправления и связи Рязанского государственного радиотехнического университета (зав. кафедрой С.Н.Кириллов)

Исследование алгоритмов защиты информации в МТКС

Составители: Кириллов Сергей Николаевич
Дмитриев Владимир Тимурович
Семин Дмитрий Сергеевич
Суздальцев Александр Дмитриевич

Редактор Н.А. Орлова
Корректор М.Е. Цветкова

Подписано в печать 20.08.16. Формат бумаги 60x84 1/16.
Бумага писчая. Печать трафаретная. Усл. печ. л. 1,5.
Тираж 50 экз. Заказ 3219.
Рязанский государственный радиотехнический университет.
390005, Рязань, ул. Гагарина, 59/1.
Редакционно-издательский центр РГРТУ.

Лабораторная работа № 1

ИССЛЕДОВАНИЕ АЛГОРИТМОВ КРИПТОГРАФИЧЕСКОЙ ЗАЩИТЫ ИНФОРМАЦИИ

Цель работы

Изучение современных симметричных алгоритмов защиты информации, криптографических хеш-функций. Получение практических навыков работы с программой OpenSSL [3].

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Обзор современных алгоритмов защиты информации

Защита информации - совокупность методов, средств и мероприятий, обеспечивающих выполнение следующих функций [1]:

- исключение несанкционированного доступа (НСД) к ресурсам ЭВМ, программам и данным;
- проверку целостности информации;
- устранение неразрешенного применения программ (защита от копирования).

Безусловная направленность использования цифровых методов передачи и хранения информации позволяет реализовывать унифицированные алгоритмы и методы для защиты дискретной (текст, факс) и непрерывной (речь) информации. Надежным методом защиты информации от НСД является шифрование (криптография). Шифрованием называют процесс преобразования открытых данных в зашифрованные или зашифрованных данных в открытые по конкретным правилам с использованием ключей. В англоязычной литературе зашифрование/расшифрование - enciphering/deciphering.

При поддержке криптографических способов возможно: криптография информации, создание электронной подписи, назначение ключей шифрования, защита от случайного или умышленного изменения информации.

Ключ шифрования - это случайная или специальным образом созданная по паролю последовательность бит, являющаяся переменным параметром алгоритма шифрования. Если зашифровать одни и те же данные одним алгоритмом, но с разными ключами, результаты получатся разные. В коммерческих программах для шифрования (WinRAR, и т.д.) ключ формируется из пароля, который создает пользователь. Ключ шифрования для любых алгоритмов бывает разной длины, которая, как правило, измеряется в битах.

Алгоритмы шифрования делятся на два больших класса: симметричные (AES, Blowfish, DES, DES3, IDEA, Camellia) и

асимметричные (RSA, DSA). Симметричные алгоритмы шифрования применяют единственный ключ для шифрования информации и для ее расшифровывания, а асимметричные алгоритмы используют два ключа - один для зашифровывания, другой для расшифровывания.

Симметричные делятся на группы [2]:

1) потоковые (зашифровывание потока данных):
– с одноразовым или бесконечным ключом;
– с конечным ключом;
– на основе генератора псевдослучайных чисел (ПСЧ).

2) блочные (зашифрование данных поблочно):
– шифры перестановки;
– шифры замены (подстановки, S-блоки):
– моноалфавитные;
– полиалфавитные.

Симметричные алгоритмы:
– DES (Data Encryption Standard, США);
– IDEA (International Data Encryption Algorithm, фирма Ascom-Tech AG, Швейцария);
– AES (Advanced Encryption Standard, США);
– Blowfish (1993год, разработан Брюсом Шнайером, США);
– Camellia (2000г, компании Nippon Telegraph and Telephone Corporation и Mitsubishi Electric Corporation, Япония).

Асимметричные алгоритмы:
– Диффи-Хеллман DH (Diffie, Hellman);
– Райвест-Шамир-Адлеман RSA (Rivest, Shamir, Adleman);
– DSA (Digital Signature Algorithm).

Кроме того, существует классификация алгоритмов криптографии на шифры (ciphers) и коды (codes) [1].

1.2. Симметричные алгоритмы защиты информации

Симметричные криптосистемы – метод криптографии, в котором для зашифровывания и расшифровывания используется один и тот же ключ [3]. Ключ алгоритма должен держаться в секрете обеими сторонами. Метод криптографии выбирается сторонами до начала обмена сообщениями. Алгоритмы шифрования информации широко распространены и применяются в ЭВМ в системах сокрытия конфиденциальной и коммерческой информации от несанкционированного применения третьими лицами. Главным принципом в них является условие, что передатчик и приемник заранее знают механизм шифрования, а также ключ к посылке, без

которых информация представляет собой всего лишь набор символов, не имеющих смысла.

В настоящее время симметричные шифры – это блочные шифры, которые обрабатывают информацию блоками определённой длины (обычно 64, 128 бит), применяя к блоку ключ в установленном порядке, обычно, несколькими циклами перемешивания и подстановки, называемыми раундами [3]. Результатом повторения раундов является лавинный эффект – нарастающая потеря соответствия битов между блоками открытых и зашифрованных данных. Типичным способом построения алгоритмов симметричного шифрования является сеть Фейстеля [3]. Алгоритм строит схему шифрования на основе функции F(D, K), где D – порция данных размером вдвое меньше блока шифрования, а K – «ключ прохода» для данного прохода. От функции не требуется обратимость, т.е. обратная ей функция может быть неизвестна. Достоинства сети Фейстеля – почти полное совпадение дешифрования с шифрованием (единственное отличие – обратный порядок «ключей прохода» в расписании), что сильно облегчает аппаратную реализацию.

Операция перестановки перемешивает биты сообщения по определенному закону. В аппаратных реализациях она обычно реализуется как перепутывание проводников. Именно операции перестановки дают возможность достижения «эффекта лавины». Операция перестановки линейна f(a), где xor f(b) = f(a xor b).

Операция подстановки выполняется как замена значения некой части сообщения (часто в 4, 6 или 8 бит) на стандартное, жестко встроенное в алгоритм иное число путем обращения к постоянному массиву и привносит в алгоритм нелинейность.

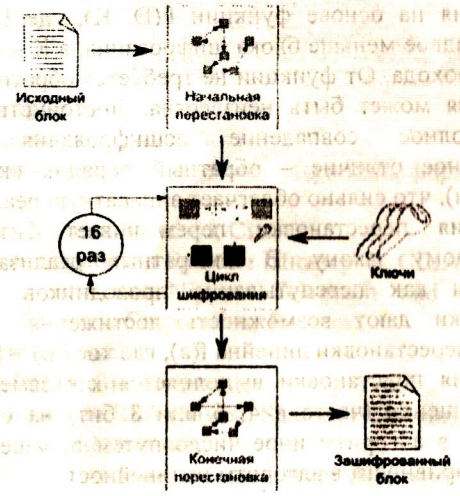
Достоинства в сравнении с асимметричными алгоритмами: скорость (на 3 порядка выше), простота реализации (за счет простых операций), требуется меньшая длина ключа для сопоставимой стойкости, изученность.

Недостатки в сравнении с асимметричными алгоритмами: сложность управления ключами в большой сети (для сети 10 абонентов требуется 45 ключей, для 1000 – 499500 ключей и т.д.), сложность обмена ключами, невозможность их использования для подтверждения авторства.

AES (Rijndael) – симметричный алгоритм блочного шифрования. Размер блока 128 бит, размер ключей 128, 192, 256 бит. В качестве стандарта принят вариант шифра только с размером блока 128 бит.

DES – алгоритм симметричного шифрования, разработанный фирмой IBM. Размер блока для DES равен 64 бита. В основе алгоритма

лежит сеть Фейстеля с 16-ю циклами (раундами) и ключом, имеющим длину 56 бит. Алгоритм использует комбинацию нелинейных (S-блоки) и линейных (перестановки) преобразований. Прямым развитием DES в настоящее время является алгоритм Triple DES (3DES). В 3DES шифрование/расшифрование выполняются путём троекратного выполнения алгоритма DES. Исходный текст T (блок 64 бит) преобразуется с помощью начальной перестановки. Полученный после перестановки 64-битовый блок участвует в 16-ти циклах преобразования Фейстеля. Конечная перестановка является обратной к первоначальной перестановке, как показано на рисунке.



Обобщенная схема шифрования алгоритмом DES

IDEA – симметричный блочный алгоритм криптографии информации [2]. Начальную версию механизма, который должен был заменить DES, спроектировали в 1990 году Лай Сюэцзя (Xuejia Lai) и Джеймс Мэсси (James Massey). Так как IDEA работает с 128-битным ключом и 64-битным размером блока, то доступный текст разделяется на группы по 64 бита. В случае когда разделение невозможно реализовать, последняя часть расширяется различными методами фиксированной последовательностью бит. Чтобы избежать утечки данных о каждом отдельном блоке, используются различные режимы шифрования. Каждая начальная нешифрованная 64-битная группа разделится на четыре подгруппы по 16 бит каждая,

исходя из того, что алгебраические операции, применяемые в процессе криптографии, происходят над 16-битными числами.

Сравнение алгоритма IDEA с другими алгоритмами [2]

В табл. 1.1 показано сравнение алгоритма IDEA с алгоритмами DES, Blowfish и ГОСТ 28147-89.

Таблица 1.1. Характеристики алгоритмов

Алгоритм	Размер ключа, бит	Длина блока, бит	Число раундов	Скорость шифрования (кбайт/сек)
DES	56	64	16	35
IDEA	128	64	8	70
Blowfish	32-448	64	16	135

Как видно из таблицы, размер ключа у IDEA больше, чем у DES, но меньше, чем у Blowfish. Скорость шифрования IDEA больше в 2 раза, чем у DES, но почти в 2 раза меньше, чем у Blowfish. Длина ключа у IDEA имеет размер 128 бит, против 56 бит у DES, что является хорошим улучшением против полного перебора ключей.

Области применения IDEA: шифрование аудио и видеоданных для кабельного телевидения, видеоконференций, дистанционного обучения; линия связи через модем, роутер, GSM-технологию.

Blowfish – криптографический алгоритм, реализующий блочное шифрование с переменной длиной ключа [3]. Он разработан Брюсом Шнайером в 1993 году и выполнен на простых и быстрых операциях: XOR, подстановка, сложение.

Критериями проектирования Blowfish были: скорость (шифрование на 32-битных процессорах происходит за 26 тактов), простота (за счёт использования простых операций, уменьшающих вероятность ошибки реализации алгоритма), компактность (возможность работать в менее, чем 5 кбайт памяти), настраиваемая безопасность (изменяемая длина ключа).

Сложная схема выработки ключевых элементов существенно затрудняет атаку на алгоритм методом перебора, но делает его непригодным для использования в системах, где ключ часто меняется, и на каждом ключе шифруются небольшие по объему данные.

Области применения данного алгоритма [1]:

– хеширование паролей;

– защита электронной почты и файлов;

– обеспечение безопасности в протоколах сетевого и транспортного уровня.

Хотя Blowfish по скорости опережает свои аналоги, но при увеличении частоты смены ключа основное время его работы будет уходить на подготовительный этап, что в сотни раз уменьшает его эффективность.

Camellia – алгоритм симметричного блочного шифрования [3]. Размер блока 128 бит, размер ключа 128, 192,256 бит. Представлен 10 марта 2000г. Является разработкой японских компаний Nippon Telegraph and Telephone Corporation и Mitsubishi Electric Corporation. Camellia является дальнейшим развитием алгоритма E2. Структура алгоритма основана на классической цепи Фейстеля с предварительным и финальным «забеливанием». Цикловая функция использует нелинейное преобразование (S-блоки), блок линейного рассеивания каждые 16 циклов (побайтовая операция XOR) и байтовую перестановку. В зависимости от длины ключа имеет 18 циклов (128 разрядный ключ) либо 24 цикла (192 и 256 разрядный ключ).

В табл. 1.2 показано сравнение с другими алгоритмами защиты информации.

Таблица 1.2. Сравнение алгоритмов защиты информации

Алгоритмы	Время вычисления ключей, нс	Время шифрования, нс	Пропускная способность, Mb/s
DES	–	55.11	1161.31
AES	57.39	65.64	1950.03
Camellia	24.36	109.35	1170.55

1.3. Криптографические хеш-функции

MD4 – алгоритм хеширования, разработанный в 1990 году, генерирует 128-разрядное хеш-значение, называемое дайджестом сообщения [1].

Главные цели алгоритма MD4 [3]:

– защищенность: это нормальное правило к хэш-функции, заключающееся в том, чтобы нельзя было вычислить и обнаружить пару сообщений, имеющих один и тот же дайджест;

– скорость: необходимо, чтобы программное исполнение метода выполнялось довольно быстро. Например, алгоритм должен соответствовать этим требованиям на 32-битной архитектуре;

– простота и компактность: метод должен быть несложным в описании и довольно простым в программировании, исключая большие программы и подстановочные таблицы. Такие требования

обладают не только безусловными программными достоинствами, но и необходимы с точки зрения защищенности, так как для анализа слабых участков лучше применять простой алгоритм.

Этими целями озадачились и при проектировании алгоритма MD5. MD5 оказался наиболее сложным, а значит, намного медленнее в исполнении, чем MD4. Но повышение сложности алгоритма обосновывается увеличением уровня безопасности.

Основные отличия между этими двумя алгоритмами состоят в следующем [2]:

– MD4 применяет три цикла, состоящих из 16 этапов каждый, в то время как MD5 использует четыре цикла из 16 этапов каждый;

– в MD4 вспомогательная константа в первом цикле не используется. Такая константа применяется для любого из этапов во втором цикле. Другая дополнительная константа употребляется для каждого из этапов в третьем цикле. В MD5 разные дополнительные константы, T [i] используются для каждого из 64 этапов;

– MD5 использует четыре простейшие логические функции по одной на каждом цикле по сравнению с тремя в MD4;

– в MD5 на любом этапе данный результат суммируется с результатом предыдущего. Например, результатом первого этапа оказывается измененное слово A. Результат второго этапа сохраняется в D и формируется прибавлением A к циклически сдвинутому влево на заданное количество бит результату элементарной функции. Подобным образом итог третьего этапа хранится в C и образуется добавлением D к циклически сдвинутому влево результату элементарной функции. В MD4 последнее сложение не используется.

Зная MD5, невозможно восстановить входное сообщение, так как одному MD5 могут соответствовать разные сообщения. Используется для проверки подлинности опубликованных сообщений путем сравнения дайджеста сообщения с опубликованным («проверка хеша»).

Хеш-функция содержит 128 бит (16 байт) и обычно представляется как последовательность из 32 шестнадцатеричных цифр. Любая малая поправка входного сообщения приведет к полному изменению хеш-функции. Такая характеристика алгоритма называется лавинным эффектом.

RIPEMD-160 – является 160-битной криптографической хэш-функцией. Он предназначен для замены менее безопасных 128-битных хэш-функций MD4, MD5 и RIPEMD. Также существуют 128-, 256- и 320-битные версии этого алгоритма, которые соответственно называются RIPEMD-128, RIPEMD-256 и RIPEMD-320. 128-битная

версия представляет собой лишь замену оригинальной RIPEMD, которая также была 128-битной и в которой были найдены уязвимости. 256и 320-битные версии отличаются удвоенной длиной дайджеста, что уменьшает вероятность взлома, но при этом функции не являются более криптостойкими.

SHA1 (Secure Hash Algorithm 1) – алгоритм криптографического хеширования. Для входного сообщения произвольной длины алгоритм генерирует 160-битное хеш-значение, называемое также дайджестом сообщения. Он используется во многих криптографических приложениях и протоколах. Также алгоритм рекомендован в качестве основного для государственных учреждений в США. Принципы, положенные в основу SHA-1, аналогичны тем, которые использовались Рональдом Ривестом при проектировании MD4. SHA-1 реализует хеш-функцию, построенную на идее функции сжатия. Входами функции сжатия являются блок сообщения длиной 512 бит и выход предыдущего блока сообщения. Выход представляет собой значение всех хеш-блоков до этого момента.

Криптоанализ хеш-функций нацелен на определение слабых мест к действию различного вида атак. Основные из них:

– поиск коллизий (состояние, когда двум разным начальным посылкам соответствует одно и то же хеш-значение),

– поиск прообраза начального сообщения происходит по его хешу.

При решении способом «грубой силы»:

– второстепенной задаче необходимо 2160 операций,

– первостепенная же требует в среднем 2160/2 = 280 операций, если использовать атаку Дней рождения.

От стабильности хеш-функции к поиску коллизий зависит защищенность электронной цифровой подписи с применением данного хеш-алгоритма.

2. ТЕХНИЧЕСКАЯ ЧАСТЬ

2.1. Описание лабораторной модели

Данная лабораторная работа выполняется с помощью криптографического пакета с открытым исходным кодом OpenSSL, который запускается с помощью командной строки. Командная строка представляет собой одну из возможностей Windows, обеспечивающую ввод команд MS-DOS и других компьютерных команд. Важность этой командной строки состоит в том, что она позволяет выполнять задачи без помощи графического интерфейса Windows.

При работе с командной строкой сам термин *командная строка* обозначает также и закрывающую угловую скобку (>, иначе: символ больше). Это указывает на то, что интерфейс командной строки может принимать команды. Другая важная информация, например текущий рабочий каталог (или расположение), где будет выполняться данная команда, может быть также указана в командной строке. Например, если при открытии окна командной строки в этой строке отображаются текст «C:\>» и мигающий курсор справа от закрывающей угловой скобки (>), это означает, что введенная команда будет выполняться на всем диске C данного компьютера. Для открытия командной строки необходимо нажать кнопку *Пуск*. В поле поиска введите *Командная строка*, а затем в списке результатов выберите пункт *Командная строка*.

2.2. Описание программы OpenSSL [3]

В данной работе используется программа с открытым исходным кодом OpenSSL. OpenSSL — универсальный криптографический инструмент, построенный вокруг протоколов SSL/TLS и сертификатов X.509. Название SSL переводится как система безопасных сокетов (secure socket layer). OpenSSL используется практически всеми сетевыми серверами для защиты передаваемой информации.

2.3. Компоненты OpenSSL, выполняющие шифрование и хеширование данных

Для использования симметричного шифрования используется команда openssl enc – cipher, где cipher – это одно из имен симметричных шифров, которыми являются следующие: bf(blowfish – 128 bit), des(56 bit), des3(168 bit), rc4(128 bit), rc5(128 bit), idea(128 bit), aes-128(128 bit),aes-256(256 bit). Для указания входного и выходного файла используют опции –in и –out. По умолчанию пароль предлагается вводить в интерактивном режиме. Возможно организовать ввод пароля, используя опцию –k, но это очень небезопасно, так как может сохраняться история командной строки. Для расшифровывания зашифрованных данных применяют опцию openssl cipher –d (при этом алгоритм шифрования и дешифрования должен совпадать!). Опция –salt должна использоваться всегда, если секретный ключ формируется на основе пароля. Для выбора алгоритма подмешивания используется «случайная соль» (salt), если вы шифруете одни и те же данные в разное время одним и тем же алгоритмом и паролем, то результаты будут разные. Когда используется «соль», первые 8 байтов резервируются под нее. Она

генерируется случайным образом при шифровании файла и считывается с зашифрованного файла во время дешифрации.

2.4. Шифрование в OpenSSL симметричными алгоритмами

Рассмотрим практические методы шифрования файлов с использованием симметричных алгоритмов шифрования [3]. Пусть необходимо зашифровать файл «text.txt» с использованием алгоритма des3.

Синтаксис команды для шифрования алгоритмом des3 [3]:

Openssl des3 –salt –in file.txt –out file.txt,

где des3 - алгоритм шифрования; -in – путь к файлу, который необходимо зашифровать; -out – путь, куда сохранится зашифрованный файл.

Подробный пример шифрования алгоритмом des3:

openssl des3 –salt –in D:/text.txt –out D:/des3.txt,

где

D:/ - путь к файлу, который необходимо зашифровать.

Пример дешифрования файла алгоритмом des3:

Openssl des3 –d –in des3.txt –out text.txt,

где –d опция дешифрования.

Синтаксис команды шифрования файла алгоритмом blowfish с использованием опции –k:

Openssl bf –k mypassword – salt –in text.txt –out bf.txt,

где –k –опция, позволяющая вводить пароль не интерактивно;

«mypassword» -пароль, который вводит пользователь.

Кодирование в формат Base64 является стандартным методом для преобразования 8-битной двоичной информации, в ограниченное подмножество символов ASCII для безопасной транспортировки через системы электронной почты и другие системы, которые не поддерживают 8-битный формат.

По умолчанию зашифрованный файл создается в бинарном формате. Чтобы зашифровать файл в кодировке Base64, необходимо добавить опцию –a.

Синтаксис шифрации файла алгоритмом aes-128 и кодировкой base64:

Openssl aes-128-cbc –a –salt –in text.txt –out aes_128.txt,

где Aes- 128-cbc – алгоритм шифрования; -a – кодирование base64; -in – путь к файлу, который необходимо зашифровать, -out путь, куда сохранится зашифрованный файл.

Синтаксис дешифрации файла алгоритмом aes -128-cbc и кодировкой base64:

Openssl aes-128-cbc –a –d –in aes_128.txt –out text.txt.

2.5. Вычисление контрольных сумм в Openssl

Для вычислений хешей (контрольных сумм) используется команда openssl dgst –hashalg или openssl hashalg. Вычисляется хеш сообщения фиксированной длины в виде одной строки. Могут применяться следующие алгоритмы хеширования: md2(128 bit), md4(128 bit), md5(128bit), mdc2(128 bit), sha(160 bit), sha1(160 bit), ripemd160(160 bit). При использовании опции –c хеш вычисляется в виде одной строки, разделённой на пары чисел двоеточием.

Синтаксис команды вычисления md5 хеш файла [3]:

openssl md5 –c text.txt.

Результат:

MD5(text.txt)=09 : c9 : ee : b6 : df : 5a : 98 : b9 : df : 26 : 9e : 9c : 0e : 3d : bf.

Синтаксис команды вычисления sha1 хеш этого же файла:

Openssl sha1 text.txt.

Результат:

SHA1(text.txt)=714f7c520fe563be927f3f82e7edab4e65cfadcc.

3. ПРАКТИЧЕСКАЯ ЧАСТЬ

3.1. Шифрования файла симметричными алгоритмами шифрования

Зашифровать файл «text.txt» симметричными алгоритмами шифрования. Алгоритмы шифрования выбрать в соответствии с табл. 1.3.

Таблица 1.3. Алгоритмы шифрования

№ бригады	1	2	3	4
Алгоритм шифрования	Blowfish, Aes-128-cbc base64	Des, Aes-256-cbc	Des3 base64, Camellia-128-cbc	Idea, Camellia-256-cbc

3.2. Вычисление контрольных сумм

1. Вычислить контрольные суммы файла «text.txt» алгоритмами хеширования.

2. Выбрать в соответствии с табл. 1.4 алгоритмы хеширования.

Таблица 1.4. Алгоритмы хеширования

№ бригады	1	2	3	4
Алгоритм хеширования	Md4, Sha	Md5, Rmd160	Mdc2, Sha1	Md5,Md4

4. СОДЕРЖАНИЕ ОТЧЕТА

Отчет о лабораторной работе должен содержать:
– цель работы;
– структурную схему симметричного алгоритма шифрования;
– структурную схему алгоритма хеширования;
– полученные результаты.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое защита информации? Какой метод используется для защиты информации? Дайте определение ключа шифрования.
2. Перечислите группы алгоритмов шифрования.
3. Назовите виды симметричных алгоритмов. Их особенности.
4. По какому принципу работают симметричные алгоритмы? Какими достоинствами и недостатками обладают?
5. Что такое хеш-функции? Какие алгоритмы хеширования используются?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Гатчин Ю.А., Коробейников А.Г. Основы криптографических алгоритмов: учеб. пособие. - СПб.: СПбГИТМО (ТУ), 2002 .- 29 с.
2. Коробейников А. Г. Математические основы криптографии: учеб. пособие. СПб: СПб ГИТМО (ТУ), 2002. - 41 с.
3. Рассел Дж., Кох Р. Симметричные криптосистемы. –VSD, 2012. – 106 с.

Лабораторная работа № 2

ИССЛЕДОВАНИЕ АЛГОРИТМОВ ЗАЩИТЫ ИНФОРМАЦИИ С ОТКРЫТЫМ КЛЮЧОМ И ЭЛЕКТРОННОЙ ЦИФРОВОЙ ПОДПИСИ

Цель работы

Изучение алгоритмов защиты информации с открытым ключом, электронной цифровой подписи и защищенных протоколов передачи сообщений.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1. Алгоритм защиты информации с открытым ключом [1]

Одним из наиболее распространенных алгоритмов с открытым ключом является алгоритм RSA. Алгоритм RSA стал первым методом, который был бы пригоден и для шифрования, и для цифровой подписи. Его криптостойкость основывается на сложности разложения на множители больших чисел, а именно – на исключительной трудности задачи определить секретный ключ на основании открытого, так как для этого потребуется решить задачу о существовании делителей целого числа. Наиболее криптостойкие системы используют 1024-битовые и большие числа.

Вместо использования одного и того же ключа для шифрования и расшифрования данных в системе RSA используется согласованная пара ключей шифрования и расшифрования. Каждый из ключей выполняет одностороннее преобразование данных. Причем функция одного ключа обратна функции другого: действия одного ключа может отменить только другой ключ, входящий с ним в пару.

Владелец обеспечивает публичный доступ к открытому ключу RSA, в то время как закрытый ключ не разглашается. Для отправки личного сообщения автор шифрует его с помощью открытого ключа получателя. Зашифрованное таким образом сообщение может быть расшифровано только с помощью закрытого ключа получателя.

Алгоритмы криптосистемы с открытым ключом можно применять [2]: как независимый способ, создающий защиту передачи и хранения данных; как метод рассредоточения ключей (обычно применяют алгоритмы криптосистем с открытым ключом, которые распределяют ключи, достаточно малые по объёму, а транспортировку больших информационных потоков осуществляют с помощью других алгоритмов); как способ аутентификации пользователей.

Преимущества асимметричных шифров перед симметричными [2]:

– нет необходимости заранее пересылать закрытый ключ по защищенному каналу;

– только одной из сторон известен ключ шифрования, который держится в тайне (что является отличием от симметричной криптографии, где такой ключ должен быть известен обеим сторонам и должен держаться в секрете);

– пару E,D можно не менять довольно продолжительное время (при симметричном шифровании необходимо создавать новый ключ после каждого факта передачи);

– в развитых сетях количество ключей в асимметричной криптосистеме значительно меньше, чем в симметричной.

Недостатки алгоритма несимметричного шифрования в сравнении с симметричным [2]:

– в алгоритм труднее вписать изменения;

– хоть сообщения и надежно защищены и шифруются, но получатель и отправитель самим фактом пересылки шифрованного сообщения «засвечиваются»;

– более длинные ключи.

Шифрование-расшифровывание с применением двух ключей происходит на два-три порядка медленнее, чем шифрование-расшифровывание одного и того же текста симметричным алгоритмом.

Необходимы значительно большие вычислительные ресурсы, поэтому при работе с асимметричными криптосистемами используют и другие алгоритмы.

При утрате закрытого ключа RSA криптоаналитик приобретает возможность осуществить расшифровку всех записанных прошлых и будущих сообщений. Передача ключа в RSA является односторонней: все требуемые данные для создания симметричного ключа, который образуется на этапе рукопожатия, пересылаются на серверную сторону и шифруются соответствующим открытым ключом сервера. Выявление секретного ключа предоставляет возможность узнать симметричный ключ данной сессии.

DSA (Digital Signature Algorithm) – криптографический алгоритм с использованием открытого ключа для создания электронно-цифровой подписи, но не для шифрования. Подпись создается секретно, но может быть публично проверена. Это означает, что только один субъект может создать подпись сообщения, но любой может проверить её корректность. Алгоритм был предложен в 1991г.

Для подписывания сообщений необходима пара ключей – открытый и закрытый. При этом закрытый ключ должен быть известен

только тому, кто подписывает сообщения, а открытый – любому желающему проверить подлинность сообщения.

1.2. Электронная цифровая подпись

Электронная цифровая подпись (ЭЦП) – реквизит электронного документа, полученный в результате криптографического преобразования данных с использованием закрытого ключа подписи и позволяющий установить отсутствие искажения информации в электронном документе с момента формирования подписи и проверить принадлежность подписи владельцу сертификата ключа подписи [2]. Электронная подпись предназначена для идентификации лица, подписавшего электронный документ, и является полноценной заменой (аналогом) собственноручной подписи.

Применение электронной подписи позволяет реализовать следующие функции [2]:

– контроль единства и целостности передаваемого документа (при изменении документа подпись станет недействительной, потому что вычислена она на основе исходного состояния документа и соответствует лишь ему);

– защиту от изменений (подделки) документа;

– невозможность отказа от авторства;

– доказательное подтверждение авторства документа.

Используется несколько схем построения цифровой подписи [2].

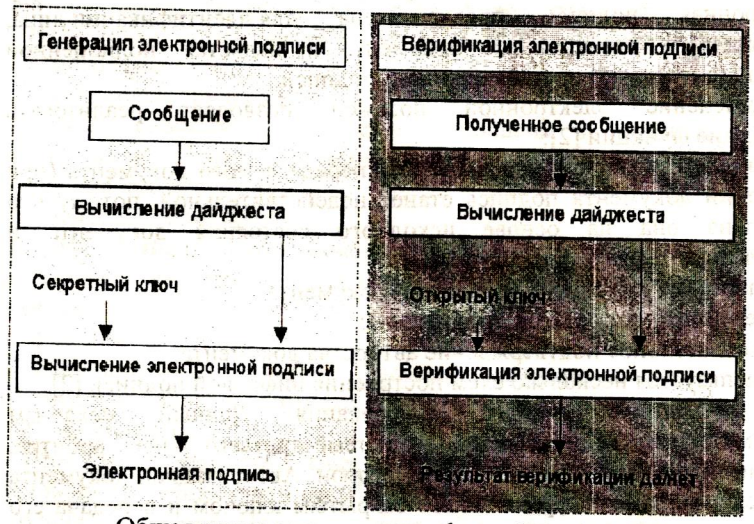
– Метод симметричного шифрования. Данный механизм предусматривает существование в системе третьего лица - арбитра, который пользуется доверием обеих сторон. Авторизацией документа является сам факт шифрования его закрытым ключом и передача его арбитру.

– Метод асимметричного шифрования. На данный момент такие механизмы ЭП широко распространены и находят различное применение.

Асимметричные схемы ЭП классифицируются к криптосистемам с открытым ключом. Их отличие от асимметричных алгоритмов шифрования, в которых шифрование производится с помощью открытого ключа, а расшифровка — с помощью закрытого, состоит в схемах цифровой подписи, где подписание производится с использованием закрытого ключа, а проверка подписи — с применением открытого. Схема электронно-цифровой подписи представлена на рисунке [2].

Закрытый ключ – наиболее слабый компонент всей криптосистемы цифровой подписи. Злоумышленник, перехвативший секретный ключ

пользователя, может сформировать идентичную цифровую подпись любого электронного документа от лица этого пользователя. Поэтому уделяют специальное внимание методу сохранения закрытого ключа. Пользователь может держать секретный ключ на личном персональном компьютере, защитив его с помощью пароля. Однако такой метод хранения обладает рядом недостатков, таких как защищенность ключа, которая вся зависит от безопасности компьютера, и пользователь может подписывать документы только на этом компьютере.



Общая схема электронно-цифровой подписи

1.3. Протоколы передачи

SSL — криптографический протокол, подразумевающий более защищенную коммуникацию. Он применяет симметричное шифрование для сохранения конфиденциальности, асимметричную криптографию для аутентификации ключей обмена, а также использует коды аутентификации сообщений для обеспечения целостности сообщений. Протокол имел широкое распространение и использовался для передачи мгновенных сообщений, голоса через IP (англ. Voice over IP — VoIP), а также в приложениях электронной почты, Интернет-факса и др. На сегодняшний день протокол больше не является безопасным. Чаще всего протокол SSL используется с самым распространенным протоколом передачи

гипертекста – http. О наличии защищенного соединения свидетельствует суффикс «s» – протокол будет называться https.

Протокол SSL применяется в тех случаях, когда необходимо обеспечить требуемый уровень защиты информации, которую пользователь передает серверу [2]. На определенные сайты, которые работают с электронными деньгами (банки, Интернет-магазины, биржи контента), передается секретная информация. Например, пароль, а также номер и серия паспорта, номер кредитной карты, пин-код и др. Именно такие данные представляют огромный интерес для злоумышленников, поэтому если вы примените для передачи незащищенный протокол http, то вашу информацию могут украсть и использовать в корыстных целях. Протокол SSL предполагает защищенную передачу данных, используя следующие элементы: аутентификацию и шифрование [1].

Протокол SSL обеспечивает "безопасный канал", имеющий три главных характеристики [2]:

– канал считается приватным. Криптография используется для всей информации после простой беседы, которая служит для определения секретного ключа;

– канал аутентифицирован. Серверная сторона всегда аутентифицируется, а клиентская делает это опционно;

– канал является надежным. Передача сообщений содержит проверку целостности информации.

Одним из преимуществ протокола SSL является то, что он не зависит от протокола прикладного уровня. Протоколы приложений (HTTP, FTP, TELNET и т.д.) могут без ошибок функционировать на протоколе SSL, т.е. SSL может сочетать метод криптографии и ключ сессии, а также аутентифицировать серверную сторону до того, как приложение успеет передать или принять первый байт сообщения.

Для образования ключей выделяют 4 основных алгоритма: RSA, Fixed Diffie-Hellman, Ephemenral Diffie-Hellman, Anonymous Diffie-Hellman.

1.4. Ключ Diffie-Hellman

Схема Fixed Diffie-Hellman применяет постоянный открытый ключ, прописанный в сертификате сервера. Это также предполагает, что при любом новом подключении клиент передает свою часть ключа. После обмена ключами создается новый симметричный ключ для обмена информацией в данной сессии. При выявлении закрытого ключа сервера криптоаналитик получит возможность расшифровать ранее записанные сообщения, а также все будущие, что становится

возможным из-за самой схемы ключа. Так как криптоаналитик знает закрытый ключ сервера, он сможет узнать симметричный ключ каждой сессии, и даже тот факт, что механизм создания ключа является . двусторонним, не поможет. Схема Anonymous Diffie-Hellman не дает гарантий секретности, так как информация транспортируется незашифрованной. Вариант, гарантирующий безопасность прошлых и будущих сообщений, — Ephemenral Diffie-Hellman. По сравнению с ранее рассмотренными алгоритмами разница заключается в том, что при любом созданном соединении между сервером и клиентом формируется одноразовый ключ. Таким образом, даже если криптоаналитик получит текущий закрытый ключ, он получит возможность расшифровать данную сессию, но не предыдущие или будущие сессии.

Протокол TLS - как и его предшественник SSL (Secure Sockets Layer — уровень защищённых сокетов) — криптографические протоколы, обеспечивающие защищённую передачу данных между узлами в сети Интернет. TLS и SSL применяют асимметричное шифрование для идентификации, симметричную криптографию для секретности и коды подлинности информации для обеспечения сохранения единства сообщений.

Данный протокол широко используется в приложениях, работающих с сетью Интернет, таких как веб-браузеры, работа с электронной почтой, обмен мгновенными сообщениями и IP-телефония (VoIP).

Так как большая часть протоколов, обеспечивающих связь, могут работать как вместе, так и раздельно TLS (или SSL), при настройке соединения необходимо точно показать серверу, требует ли пользователь использовать TLS. Это может быть установлено либо с использованием стандартизированного номера порта, который устанавливает соединение с применением TLS (например, для https такой порт имеет номер 443), либо с любым портом и особенной командой серверу со стороны пользователя на переход соединения на TLS с применением индивидуальных схем протокола (например, STARTTLS, использующихся в протоколах электронной почты). Как только пользовательская и серверная стороны согласились о применении TLS, им требуется создать надежное соединение. Это делается с помощью процедуры подтверждения связи. Во время этого процесса пользователь и сервер принимают соглашение относительно различных параметров, необходимых для установки безопасного соединения.

Протокол TLS содержит несколько мер защищенности [3]:

– защита от снижения варианта протокола к прошлой (менее надежной) версии или менее безопасному алгоритму криптографии;

– пронумерованные последовательные записи приложений и применение соответствующего номера в коде аутентификации посылок (MAC);

– использование ключа в идентификаторе сообщения (только обладатель ключа может проконтролировать код аутентификации посылки). Хэш-код идентификации сообщений (HMAC), используемый в большинстве шифров из набора шифров TLS. Сообщение, которым заканчивается подтверждение связи («Finished»), содержит в себе хэш всех сообщений, которыми обменялись стороны в процессе подтверждения связи.

Псевдослучайная функция разделяет входную информацию на две части и производит обработку каждой различной хэш-функции (MD5 и SHA-1), а после этого вычисляет XOR от обеих итоговых свёрток, чтобы сформировать код аутентификации посылки. Это обеспечит защищенность даже в случае уязвимости одной из хэш-функций.

Алгоритмы, используемые в TLS [3]:

– для обмена ключами и проверки их подлинности применяются комбинации алгоритмов: RSA (асимметричный шифр), Diffie-Hellman (безопасный обмен ключами), DSA (алгоритм цифровой подписи), ECDSA;

– для симметричного шифрования: RC4, IDEA, Triple DES, Camellia или AES;

– для хеш-функций: MD5, SHA, SHA-256/384.

Алгоритмы могут дополняться в зависимости от версии протокола. До последней версии протокола TLS 1.2 были доступны также следующие алгоритмы симметричного шифрования, но они были убраны как небезопасные: IDEA, DES.

2. ТЕХНИЧЕСКАЯ ЧАСТЬ

2.1. Генерация ключей в Openssl [4]

Для создания ключей RSA применяется оператор genrsa. Оператор genrsa формирует секретный ключ, имеющий длину bits в формате PEM, зашифровывает его одним из алгоритмов des(56 bit), des3(168 bit) или idea(128 bit). При подборе метода криптографии будет запрошен пароль для зашифровки создаваемого ключа. Команда –out показывает программе то, что результат необходимо привести в файл. Команда –rand покажет файлы, из которых считается информация для установки seed(зерна) генератора случайных чисел.

Синтаксис команды создания ключей RSA [1]:

Openssl genrsa [-out file] [-des] [-des3] [-idea] [-rand file] [bits]

Пример генерации 4096 битового секретного ключа RSA:

```
Openssl genrsa –out D:/keys/secretkey.pem –idea 4096
Generating RSA private key 4096 bit long modulus
.....................................................
.....................++
Enter pass phrase for D:/keys/secretkey.pem:
Verifuing – Enter pass phrase for D:/keys/secretkey.pem:
```

где

genrsa – команда, отвечающая за создание ключей RSA;

-out D:/keys/ - путь, куда сохранится секретный ключ RSA;

secretkey.pem – название секретного ключа RSA;

-idea – алгоритм шифрации ключа;

4096 – желаемая длина ключа.

После ввода команды необходимо ввести пароль в интерактивном режиме и подтвердить его. После этой операции закрытый ключ зашифровывается и регистрируется в файл в текстовом виде. В самом начале ключа вписывается метод шифрования.

Для создания открытого ключа используется команда openssl rsa.

Синтаксис команды openssl rsa [1]:

openssl rsa –in file [-out file] [-des] [-des3] [-idea] [-chek] [-pubout].

Команда openssl rsa может изменять пароль и метод шифрования секретного ключа, вызывается опциями –in и –out. Если применить опцию -pubout, то в указанный файл –out будет записан открытый ключ, вычисленный на основе –in секретного.

Пример создания открытого ключа на основании секретного:

Openssl rsa –in D:/keys/secretkey.pem –out D:/keys/publickey.pem –pubout.

Пример изменения пароля и алгоритма шифрования секретного ключа с idea на des3:

Openssl rsa –in D:/keys/secretkey.pem –out D:/keys/secretkey1.pem –des3.

Для создания ключей DSA используется программа openssl gendsa, она отличается от программы genrsa тем, что для ключей DSA нельзя прямо указать длину в битах и ключи DSA могут создаваться согласно некоторым параметрам, записанным в файл paramfile командой openssl dsaparam.

Пример команды dsaparam:

Openssl dsaparam [-rand file] [-C] [-genkey] [-out file] numbits.

где numbits – длина необходимого ключа; - C заставляет dsaparam вывести код на СИ для программной генерации DSA на основе необходимых параметров; опция – genkey говорит, что в выходной файл вместе с параметрами дополнительно записывается созданный секретный ключ DSA, но нельзя его сразу же зашифровать.

Поэтому удобней воспользоваться командой openssl gendsa, в которой вместо числа бит указывается файл параметров, созданный dsaparam.

Пример команды dsaparam:

Openssl dsaparam -C –genkey –out D:/keys/param 4096.

Синтаксис команды gendsa:

Openssl gendsa [-out file] [-rand] [-idea] [-des] [-des3] paramfile3.

Генерация ключа dsa:

Openssl gendsa –out D:/keys/dsakey.pem –idea D:/keys/param.

Для управления ключами dsa используется команда openssl dsa, которая аналогична команде openssl rsa.

Генераци открытого ключа dsa:

Openssl dsa –in D:/keys/dsakey.pem –out D:/keys/dsapublickey.pem –pubout.

2.2. Шифрование в Openssl алгоритмом RSA [1]

Для шифрации и дешифрации RSA алгоритмом используется команда rsautl.

Чтобы зашифровать информацию, применяется следующий синтаксис [4]:

Openssl rsautl –in file.txt –out file.cr –inkey pubkey.pem –pubin –encrypt.

Для дешифрации используется следующий синтаксис:

Openssl rsautl –in file.cr –out file.txt –inkey secretkey.pem –decrypr.

Пример шифрации алгоритмом RSA:

Openssl rsautl –in D:/text.txt –out D:/rsacode.cr –inkey D:/keys/publickey.pem –pubin –encrypt.

Пример дешифрации алгоритмом RSA:

Openssl rsautl –in D:/rsacode.cr –out D:/rsatext.txt –inkey D:/keys/secretkey.pem –decrypt.

2.3. Электронная цифровая подпись в Openssl [4]

В этой работе выполнение ЭЦП (электронная цифровая подпись) будет производиться с использованием дайджестов (криптографических контрольных сумм) и команды dgst. Команда dgst- одна из наиболее распространённых задач, выполняемых с помощью

Openssl. Для использования электронно-цифровой подписи необходимо создать дайджест и подписать его с помощью секретного ключа, чтобы быть уверенным, что дайджест никто не изменял, и проверить его.

Синтаксис команды dgst алгоритма Sha1:

Openssl dgst –sha1 filename,

где -sha1 – алгоритм хеширования; filename – имя файла, для которого необходимо посчитать контрольную сумму.

Рассмотрим практические методы подписи файлов с использованием ключей алгоритма DSA. Пусть необходимо подписать файл «text.txt» с использованием алгоритма SHA1.

Команда подписи файла text.txt алгоритмом sha1 с использованием ключей DSA:

Openssl dgst –sha1 –sign D:/keys/dsasecretkey.pem –out D:/text.txt.sha1 D:/text.txt,

где -sha1 –алгоритм контрольных сумм; -sign - путь к секретному ключу алгоритма DSA; -out – путь, куда сохранится подписанный дайджест; D:/text.txt – путь к файлу, по которому мы считаем контрольную сумму.

Для того чтобы проверить правильность дайджеста, нужен файл, на основе которого был получен дайджест, сам подписанный дайджест и открытый ключ того, кто подписывал дайджест.

Пример команды проверки дайджеста:

Openssl dgst –sha1 –verify D:/keys/dsapublickey.pem –signature D:/text.txtsha1 D:/text.txt,

где -sha1 – алгоритм хеширования; -verify - путь к открытому ключу алгоритма DSA; -signature – путь к подписанному дайджесту; D:/text.txt – путь к файлу, по которому мы расчитываем контрольную сумму.

Если файл text.txt будет изменён, то контрольная сумма не будет совпадать с контрольной суммой подписанного файла. Об этом программа выдаст сообщение.

3. ПРАКТИЧЕСКАЯ ЧАСТЬ

3.1. Генерация ключей

Создать секретный и открытый ключи RSA и DSA алгоритмов длиной 4096 бит с использованием одного из алгоритмов шифрования. Выбрать в соответствии с табл. 2.1 алгоритм шифрования.

Таблица 2.1. Алгоритм шифрования

№ бригады	RSA	DSA
1	Des3	Idea
2	Des	Des3
3	Idea	Des
4	Idea	Des3

3.2. Шифрование и дешифрация алгоритмом RSA

1. Зашифровать файл «text.txt» алгоритмом RSA с использованием открытого ключа RSA.

2. Дешифровать файл «text.txt» с использованием секретного ключа RSA.

3. Исследовать устойчивость к ошибкам. Убедиться в том, что файл был правильно дешифрован. Проверить файл на идентичность после шифрования.

4. Внести ошибку в зашифрованный файл, изменив один символ. Попытаться дешифровать файл.

3.3. Электронная цифровая подпись

Создать контрольную сумму файла «text.txt» с использованием алгоритма хеширования, подписать её, используя ключи RSA, DSA, и проверить подпись. Выбрать в соответствии с табл. 2.2 ключи и алгоритм хеширования.

Таблица 2.2. Алгоритм хеширования

№ бригады	1	2	3	4
Алгоритм хеширования	Sha1	Md5	Sha	Mdc2
Ключи шифрования	RSA	DSA	DSA	RSA

4. СОДЕРЖАНИЕ ОТЧЕТА

Отчет о лабораторной работе должен содержать:

– цель работы;

– структурную схему алгоритма шифрования;

– структурную схему алгоритма хеширования;

– полученные результаты.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назовите особенности алгоритмов защиты информации с открытым ключом. Какие преимущества и недостатки они имеют?
2. Что такое RSA и DSA? Какие ключи вам известны?
3. Для чего предназначена электронная цифровая подпись? Что позволяет осуществить использование ЭП?
4. Назовите и объясните схемы построения ЭП.
5. Что такое SSL? Объясните принципы работы SSL, какие существуют алгоритмы образования ключей?
6. Объясните отличия протокола TLS от SSL.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Бернет С., Пэйн С. Криптография. Официальное руководство RSA Security. – 2-е изд.: пер. с англ. –М.: БИНОМ, 2009. - 384 с.
2. Рассел Дж., Кох Р. Симметричные криптосистемы. –VSD, 2012. – 106 с.
3. Шангин В.Ф. Информационная безопасность компьютерных систем и сетей.- СПб.: Издательский Дом «ФОРУМ»: ИНФА-М, 2008. - 416 с.
4. Ященко В.В. Введение в криптографию: Новые математические дисциплины. - СПб. и др.: МЦНМО; Питер, 2001. - 288с.

СОДЕРЖАНИЕ