Лекція 14. Алгоритмы обмена ключей и протоколы аутентификации.

Аутентификация

Сервис аутентификации предназначен для того, чтобы обеспечить надежную идентификацию подлинного источника информации. В случае единичного сообщения, например извещения или сигнала тревоги, задачей службы аутентификации является проверка и гарантия того. Что источником такого сообщения является именно тот объект, за который выдает себя отправитель. В случае внешнего интерактивного взаимодействия, например при подключении к главному узлу с терминала, можно выделить два аспекта функционирования таких средств. Во-первых, в момент подключения средства аутентификации должны гарантировать, что оба участвующих в сеансе связи объекта аутентичны (т.е. действительно являются теми, за кого себя выдают). Во-вторых, в ходе последующего обмена данными эти средства должны не допускать того, чтобы на поток данных могла повлиять какая-либо третья сторона путем имитации, т.е. представления себя одним из двух законных объектов с целью несанкционированной отправки или получения информации.

Идентификация и аутентификация

Основные понятия

Идентификацию и аутентификацию можно считать основой программно-технических средств безопасности, поскольку остальные сервисы рассчитаны на обслуживание именованных субъектов. Идентификация и аутентификация – это первая линия обороны, "проходная" информационного пространства организации.

Идентификация позволяет субъекту (пользователю, процессу, действующему от имени определенного пользователя, или иному аппаратно-программному компоненту) назвать себя (сообщить свое имя). Посредством аутентификации вторая сторона убеждается, что субъект действительно тот, за кого он себя выдает. В качестве синонима слова "аутентификация" иногда используют словосочетание "проверка подлинности".

Аутентификация бывает односторонней (обычно клиент доказывает свою подлинность серверу) и двусторонней (взаимной). Пример односторонней аутентификации – процедура входа пользователя в систему.

В сетевой среде, когда стороны идентификации/аутентификации территориально разнесены, у рассматриваемого сервиса есть два основных аспекта:

- что служит аутентификатором (то есть используется для подтверждения подлинности субъекта);
- как организован (и защищен) обмен данными идентификации/аутентификации.

Субъект может подтвердить свою подлинность, предъявив по крайней мере одну из следующих сущностей:

- нечто, что он знает (пароль, личный идентификационный номер, криптографический ключ и т.п.);
- нечто, чем он владеет (личную карточку или иное устройство аналогичного назначения);
- нечто, что есть часть его самого (голос, отпечатки пальцев и т.п., то есть свои биометрические характеристики).

В открытой сетевой среде между сторонами идентификации/аутентификации не существует доверенного маршрута; это значит, что в общем случае данные, переданные субъектом, могут не совпадать с данными, полученными и использованными для проверки подлинности. Необходимо обеспечить защиту от пассивного и активного прослушивания сети, то есть от перехвата, изменения и/или воспроизведения данных. Передача паролей в открытом виде, очевидно, неудовлетворительна; не спасает положение и шифрование паролей, так как оно не защищает от воспроизведения. Нужны более сложные протоколы аутентификации.

Надежная идентификация и затруднена не только из-за сетевых угроз, но и по целому ряду причин. Во-первых, почти все аутентификационные сущности можно узнать, украсть или подделать. Во-вторых, имеется противоречие между надежностью аутентификации, с одной стороны, и удобствами пользователя и системного администратора с другой. Так, из соображений безопасности необходимо с определенной частотой просить пользователя повторно вводить аутентификационную информацию (ведь на его место мог сесть другой человек), а это не только хлопотно, но и повышает вероятность того, что кто-то может подсмотреть за вводом данных. В-третьих, чем надежнее средство защиты, тем оно дороже.

Современные средства идентификации/аутентификации должны поддерживать концепцию единого входа в сеть. Единый вход в сеть – это, в первую очередь, требование удобства для пользователей. Если в корпоративной сети много информационных сервисов, допускающих независимое обращение, то многократная идентификация/аутентификация становится слишком обременительной. К сожалению, пока нельзя сказать, что единый вход в сеть стал нормой, доминирующие решения пока не сформировались.

Таким образом, необходимо искать компромисс между надежностью, доступностью по цене и удобством использования и администрирования средств идентификации и аутентификации.

Любопытно отметить, что сервис идентификации/аутентификации может стать объектом атак на доступность. Если система сконфигурирована так, что после определенного числа неудачных попыток устройство ввода идентификационной информации (такое, например, как терминал) блокируется, то

злоумышленник может остановить работу легального пользователя буквально несколькими нажатиями клавиш.

Парольная аутентификация

Главное достоинство парольной аутентификации – простота и привычность. Пароли давно встроены в операционные системы и иные сервисы. При правильном использовании пароли могут обеспечить приемлемый для многих организаций уровень безопасности. Тем не менее, по совокупности характеристик их следует признать самым слабым средством проверки подлинности.

Чтобы пароль был запоминающимся, его зачастую делают простым (имя подруги, название спортивной команды и т.п.). Однако простой пароль нетрудно угадать, особенно если знать пристрастия данного пользователя. Известна классическая история про советского разведчика Рихарда Зорге, объект внимания которого через слово говорил "карамба"; разумеется, этим же словом открывался сверхсекретный сейф.

Иногда пароли с самого начала не хранятся в тайне, так как имеют стандартные значения, указанные в документации, и далеко не всегда после установки системы производится их смена.

Ввод пароля можно подсмотреть. Иногда для подглядывания используются даже оптические приборы.

Пароли нередко сообщают коллегам, чтобы те могли, например, подменить на некоторое время владельца пароля. Теоретически в подобных случаях более правильно задействовать средства управления доступом, но на практике так никто не поступает; а тайна, которую знают двое, это уже не тайна.

Пароль можно угадать "методом грубой силы", используя, скажем, словарь. Если файл паролей зашифрован, но доступен для чтения, его можно скачать к себе на компьютер и попытаться подобрать пароль, запрограммировав полный перебор (предполагается, что алгоритм шифрования известен).

Тем не менее, следующие меры позволяют значительно повысить надежность парольной защиты:

- наложение технических ограничений (пароль должен быть не слишком коротким, он должен содержать буквы, цифры, знаки пунктуации и т.п.);
- управление сроком действия паролей, их периодическая смена;
- ограничение доступа к файлу паролей;
- ограничение числа неудачных попыток входа в систему (это затруднит применение "метода грубой силы");
- обучение пользователей;
- использование программных генераторов паролей (такая программа, основываясь на несложных правилах, может порождать только благозвучные и, следовательно, запоминающиеся пароли).

Перечисленные меры целесообразно применять всегда, даже если наряду с паролями используются другие методы аутентификации.

Одноразовые пароли

Рассмотренные выше **пароли можно назвать многоразовыми**; их раскрытие позволяет злоумышленнику действовать от имени легального пользователя. Гораздо более сильным средством, устойчивым к пассивному прослушиванию сети, являются **одноразовые пароли**.

Наиболее известным программным генератором одноразовых паролей является система **S/KEY** компании Bellcore. Идея этой системы состоит в следующем. Пусть имеется **односторонняя функция** f (то есть функция, вычислить обратную которой за приемлемое время не представляется возможным). Эта функция известна и пользователю, и **серверу аутентификации**. Пусть, далее, имеется **секретный ключ** K, известный только пользователю.

На этапе начального администрирования пользователя функция f применяется к ключу K n раз, после чего результат сохраняется на сервере. После этого процедура проверки подлинности пользователя выглядит следующим образом:

- сервер присылает на пользовательскую систему число ($n-1$);
- пользователь применяет функцию f к секретному ключу K ($n-1$) раз и отправляет результат по сети на сервер аутентификации;
- сервер применяет функцию f к полученному от пользователя значению и сравнивает результат с ранее сохраненной величиной. В случае совпадения подлинность пользователя считается установленной, сервер запоминает новое значение (присланное пользователем) и уменьшает на единицу счетчик (n).

На самом деле реализация устроена чуть сложнее (кроме счетчика, сервер посылает затравочное значение, используемое функцией f), но для нас сейчас это не важно. Поскольку функция f необратима, перехват пароля, равно как и получение доступа к серверу аутентификации, не позволяют узнать секретный ключ K и предсказать следующий одноразовый пароль.

Система S/KEY имеет статус Internet-стандарта (RFC 1938).

Другой подход к надежной аутентификации состоит в генерации нового пароля через небольшой промежуток времени (например, каждые 60 секунд), для чего могут использоваться программы или специальные интеллектуальные карты (с практической точки зрения такие пароли можно считать одноразовыми). Серверу аутентификации должен быть известен алгоритм генерации паролей и ассоциированные с ним параметры; кроме того, часы клиента и сервера должны быть синхронизированы.

Идентификация/аутентификация с помощью биометрических данных

Биометрия представляет собой совокупность автоматизированных методов идентификации и/или аутентификации людей на основе их физиологических и поведенческих характеристик. К числу физиологических характеристик принадлежат особенности **отпечатков пальцев**, **сетчатки** и **роговицы** глаз, **геометрия руки и лица** и т.п. К поведенческим характеристикам относятся **динамика подписи** (ручной), стиль **работы с клавиатурой**. На стыке физиологии и поведения находятся анализ особенностей **голоса** и **распознавание речи**.

Биометрией во всем мире занимаются очень давно, однако долгое время все, что было связано с ней, отличалось сложностью и дороговизной. В последнее время спрос на биометрические продукты, в первую очередь в связи с развитием электронной коммерции, постоянно и весьма интенсивно растет. Это понятно, поскольку с точки зрения пользователя гораздо удобнее предъявить себя самого, чем что-то запоминать. Спрос рождает предложение, и на рынке появились относительно недорогие аппаратно-программные продукты, ориентированные в основном на распознавание отпечатков пальцев.

В общем виде работа с биометрическими данными организована следующим образом. Сначала создается и поддерживается база данных характеристик потенциальных пользователей. Для этого биометрические характеристики пользователя снимаются, обрабатываются, и результат обработки (называемый **биометрическим шаблоном**) заносится в базу данных (исходные данные, такие как результат сканирования пальца или роговицы, обычно не хранятся).

В дальнейшем для идентификации (и одновременно аутентификации) пользователя процесс снятия и обработки повторяется, после чего производится поиск в базе данных шаблонов. В случае успешного поиска личность пользователя и ее подлинность считаются установленными. Для аутентификации достаточно произвести сравнение с одним биометрическим шаблоном, выбранным на основе предварительно введенных данных.

Обычно биометрию применяют вместе с другими аутентификаторами, такими, например, как интеллектуальные карты. Иногда биометрическая аутентификация является лишь первым рубежом защиты и служит для активизации интеллектуальных карт, хранящих криптографические секреты; в таком случае биометрический шаблон хранится на той же карте.

Активность в области биометрии очень велика. Организован соответствующий консорциум (см. http://www.biometrics.org/), активно ведутся работы по стандартизации разных аспектов технологии (формата обмена данными, прикладного программного интерфейса и т.п.), публикуется масса рекламных статей, в которых биометрия преподносится как средство обеспечения сверхбезопасности, ставшее доступным широким массам.

На наш взгляд, к биометрии следует относиться весьма осторожно. Необходимо учитывать, что она подвержена тем же угрозам, что и другие методы аутентификации. Во-первых, биометрический шаблон сравнивается не с результатом первоначальной обработки характеристик пользователя, а с тем, что пришло к месту сравнения. А, как известно, за время пути... много чего может произойти. Во-вторых, биометрические методы не более надежны, чем база данных шаблонов. В-третьих, следует учитывать разницу между применением биометрии на контролируемой территории, под бдительным оком охраны, и в "полевых" условиях, когда, например к устройству сканирования роговицы могут поднести муляж и т.п. В-четвертых, биометрические данные человека меняются, так что база шаблонов нуждается в сопровождении, что создает определенные проблемы и для пользователей, и для администраторов.

Но главная опасность состоит в том, что любая "пробоина" для биометрии оказывается фатальной. Пароли, при всей их ненадежности, в крайнем случае можно сменить. Утерянную аутентификационную карту можно аннулировать и завести новую. Палец же, глаз или голос сменить нельзя. Если биометрические данные окажутся скомпрометированы, придется как минимум производить существенную модернизацию всей системы.

Алгоритмы распределения ключей с использованием третьей доверенной стороны

Понятие мастер-ключа

При симметричном шифровании два участника, которые хотят обмениваться конфиденциальной информацией, должны иметь один и тот же ключ. Частота изменения ключа должна быть достаточно большой, чтобы у противника не хватило времени для полного перебора ключа. Следовательно, сила любой криптосистемы во многом зависит от технологии распределения ключа. Этот термин означает передачу ключа двум участникам, которые хотят обмениваться данными, таким способом, чтобы никто другой не мог ни подсмотреть, ни изменить этот ключ. Для двух участников **A** и **B** распределение ключа может быть выполнено одним из следующих способов.

1. Ключ может быть создан **A** и физически передан **B**.
2. Третья сторона может создать ключ и физически передать его **A** и **B**.

3. **A** и **B** имеют предварительно созданный и недолго используемый ключ, один участник может передать новый ключ другому, применив для шифрования старый ключ.
4. Если **A** и **B** каждый имеют безопасное соединение с третьим участником **C**, **C** может передать ключ по этому безопасному каналу **A** и **B**.

Первый и второй способы называются ручным распределением ключа. Это самые надежные способы распределения ключа, однако во многих случаях пользоваться ими неудобно и даже невозможно. В распределенной системе любой хост или сервер должен иметь возможность обмениваться конфиденциальной информацией со многими аутентифицированными хостами и серверами. Таким образом, каждый хост должен иметь набор ключей, поддерживаемый динамически. Проблема особенно актуальна в больших распределенных системах.

Количество требуемых ключей зависит от числа участников, которые должны взаимодействовать. Если выполняется шифрование на сетевом или IP-уровне, то ключ необходим для каждой пары хостов в сети. Таким образом, если есть N хостов, то необходимое число ключей $\left[N \cdot (N-1)\right]/2$. Если шифрование выполняется на прикладном уровне, то ключ нужен для каждой пары прикладных процессов, которых гораздо больше, чем хостов.

Третий способ распределения ключей может применяться на любом уровне стека протоколов, но если атакующий получает возможность доступа к одному ключу, то вся последовательность ключей будет раскрыта. Более того, все равно должно быть проведено первоначальное распространение большого количества ключей.

Поэтому в больших автоматизированных системах широко применяются различные варианты четвертого способа. В этой схеме предполагается существование так называемого центра распределения ключей (Key Destribution Center – KDC), который отвечает за распределение ключей для хостов, процессов и приложений. Каждый участник должен разделять уникальный ключ с KDC.

Использование центра распределения ключей основано на использовании иерархии ключей. Как минимум используется два типа ключей: мастер-ключи и ключи сессии.

Для обеспечения конфиденциальной связи между конечными системами используется временный ключ, называемый ключом сессии. Обычно ключ сессии используется для шифрования транспортного соединения и затем уничтожается. Каждый ключ сессии должен быть получен по сети из центра распределения ключей. Ключи сессии передаются в зашифрованном виде, используя мастер-ключ, который разделяется между центром распределения ключей и конечной системой.

Эти мастер-ключи также должны распределяться некоторым безопасным способом. Однако при этом существенно уменьшается количество ключей, требующих ручного распределения. Если существует N участников, которые хотят устанавливать соединения, то в каждый момент времени необходимо $\left[N \cdot (N-1)\right]/2$ ключей сессии. Но требуется только N мастер-ключей, по одному для каждого участника.

Время жизни ключа сессии как правило равно времени жизни самой сессии.

Чем чаще меняются ключи сессии, тем более безопасными они являются, так как противник имеет меньше времени для взламывания данного ключа сессии. С другой стороны, распределение ключей сессии задерживает начало любого обмена и загружает сеть. Политика безопасности должна сбалансировать эти условия для определения оптимального времени жизни конкретного ключа сессии.

Если соединение имеет долгое время жизни, то должна существовать возможность периодически менять ключ сессии.

Для протоколов, не поддерживающих соединение, таких как протокол, ориентированный на транзакции, нет явной инициализации или прерывания соединения. Следовательно, неясно, как часто надо менять ключ сессии. Большинство подходов основывается на использовании нового ключа сессии для каждого нового обмена. Наиболее часто применяется стратегия использования ключа сессии только для фиксированного периода времени или только для определенного количества транзакций.

Протоколы аутентификации

Рассмотрим основные протоколы, обеспечивающие как взаимную аутентификацию участников, так и аутентификацию только одного из участников.

1. Взаимная аутентификация

Данные протоколы применяются для взаимной аутентификации участников и для обмена ключом сессии.

Основной задачей таких протоколов является обеспечение конфиденциального распределения ключа сессии и гарантирование его своевременности, то есть протокол не должен допускать повторного использования старого ключа сессии. Для обеспечения конфиденциальности ключи сессии должны передаваться в зашифрованном виде. Вторая задача, обеспечение своевременности, важна, потому что существует угроза перехвата передаваемого сообщения и повторной его пересылки. Такие повторения в худшем случае могут позволять взломщику использовать скомпрометированный ключ сессии, при этом успешно подделываясь под другого участника. Успешное повторение может, как минимум, разорвать операцию аутентификации участников.

Такие повторы называются replay-атаками. Рассмотрим возможные примеры подобных replay-атак:

1. **Простое повторение**: противник просто копирует сообщение и повторяет его позднее.
2. **Повторение, которое можно зарегистрировать**: противник может воспроизвести сообщение с меткой даты/времени внутри реального окна времени.
3. **Повторение, которое не может быть определено**: эта ситуация может иметь место ввиду того, что оригинальное сообщение можно принудительно задержать, чтобы оно не достигло адресата, тогда адресат получит только воспроизведенное сообщение.
4. **Возвратное воспроизведение без модификации**: это воспроизведение сообщения, возвращаемое назад отправителю. Такой вид атаки оказывается возможным, когда используется традиционное шифрование и для отправителя не слишком просто отличить по содержимому посылаемые сообщения от получаемых.

Один из возможных подходов для предотвращения replay-атак мог бы состоять в присоединении последовательного номера (sequence number) к каждому сообщению, используемому в аутентификационном обмене. Новое сообщение принимается только тогда, когда его последовательный номер правильный. Трудность данного подхода состоит в том, что каждому участнику требуется поддерживать значения sequence number для каждого участника, с которым он взаимодействует в данный момент. Поэтому обычно sequence number не используются для аутентификации и обмена ключами. Вместо этого применяется один из следующих способов:

1. **Отметки времени**: участник **A** принимает сообщение как не устаревшее только в том случае, если оно содержит отметку времени, которая, по мнению **A**, соответствует текущему времени. Этот подход требует, чтобы часы всех участников были синхронизированы.
2. **Запрос/ответ**: участник **A** посылает в запросе к **B** случайное число (nonce – number only once) и проверяет, чтобы ответ от B содержал корректное значение этого nonce.

Считается, что подход с отметкой времени не следует использовать в приложениях, ориентированных на соединение, потому что это технически трудно, так как таким протоколам, кроме поддержки соединения, необходимо будет поддерживать синхронизацию часов различных процессоров. При этом возможный способ осуществления успешной атаки может возникнуть, если временно будет отсутствовать синхронизация часов одного из участников. В результате различной и непредсказуемой природы сетевых задержек распределенные часы не могут поддерживать точную синхронизацию. Следовательно, процедуры, основанные на любых отметках времени, должны допускать окно времени, достаточно большое для приспособления к сетевым задержкам, и достаточно маленькое для минимизации возможности атак.

С другой стороны, подход запрос/ответ не годится для приложений, не устанавливающих соединения, так как он требует предварительного рукопожатия перед началом передач, тем самым отвергая основное свойство транзакции без установления соединения. Для таких приложений доверие к некоторому безопасному серверу часов и постоянные попытки каждой из частей синхронизировать свои часы с этим сервером может быть оптимальным подходом.

1.1. Использование симметричного шифрования

Для обеспечения аутентификации и распределения ключа сессии часто используется двухуровневая иерархия ключей симметричного шифрования. В общих чертах эта стратегия включает использование доверенного центра распределения ключей (KDC). Каждый участник разделяет секретный ключ, называемый также мастер-ключом, с KDC. KDC отвечает за создание ключей, называемых ключами сессии, и за распределение этих ключей с использованием мастер-ключей. Ключи сессии применяются в течение короткого времени для шифрования только данной сессии между двумя участниками. Такой подход является общепринятым. Как пример его использования мы рассмотрим в 14-ой лекции систему Kerberos. Материал этой лекции будет полезен для лучшего пониманиямеханизма работы системы Kerberos.

Большинство алгоритмов распределения секретного ключа с использованием KDC, включает также возможность аутентификации участников.

Протокол Нидхэма и Шредера

Предполагается, что секретные мастер-ключи K_A и K_B разделяют соответственно **A** и **KDC** и **B** и **KDC**. Целью протокола является безопасное распределение ключа сессии K_S между **A** и **B**. Протокол представляет собой следующую последовательность шагов:

1. $A \rightarrow KDC: \quad ID_A \parallel ID_B \parallel N_1$
2. $KDC \rightarrow A: \quad E_{KA}\left[K_S \parallel ID_B \parallel N_1 \parallel E_{KB}\left[K_S \parallel ID_A\right]\right]$
3. $A \rightarrow B: \quad E_{KB}\left[K_S \parallel ID_A\right]$
4. $B \rightarrow A: \quad E_{KS}\left[N_2\right]$
5. $A \rightarrow B: \quad E_{KS}\left[f\left(N_2\right)\right]$

1. **A** запрашивает у **KDC** ключ сессии для установления защищенного соединения с **B**. Сообщение включает идентификацию **A** и **B** и уникальный идентификатор данной транзакции, который обозначен как N_1 и называется nonce. Nonce может быть временной меткой, счетчиком или случайным числом; минимальное требование состоит в том, чтобы он отличался для каждого запроса. Кроме того, для предотвращения подделки желательно, чтобы противнику было трудно предугадать nonce. Таким образом, случайное число является лучшим вариантом для nonce.
2. **KDC** отвечает сообщением, зашифрованным ключом K_A. Таким образом, только **A** может расшифровать сообщение, и **A** уверен, что оно получено от **KDC**, так как предполагается, что кроме **A** и **KDC** этот ключ не знает никто. Это сообщение включает следующие элементы, предназначенные для **A**:
   - Одноразовый ключ сессии K_S.
   - Идентификатор **B** ID_B.
   - nonce, который идентифицирует данную сессию N_1.

A должен убедиться, что полученный nonce равен значению nonce из первого запроса. Это доказывает, что ответ от **KDC** не был модифицирован при пересылке и не является повтором некоторого предыдущего запроса. Кроме того, сообщение включает два элемента, предназначенные для **B**:

- Одноразовый ключ сессии K_S.
- Идентификатор **A** ID_A.

Эти два последних элемента шифруются мастер-ключом, который **KDC** разделяет с **B**. Они посылаются **B** при установлении соединения и доказывают идентификацию **A**.

3. **A** сохраняет у себя ключ сессии и передает **B** информацию от **KDC**, предназначенную **B**: $E_{KB}\left[K_S \parallel ID_A\right]$. Так как эта информация зашифрована K_B, она защищена от просмотра. Теперь **B** знает ключ сессии (K_S), знает, что другим участником является **A**, (ID_A) и что начальная информация передана от ID_A, т.к. она зашифрована с использованием K_B.

В этой точке ключ сессии безопасно передан от **A** к **B**, и они могут начать безопасный обмен. Тем не менее, существует еще два дополнительных шага:

4. Используя созданный ключ сессии, **B** пересылает **A** nonce N_2.
5. Также используя K_S, **A** отвечает $f(N_2)$, где f – функция, выполняющая некоторую модификацию N_2.

Эти шаги гарантируют **B**, что сообщение, которое он получил, не изменено и не является повтором предыдущего сообщения.

Заметим, что реальное распределение ключа включает только шаги 1–3, а шаги 4 и 5, как и 3, выполняют функцию аутентификации.

A безопасно получает ключ сессии на шаге 2. Сообщение на шаге 3 может быть дешифровано только **B**. Шаг 4 отражает знание **B** ключа K_S, и шаг 5 гарантирует **B** знание участником **A** ключа K_S и подтверждает, что это не устаревшее сообщение, так как используется nonce N_2. Шаги 4 и 5 призваны предотвратить общий тип replay-атак. В частности, если противник имеет возможность захватить сообщение на шаге 3 и повторить его, то это должно привести к разрыву соединения.

Разрывая рукопожатие на шагах 4 и 5, протокол все еще уязвим для некоторых форм атак повторения. Предположим, что противник **X** имеет возможность скомпрометировать старый ключ сессии. Маловероятно, чтобы противник мог сделать больше, чем просто копировать сообщение шага 3. Потенциальный риск состоит в том, что **X** может заставить взаимодействовать **A** и **B**, используя старый ключ сессии. Для этого **X** просто повторяет сообщение шага 3, которое было перехвачено ранее и содержит скомпрометированный ключ сессии. Если **B** не запоминает идентификацию всех предыдущих ключей сессий с **A**, он не сможет определить, что это повтор. Далее **X** должен перехватить сообщение рукопожатия на шаге 4 и представиться **A** в ответе на шаге 5.

Протокол Деннинга

Деннинг предложил преодолеть эту слабость модификацией протокола Нидхэма и Шредера, которая включает дополнительную отметку времени на шагах 2 и 3:

1. $A \rightarrow KDC:\ \ ID_A \parallel ID_B$
2. $KDC \rightarrow A:\ \ E_{KA}\left[K_S \parallel ID_B \parallel T \parallel E_{KB}\left[K_S \parallel ID_A \parallel T\right]\right]$
3. $A \rightarrow B:\ \ \ \ E_{KB}\left[K_S \parallel ID_A \parallel T\right]$

4. $B \to A$: $E_{KS}[N_1]$
5. $A \to B$: $E_{KS}[f(N_1)]$

T – это отметка времени, которая гарантирует **A** и **B**, что ключ сессии является только что созданным. Таким образом, и **A**, и **B** знают, что распределенный ключ не является старым. **A** и **B** могут убедиться в соответствии времени, проверив неравенство

$$|Время - T| < \Delta t_1 + \Delta t_2 ,$$

где Δt_1 – оцениваемое нормальное расхождение между часами **KDC** и локальными часами (у **A** или **B**), а Δt_2 – ожидаемое время задержки в сети. Каждый участник может установить свои часы, ориентируясь на определенный доверенный источник. Поскольку временная отметка T шифруется с использованием секретных мастер-ключей, взломщик, даже зная старый ключ сессии, не сможет достигнуть цели повторением шага 3 так, чтобы **B** не заметил искажения времени.

Шаги 4 и 5 не были включены в первоначальное представление, но были добавлены позднее. Эти шаги подтверждают **A**, что **B** получил ключ сессии.

Протокол Деннинга обеспечивает большую степень безопасности по сравнению с протоколом Нидхэма и Шредера. Однако данная схема требует доверия к часам, которые должны быть синхронизированы в сети. В этом есть определенный риск, который состоит в том, что распределенные часы могут рассинхронизироваться в результате диверсии или повреждений. Проблема возникает, когда часы отправителя спешат по отношению к часам получателя. В этом случае противник может перехватить сообщение от отправителя и повторить его позднее, когда отметка времени в сообщении станет равной времени на узле получателя. Это повторение может иметь непредсказуемые последствия.

Один способ вычисления атак повторения состоит в требовании, чтобы участники регулярно сверяли свои часы с часами **KDC**. Другая альтернатива, при которой нет необходимости всем синхронизировать часы, состоит в доверии протоколам рукопожатия, использующим nonce.

Протокол аутентификации с использованием билета

Данный протокол пытается преодолеть проблемы, возникшие в предыдущих двух протоколах. Он выглядит следующим образом:

1. $A \to B$: $ID_A \| N_A$
2. $B \to KDC$: $ID_B \| N_B \| E_{KB}[ID_A \| N_A \| T_B]$
3. $KDC \to A$: $E_{KA}[ID_B \| N_A \| K_S \| T_B] \| E_{KB}[ID_A \| K_S \| T_B] \| N_B$
4. $A \to B$: $E_{KB}[ID_A \| K_S \| T_B] \| E_{KS}[N_B]$

1. **A** инициализирует аутентификационный обмен созданием nonce N_A и посылкой его и своего идентификатора к **B** в незашифрованном виде. Этот nonce вернется к **A** в зашифрованном сообщении, включающем ключ сессии, гарантируя **A**, что ключ сессии не старый.
2. **B** сообщает **KDC**, что необходим ключ сессии. Это сообщение к **KDC** включает идентификатор **B** и nonce N_B. Данный nonce вернется к **B** в зашифрованном сообщении, которое включает ключ сессии, гарантируя **B**, что ключ сессии не устарел. Сообщение **B** к **KDC** также включает блок, зашифрованный секретным ключом, разделяемым **B** и **KDC**. Этот блок используется для указания **KDC**, когда заканчивается время жизни данного ключа сессии. Блок также специфицирует намеченного получателя и содержит nonce, полученный от **A**. Этот блок является своего рода "верительной грамотой" или "билетом" для **A**.
3. **KDC** получил nonces от **A** и **B** и блок, зашифрованный секретным ключом, который **B** разделяет с **KDC**. Блок служит билетом, который может быть использован **A** для последующих аутентификаций. **KDC** также посылает **A** блок, зашифрованный секретным ключом, разделяемым **A** и **KDC**. Этот блок доказывает, что **B** получил начальное сообщение **A** (ID_B), что в нем содержится допустимая отметка времени и нет повтора (N_A). Этот блок обеспечивает **A** ключом сессии (K_S) и устанавливает ограничение времени на его использование (T_B).
4. **A** посылает полученный билет **B** вместе с nonce **B**, зашифрованным ключом сессии. Этот билет обеспечивает **B** ключом сессии, который тот использует для дешифрования и проверки nonce. Тот факт, что nonce **B** расшифрован ключом сессии, доказывает, что сообщение пришло от **A** и не является повтором.

Данный протокол аутентифицирует **A** и **B** и распределяет ключ сессии. Более того, протокол предоставляет в распоряжение **A** билет, который может использоваться для его последующей аутентификации, исключая необходимость повторных контактов с аутентификационным сервером. Предположим, что **A** и **B** установили сессию с использованием описанного выше протокола и затем

завершили эту сессию. Впоследствии, но до истечения лимита времени, установленного протоколом, **A** может создать новую сессию с **B**. Используется следующий протокол:

1. $A \rightarrow B$: $E_{KB}[ID_A \| K_S \| T_B] \| N'_A$
2. $B \rightarrow A$: $N'_B \| E_{KS}[N'_A]$
3. $A \rightarrow B$: $E_{KS}[N'_B]$

Когда **B** получает сообщение на шаге 1, он проверяет, что билет не просрочен. Заново созданные nonces N'_A и N'_B гарантируют каждому участнику, что не было атак повтора. Время T_B является временем относительно часов **B**. Таким образом, эта временная метка не требует синхронизации, потому что **B** проверяет только им самим созданные временные отметки.

1.2. Использование шифрования с открытым ключом.

В предыдущей лекции был представлен подход, позволяющий использовать схемы шифрования с открытым ключом для распределения сеансовых ключей. Соответствующий протокол предполагает, что в распоряжении каждой из двух сторон имеется текущий открытый ключ другой стороны. Но на практике требование выполнения этого условия может оказаться неудобным.

Протокол аутентификации с использованием аутентификационного сервера.

Рассмотрим протокол, использующий отметки времени и аутентификационный сервер:

1. $A \rightarrow AS$: $ID_A \| ID_B$
2. $AS \rightarrow A$: $E_{KRas}[ID_A \| KU_A \| T] \| E_{KRas}[ID_B \| KU_B \| T]$
3. $A \rightarrow B$: $E_{KRas}[ID_A \| KU_A \| T] \| E_{KRas}[ID_B \| KU_B \| T] \| E_{KUb}[E_{KRa}[K_S \| T]]$

В данном случае третья доверенная сторона является просто аутентификационным сервером **AS**, потому что третья сторона не создает и не распределяет секретный ключ. **AS** просто обеспечивает сертификацию открытых ключей участников. Ключ сессии выбирается и шифруется **A**, следовательно, не существует риска, что **AS** взломают и заставят распределять скомпрометированные ключи сессии. Отметки времени защищают от повтора скомпрометированных ключей сессии.

Данный протокол компактный, но, как и прежде, требует синхронизации часов.

Протокол аутентификации с использованием KDC

Другой подход использует nonces. Этот протокол состоит из следующих шагов:

1. $A \rightarrow KDC$: $ID_A \| ID_B$
2. $KDC \rightarrow A$: $E_{KRauth}[ID_B \| KU_B]$
3. $A \rightarrow B$: $E_{KUb}[N_A \| ID_A]$
4. $B \rightarrow KDC$: $ID_B \| ID_A \| E_{KUauth}[N_A]$
5. $KDC \rightarrow B$: $E_{KRauth}[ID_A \| KU_A] \| E_{KUb}[E_{KRauth}[N_A \| K_S \| ID_B]]$
6. $B \rightarrow A$: $E_{KUa}[E_{KRauth}[N_A \| K_S \| ID_B] \| N_B]$
7. $A \rightarrow B$: $E_{KS}[N_B]$

На первом шаге **A** информирует **KDC**, что хочет установить безопасное соединение с **B**. **KDC** возвращает **A** сертификат открытого ключа **B** (шаг 2). Используя открытый ключ **B**, **A** информирует **B** о создании защищенного соединения и посылает nonce N_A (шаг 3). На 4-м шаге **B** спрашивает **KDC** о сертификате открытого ключа **A** и запрашивает ключ сессии. **B** включает nonce **A**, чтобы **KDC** мог пометить ключ сессии этим nonce. Nonce защищен использованием открытого ключа **KDC**. На 5-м шаге **KDC** возвращает **B** сертификат открытого ключа **A** плюс информацию $\{N_A, K_S, ID_B\}$. Эта информация означает, что K_S является секретным ключом, созданным **KDC** в интересах **B** и связан с N_A. Связывание K_S и N_A гарантирует **A**, что K_S не устарел. Эта тройка шифруется с использованием закрытого ключа **KDC**, это гарантирует **B**, что тройка действительно получена от **KDC**. Она также шифруется с использованием открытого ключа **B**, чтобы никто другой не мог подсмотреть ключ сессии и использовать эту тройку для установления соединения с **A**. На шаге 6 тройка $\{N_A, K_S, ID_B\}$, зашифрованная закрытым ключом **KDC**, передается **A** вместе с nonce N_B, созданным **B**. Все сообщение шифруется открытым ключом **A**. **A** восстанавливает ключ сессии K_S, использует его для

шифрования N_B, который возвращает **B**. Это последнее сообщение гарантирует **B**, что **A** знает ключ сессии.

Описанный протокол кажется вполне защищенным в отношении атак самого разного вида, однако его авторы сами обнаружили в нем дефект и представили следующую исправленную версию алгоритма.

1. $A \to KDC: \ ID_A \| ID_B$
2. $KDC \to A: \ E_{KRauth}[ID_B \| KU_B]$
3. $A \to B: \ E_{KUb}[N_A \| ID_A]$
4. $B \to KDC: \ ID_B \| ID_A \| E_{KUauth}[N_A]$
5. $KDC \to B: \ E_{KRauth}[ID_A \| KU_A] \| E_{KUb}[E_{KRauth}[N_A \| K_S \| ID_A \| ID_B]]$
6. $B \to A: \ E_{KUa}[E_{KRauth}[N_A \| K_S \| ID_A \| ID_B] \| N_B]$
7. $A \to B: \ E_{KS}[N_B]$

Добавляется идентификатор **A** ID_A к данным, зашифрованным с использованием закрытого ключа **KDC** на шагах 5 и 6 для идентификации обоих участников сессии. Это включение ID_A приводит к тому, что значение nonce N_A должно быть уникальным только среди всех nonces, созданных **A**, но не среди nonces, созданных всеми участниками. Таким образом, пара $\{ID_A, N_A\}$ уникально идентифицирует соединение, созданное **A**.

Как в случае этого протокола, так и в случаях протоколов, описанных выше, первоначальные версии протоколов в дальнейшем подвергались ревизии и после дополнительного анализа появились их исправленные версии. Как видите, в области аутентификации весьма непросто добиться приемлемого уровня надежности с первой попытки.

2. Односторонняя аутентификация

Одним из приложений, для которых популярность использования шифрования постоянно растет, является электронная почта. Особенность e-mail состоит в том, что отправителю и получателю нет необходимости быть на связи в одно и то же время. Вместо этого сообщения направляются в почтовый ящик получателя, где они хранятся до тех пор, пока у того не появится возможность получить их.

Заголовок сообщения должен быть незашифрованным, чтобы сообщение могло пересылаться протоколами передачи, такими как X.400 или SMTP. Однако желательно, чтобы протоколы управления почтой не имели бы доступа к самому сообщению. Соответственно, e-mail сообщение должно быть зашифровано так, чтобы система управления почтой могла бы не знать ключ шифрования.

Вторым требованием является аутентификация сообщения. Обычно получателю нужна определенная гарантия того, что сообщение пришло от законного отправителя.

2.1. Использование симметричного шифрования

При использовании симметричного шифрования сценарий централизованного распределения ключей в полном объеме непригоден. Эти схемы требуют, чтобы в двух заключительных шагах отправитель посылал запрос получателю, ожидая ответа с созданным ключом сессии, и только после этого отправитель может послать сообщение.

С учетом перечисленных ограничений протоколы использования **KDC** являются возможными кандидатами для шифрования электронной почты. Для того чтобы избежать требования к получателю **B** находиться на связи в то же самое время, когда и отправитель **A**, шаги 4 и 5 должны быть опущены. Таким образом, остается последовательность шагов:

1. $A \to KDC: \ ID_A \| ID_B \| N_1$
2. $KDC \to A: \ E_{KA}[K_S \| ID_B \| N_1 \| E_{KB}[K_S \| ID_A]]$
3. $A \to B: \ E_{KB}[K_S \| ID_A] \| E_{KS}[M]$

Данный подход гарантирует, что только требуемый получатель сообщения сможет прочитать его. Это также обеспечивает определенный уровень аутентификации, что отправителем является **A**. Очевидно, что протокол не защищает от атак повтора. Некоторая мера защиты может быть обеспечена включением отметки времени в сообщение. Однако поскольку существуют потенциальные задержки в процессе передачи e-mail сообщений, такие временные отметки имеют ограниченный срок действия.

2.2. Использование шифрования с открытым ключом

При использовании шифрования с открытым ключом требуется, чтобы отправитель знал открытый ключ получателя (для обеспечения конфиденциальности), получатель знал открытый ключ отправителя

(для обеспечения аутентификации), или и то, и другое (для обеспечения конфиденциальности и аутентификации).

Если требуется конфиденциальность, то может быть использована следующая схема:

$$A \to B: \; E_{KUb}[K_S] \| E_{KS}[M].$$

В этом случае сообщение шифруется одноразовым секретным ключом. **A** шифрует этот одноразовый ключ открытым ключом **B**. Только **B** имеет соответствующий закрытый ключ для получения одноразового ключа и использования этого ключа для дешифрования сообщения. Эта схема более эффективна, чем простое шифрование всего сообщения открытым ключом **B**.

Если требуется аутентификация, то цифровая подпись может быть создана по такой схеме:

$$A \to B: \; M \| E_{KRa}[H(M)].$$

Этот метод гарантирует, что **A** не сможет впоследствии отвергнуть полученное сообщение. Однако данная технология открыта для другого типа подделок. Например, можно получить доступ к почтовой очереди перед доставкой, вырезать подпись отправителя, вставить свою и опять поставить сообщение в очередь на доставку.

Чтобы этого не допустить, сообщение и подпись можно зашифровать ключом симметричного шифрования, который в свою очередь шифруется открытым ключом получателя:

$$A \to B: \; E_{KUb}[K_S] \| E_{KS}[M \| E_{KRa}[H(M)]].$$

Следующая схема не требует, чтобы **B** знал открытый ключ **A**. В этом случае должен использоваться сертификат открытого ключа:

$$A \to B: \; M \| E_{KRa}[H(M)] \| E_{KRas}[T \| ID_A \| KU_A].$$

В конце сообщения **A** посылает **B** подпись, зашифрованную закрытым ключом **A**, и сертификат **A**, зашифрованный закрытым ключом аутентификационного сервера. Получатель применяет сертификат для получения открытого ключа отправителя и затем использует открытый ключ отправителя для проверки самого сообщения. Конфиденциальность может быть добавлена аналогично предыдущей схеме.