**Лабораторные работы**

**1. Шифрование файлов**

Реализуйте программное средство шифрования файлов. Обязательные функции: шифрование и дешифрование файлов на дисках с помощью алгоритма симметричного шифрования. Входные данные: тип криптографической операции (шифрование/дешифрование), имя входного и выходного файлов и пароль. При выполнении шифрования на выходе получается зашифрованный файл, при выполнении дешифрования – файл, идентичный исходному (если введен правильный пароль).

**2. Аутентификация сообщений**

Реализуйте аутентификацию сообщений с использованием криптографической хэш-функции. Дополните программное средство из лабораторной работы № 1 возможностью выработки и проверки имитовставки.

**3. Идентификация и аутентификция пользователей**

Реализуйте систему аутентификации на основе протокола Kerberos. Предусмотрите функцию отображения всех сообщений, передаваемых между субъектами взаимодействия в исходном и зашифрованном виде. Результатом успешной аутентификации является наличие общего сеансового ключа, сгенерированного сервером выдачи мандатов, у клиента и сервера ресурсов. Взаимодействие узлов системы должно осуществляться по сети.

**Kerberos**

Протокол Kerberos использует дополнительную аутентификацию на доверенном посреднике (рис. 1). Роль посредника здесь играет так называемый ***центр распределения ключей KDC*** (Key Distribution Center).



Рисунок 1 – Схема аутентификации Kerberos.

KDC представляет собой службу, работающую на физически защищенном сервере. Она ведет базу данных с информацией об учетных записях всех своих абонентов безопасности. Вместе с информацией о каждом из них в базе данных KDC сохраняется криптографический ключ, известный только этому абоненту и службе KDC. Данный ключ, который называют ***долговременным***, используется для связи пользователя системы безопасности с центром распределения ключей. В большинстве практических реализаций протокола Kerberos долговременные ключи создаются на основе пароля пользователя.

Путь клиента к ресурсу в системе Kerberos состоит из трех этапов:

1. Определение легальности клиента, осуществляющего вход в систему (сообщения 1 и 2 на рис. 1).
2. Получение разрешения на обращение к ресурсному серверу (сообщения 3 и 4).
3. Получение доступа к ресурсу (сообщения 5 и 6).

Для решения первой и второй задач клиент обращается к серверу KDC, который соответственно делится на две вспомогательные службы. Выполнение первичной аутентификации осуществляется так называемым ***аутентификационным сервером******AS***(Authentication Server)*.* Этот сервер хранит в своей базе данных информацию об идентификаторах и паролях пользователей. Вторую задачу, связанную с получением разрешения на обращение к ресурсному серверу, решает ***сервер мандатов TGS***(Ticket-Granting Server). Сервер TGS для легальных клиентов выполняет дополнительную проверку и дает клиенту разрешение на доступ к нужному ему ресурсному серверу, для чего наделяет его электронной формой-мандатом. Для выполнения своих функций сервер мандатов использует копии секретных ключей всех ресурсных серверов, которые хранятся у него в базе данных. Кроме этих ключей, сервер TGS имеет еще один секретный ключ, который разделяется с сервером AS.

Протокол Kerberos описывается следующими сообщениями:

1. Клиент-AS: *C*, *TGS*.
2. AS-клиент: {*KС*,*TGS*}*KС*, {*TС*,*TGS*}*KTGS*.
3. Клиент-TGS: *S*, {*AC*, *TGS*}*KС*,*TGS*, {*TС*,*TGS*}*KTGS*.
4. TGS-клиент: {*KC,S*}*KС,TGS*, {*TC,S*}*KS*.
5. Клиент-сервер: {*AC,S*}*KC,S*, {*TC,S*}*KS*.
6. Сервер-клиент: {*t* + 1}*KC,S*.

Здесь *С* обозначает клиента, *S* – сервер, *Kx* – секретный ключ *х*, *Kx,y* – сеансовый ключ для *х* и *у*, {*т*}*Kx* – сообщение *т,* шифрованное секретным ключом *х*, *Тx,y* – мандат (билет) *х* на использование *у*, *Ax,y* –удостоверение (аутентификатор) *х* для *у*, *t* – метка времени, входящая в состав удостоверения.

Мандат используется для безопасной передачи серверу личности клиента, которому выдан этот мандат. Мандат Kerberos имеет следующую форму:

*TС,S* = *S*, {*С*, *a*, *v*, *KС,S*}*KS*.

Мандат пригоден для одного сервера и одного клиента. Он содержит имя клиента *С*, его сетевой адрес *a*, имя сервера *S*, срок действия *v* и сеансовый ключ *KС,S*. Эта информация шифруется секретным ключом сервера *KS*. Если клиент получил мандат, он может использовать его для доступа к серверу много раз – пока не истечет срок действия мандата. Клиент не может расшифровать мандат (он не знает секретного ключа сервера), но он может предъявить его серверу в зашифрованной форме. Прочитать или изменить мандат при передаче его по сети невозможно.

Удостоверение – это дополнительный атрибут, предъявляемый вместе с мандатом, который преследует две цели. Во-первых, удостоверение содержит некоторый открытый текст, зашифрованный сеансовым ключом. Это доказывает, что клиенту известен ключ. Кроме того, зашифрованный открытый текст включает метку времени. Злоумышленник, которому удалось записать и мандат, и удостоверение, не сможет использовать их спустя два дня. Удостоверение Kerberos имеет следующую форму:

*AС,S* = {*С*, *t*, *ключ*}*КС,S*.

Клиент создает его каждый раз, когда ему нужно воспользоваться услугами сервера. Удостоверение содержит имя клиента *С*, метку времени *t* и необязательный дополнительный сеансовый ключ. Все эти данные шифруются сеансовым ключом *КС,S*, общим для клиента и сервера. В отличие от мандата, удостоверение используется только один раз. Однако это не проблема, так как клиент может генерировать удостоверения по мере надобности (ему известен общий секретный ключ).

Этапы аутентификации по протоколу Kerberos:

***1. Получение первоначального мандата***. У клиента есть часть информации, доказывающей его личность, – его пароль. Однако он не пересылается по сети. Клиент посылает в адрес сервера аутентификации сообщение 1, содержащее только его имя и имя его сервера TGS. В случае обнаружения данных о пользователе в базе данных сервер AS генерирует сеансовый ключ, который будет использоваться для обмена данными между клиентом и TGS, и шифрует этот сеансовый ключ секретным ключом клиента. Затем он создает мандат для доступа клиента к серверу TGS (***мандат на выделение мандата*** TGT, Ticket Granting Ticket) и шифрует его секретным ключом TGS. В сообщение 2 включаются зашифрованный сеансовый ключ и мандат. Легитимный клиент может расшифровывать только первую часть сообщения. Злоумышленник не может расшифровать ни одну часть сообщения 2.

***2. Получение серверных мандатов*.** Мандаты для получения доступа к конкретным серверам выдает центр распределения мандатов TGS при поступлении запросов от клиентов с корректными TGT и удостоверениями (сообщение 3). Сервер TGS, получив запрос, расшифровывает TGT своим секретным ключом. Затем TGS использует включенный в TGT сеансовый ключ, чтобы расшифровать удостоверение. Наконец, TGS сравнивает информацию удостоверения с информацией мандата, сетевой адрес клиента – с адресом отправителя запроса и метку времени – с текущим временем. Если все совпадает, TGS разрешает выполнение запроса. Проверка меток времени предполагает, что часы всех компьютеров синхронизированы с точностью, по крайней мере, до нескольких минут.

В ответ на правильный запрос TGS возвращает правильный мандат, который клиент может предъявить серверу. TGS также создает новый сеансовый ключ для клиента и сервера, зашифрованный сеансовым ключом, общим для клиента и TGS. Оба этих значения отправляются клиенту (сообщение 4). Клиент расшифровывает сообщение и извлекает сеансовый ключ.

***3. Запрос услуги.*** Теперь клиент может доказать свою подлинность серверу. Клиент создает удостоверение и вместе с полученным в TGS мандатом передает запрос на сервер ресурсов (сообщение 5). Сервер расшифровывает и проверяет мандат и удостоверение, а также проверяет адрес клиента и метку времени. Если все в порядке, то сервер уверен, что, согласно Kerberos, клиент – именно тот, за кого он себя выдает. Если приложение требует взаимной проверки подлинности, сервер посылает клиенту сообщение, состоящее из метки времени, зашифрованной сеансовым ключом (сообщение 6). Это доказывает, что серверу известен правильный секретный ключ и он может расшифровать мандат и удостоверение. При необходимости клиент и сервер могут шифровать дальнейшие сообщения общим ключом. Так как этот ключ известен только им, они оба могут быть уверены, что последнее сообщение, зашифрованное этим ключом, отправлено другой стороной.

**4. Стеганографические методы защиты мультимедиа информации**

1. Реализуйте стеганографическое внедрение сообщений с помощью метода LSB в контейнеры заданного формата (графический или аудио). Количество внедряемых в каждую позицию контейнера битов 1, 2 или 3 (в зависимости от варианта).
2. Реализуйте стеганографическую систему постановки и проверки цифрового водяного знака с помощью метода Patchwork. Параметры контейнера и метода в зависимости от варианта: контейнер графический или аудио, = 3, 4 или 5, *n* = 10000, 20000 или 30000.
3. Проведите атаки на заданные методы, например, путем замены младших битов случайными, трансформации контейнера (в случае изображения – поворот, масштабирование и др.), внесения шума, обмена местами соседних значений. Оцените устойчивость методов к таким атакам, в частности, посчитайте процент внедренной информации, сохранившейся в контейнере после атаки.

**Метод LSB**

Метод замены наименее значащих битов в байтах или словах мультимедийных контейнеров (Least Significant Bit, LSB) основан на том факте, что при оцифровке изображения или звука всегда присутствует погрешность дискретизации, равная наименьшему значащему разряду числа, определяющему величину цветовой составляющей элемента изображения или амплитуды звукового сигнала. Поэтому замена наименее значащих битов скрытым сообщением в большинстве случаев не вызывает значительной трансформации сигнала и не обнаруживается визуально или аудиально.

Рассмотрим использование данного метода на примере 24-битного растрового RGB-изображения. Одна точка изображения в этом формате кодируется тремя байтами, каждый из которых отвечает за интенсивность одного из трех составляющих цветов: красного (Red), зеленого (Green) и синего (Blue). Интенсивность каждой составляющей лежит в пределах от 0 до 255, то есть каждая составляющая имеет 256 оттенков. Младшие разряды в меньшей степени влияют на итоговое изображение, чем старшие. Из этого можно сделать вывод, что замена одного или двух младших, наименее значащих битов на другие произвольные биты настолько незначительно исказит оттенок пиксела, что зритель просто не заметит изменения. Максимальное количество возможных цветов для этого формата составляет более 16 миллионов. Однако следует иметь в виду, что глаз человека способен различать только около 4 тысяч цветов. Для кодирования этого количества цветов достаточно всего четырех битов . В случае необходимости можно занять и три разряда, что весьма незначительно скажется на качестве картинки.

Давайте подсчитаем полезный объем такого RGB-контейнера. Занимая два бита из восьми на каждый канал, мы будем иметь возможность спрятать три байта полезной информации на каждые четыре пиксела изображения, что соответствует 25 % объема картинки. Таким образом, имея файл изображения размером 200 Кбайт, мы можем скрыть в нем до 50 Кбайт произвольных данных так, что невооруженному глазу эти изменения не будут заметны.

**Метод Patchwork**

Данный метод используется для постановки водяных знаков. Он основан на внесении изменений в два участка изображения: на участке *А* яркость изображения незначительно увеличивается, а на участке *В* – уменьшается. Рассмотрим основную идею Patchwork на примере изображения, в котором для простоты примем, что все возможные значения яркости пикселей распределены равномерно в диапазоне от 0 до 255.

Выберем на изображении случайным образом две точки *А* и *В*, яркость в которых равна *а* и *b* соответственно. Теперь положим, что

*S* = *a* – *b*.

Среднее значение разницы *S* (обозначим его *MS*)после многократного повторения данной процедуры будет равно 0.

Теперь предположим, что описанная процедура повторяется *п* раз, полагая, что значения *а*, *b* и *S* на *i*-й итерации равны *ai*, *bi* и *Si* соответственно. Тогда *MS* выразится как



Учитывая приведенные выше рассуждения, общий алгоритм встраивания метки может быть представлен следующим образом:

1. Используя оговоренный заранее секретный ключ как начальное значение для криптостойкого генератора псевдослучайных чисел, сгенерировать координаты пары точек (*ai*, *bi*).
2. Увеличить яркость изображения в точке *ai* на значение *,* обычно выбираемое в диапазоне от 1 до 5 для изображения с 256 уровнями яркости.
3. Уменьшить яркость изображения в точке *bi* на значение *.*
4. Повторить шаги 1–3 *п* раз (*п* выбирается порядка 10 000).

Модифицированное значение ** может быть выражено как

.

Таким образом, с каждым новым шагом приведенного выше алгоритма накапливается отклонение на величину  (рис. 2).



Рисунок 2 – Сдвиг распределения *MS* после внедрения водяного знака

Наличие подобного отклонения от ожидаемого значения свидетельствует о наличии встроенной в изображение метки. Таким образом, владелец может доказать свои интеллектуальные права на изображение, предъявив секретный ключ, который использовался для встраивания метки в изображения.

**5.\*(только для претендующих на автомат)**

**Защита программного обеспечения от несанкционированного использования**

1. Внедрите динамический цифровой водяной знак в простейшую программу (например, в программу решения задачи перемножения матриц или умножения вектора на матрицу).
2. Проведите обфускацию программы, содержащей цифровой водяной знак.
3. Оцените устойчивость произведенных преобразований к атакам.
4. Подготовьте краткий отчет о проведенных преобразованиях и оценке устойчивости. Оформление произвольное.