

# Kerberos

---

- Причины создания Kerberos
- Kerberos версии 4
  - Простой аутентификационный протокол
  - Более безопасный аутентификационный протокол
  - Аутентификационный протокол версии 4
- Kerberos версии 5
  - Различия между версиями 4 и 5
  - Аутентификационный протокол версии 5

# Причины создания Kerberos

- Kerberos является аутентификационным сервисом, разработанным как часть проекта Athena в MIT. В настоящее время Kerberos реализован также в Windows, начиная с 2000 Сервера. Kerberos решает следующую задачу: предположим, что существует открытое распределенное окружение, в котором пользователи, работающие за своими компьютерами, хотят получить доступ к распределенным в сети серверам. Серверы должны иметь возможность предоставлять доступ только авторизованным пользователям, т.е. аутентифицировать запросы на предоставление тех или иных услуг. В распределенном окружении рабочая станция не может быть доверенной системой, корректно идентифицирующей своих пользователей для доступа к сетевым серверам.

# Причины создания Kerberos

---

В частности, существуют следующие угрозы:

1. Пользователь может получить физический доступ к какой-либо рабочей станции и попытаться войти под чужим именем.
2. Пользователь может изменить сетевой адрес своей рабочей станции, чтобы запросы, посылаемые с неизвестной рабочей станции, приходили с известной рабочей станции.
3. Пользователь может просматривать трафик и использовать replay-атаку для получения доступа на сервер или разрыва соединения законных пользователей.

# Причины создания Kerberos

- Во всех этих случаях неавторизованный пользователь может получить доступ к сервисам и данным, не имея на то права. Для того чтобы не встраивать тщательно разработанные протоколы аутентификации на каждый сервер, Kerberos создает централизованный аутентифицирующий сервер, в чьи функции входит аутентификация пользователей для серверов и серверов для пользователей. В отличие от большинства схем аутентификации, Kerberos применяет исключительно симметричное шифрование, не используя шифрование с открытым ключом.
- Существует две версии Kerberos. Наиболее широко используется версия 4. Версия 5, в которой исправлены некоторые недостатки версии 4, описана в RFC 1510.
- Сначала кратко рассмотрим основной подход Kerberos. Затем будет описан аутентификационный протокол, используемый в версии 4. При этом мы рассмотрим суть подхода Kerberos, опуская некоторые детали, важные для устранения более тонких угроз безопасности. Затем рассмотрим версию 5.

# Причины создания Kerberos

- Если пользователи используют не соединенные в сеть компьютеры, то пользовательские и системные ресурсы и файлы можно уберечь от злоумышленников, обеспечив физическую защиту каждого компьютера. Когда пользователи обслуживаются централизованной системой разделения времени, безопасность должна обеспечивать именно она. Операционная система может проводить политику управления доступом на основе идентификатора пользователя и поддерживать строго определенную процедуру входа для идентификации и аутентификации пользователя.
- В условиях сетевого взаимодействия подобные сценарии неприемлемы. Наиболее общим случаем является распределенная архитектура, состоящая из рабочих станций пользователя (клиентов) и распределенных серверов. В подобном окружении могут использоваться три подхода к обеспечению безопасности:

# Причины создания Kerberos

1. Аутентификация пользователя на своей рабочей станции.
  2. Аутентификация пользователя на каждом сервере, к которому он должен иметь доступ.
  3. Аутентификация пользователя на специальном сервере, который выдает соответствующий билет (Ticket) для доступа на сервер.
- В небольших закрытых окружениях, в которых все системы работают в единственной организации, первой и, возможно, второй стратегии оказывается достаточно. Но в более открытом окружении, где поддерживаются сетевые соединения между компьютерами, для защиты информации и ресурсов пользователей необходим третий подход. Этот третий подход поддерживается Kerberos. Kerberos предполагает наличие распределенной архитектуры клиент/сервер и использует один или более серверов Kerberos для предоставления аутентификационных сервисов.

# Причины создания Kerberos

Kerberos должен удовлетворять следующим требованиям:

1. *Безопасность*: при просмотре трафика у оппонента не должно быть возможности получить необходимую информацию для того, чтобы сыграть роль законного пользователя.
2. *Надежность*: для всех сервисов, которые используют Kerberos для управления доступом, отсутствие сервера Kerberos означает отсутствие поддерживаемых сервисов. Следовательно, Kerberos должен иметь высокую надежность и реализовывать распределенную серверную архитектуру, в которой одна система может быть вспомогательной для другой.
3. *Прозрачность*: в идеале пользователь не должен знать, что имеет место аутентификация, кроме того случая, когда требуется ввести пароль.
4. *Масштабируемость*: система должна иметь возможность поддерживать большое число клиентов и серверов. Это требование означает, что Kerberos должен иметь модульную, распределенную архитектуру.

# Причины создания Kerberos

---

- Для реализации этих требований Kerberos использует трехсторонний аутентификационный диалог, основанный на протоколе Нидхэма и Шредера. Такая система является доверенной в том случае, если клиенты и серверы доверяют Kerberos быть посредником при аутентификации. Этот аутентификационный диалог безопасен в той же степени, в какой безопасен сам сервер Kerberos.

# Kerberos версии 4

---

- Версия 4 Kerberos в качестве алгоритма симметричного шифрования использует DES. Рассматривая протокол в целом, трудно оценить необходимость многих содержащихся в нем элементов. Поэтому сначала рассмотрим простейший вариант протокола, затем будем добавлять отдельные элементы для ликвидации конкретных уязвимостей.

# *Простой аутентификационный протокол*

---

В незащищенном сетевом окружении любой клиент может использовать любой сервер в качестве сервиса. В этом случае существует очевидный риск для системы безопасности. Оппонент может попытаться представиться другим клиентом и получить неавторизованные привилегии на сервере. Для того чтобы избежать этой опасности, сервер должен иметь возможность проверить идентификацию клиента, который запрашивает сервис. Практически не представляется возможным, чтобы каждый сервер выполнял эту задачу при соединении с каждым клиентом.

# Простой аутентификационный протокол

Альтернативой является использование аутентификационного сервера AS, который знает пароли всех пользователей и хранит их в специальной базе данных. Кроме того, AS разделяет уникальный секретный ключ с каждым сервером. Эти ключи распределяются физически или некоторым другим безопасным способом. Рассмотрим следующий протокол, который является скорее гипотетическим:

1.  $C \rightarrow AS$ :  $ID_C, P_C, ID_S$
2.  $AS \rightarrow C$ : Ticket
3.  $C \rightarrow S$ :  $ID_C, Ticket$

$$Ticket = E_{K_S} [ID_C, AD_C, ID_S]$$

Где

C	- клиент
AS	- аутентификационный сервер
S	- сервер
$ID_C$	- идентификатор пользователя на C
$ID_S$	- идентификатор S
$P_C$	- пароль пользователя на C
$AD_C$	- сетевой адрес C
$K_S$	- секретный ключ шифрования, разделяемый AS и S

# Простой аутентификационный протокол

- В данном сценарии предполагается, что пользователь входит на рабочую станцию и хочет получить доступ к серверу S. Клиентский модуль C на пользовательской рабочей станции запрашивает пользовательский пароль и затем посылает сообщение AS, которое включает идентификатор пользователя, идентификатор сервера и пароль пользователя. AS проверяет в своей базе данных правильность пароля пользователя и то, что данному пользователю разрешен доступ к серверу S. Если обе проверки выполнены успешно, AS считает, что пользователь аутентифицирован, и должен теперь убедить сервер, что это так. Для того, чтобы это сделать, AS создает билет (ticket), который содержит идентификатор пользователя, сетевой адрес, с которого вошел пользователь, и идентификатор сервера. Этот билет шифруется с использованием секретного ключа, разделяемого AS и S. Он посылается C. Так как билет зашифрован, его не может изменить ни C, ни оппонент.
- Имея данный билет, C может теперь обращаться к S за сервисом. Для этого он посылает серверу сообщение, содержащее идентификатор C и билет. S расшифровывает билет и проверяет, совпадают ли идентификатор пользователя в билете и незашифрованный идентификатор пользователя в сообщении. Если это соответствие выполняется, то сервер считает пользователя аутентифицированным и предоставляет соответствующий сервис.

## Простой аутентификационный протокол

Каждая часть сообщения (3) важна. Билет зашифрован для предотвращения изменения или подделки. Идентификатор сервера  $ID_S$  включается в билет, чтобы сервер мог убедиться, что он расшифровал билет корректно.  $ID_C$  включается в билет, чтобы определить, что данный билет послан от имени С. Наконец,  $AD_C$  служит для предотвращения следующей угрозы. Оппонент может перехватить билет, передаваемый в сообщении (2), затем использовать имя  $ID_C$  и передать сообщение в форме (3) с другой рабочей станции. Сервер получит законный билет, который соответствует пользователю ID, и предоставит доступ пользователю с другой рабочей станции. Для предотвращения подобной атаки AS включает в билет сетевой адрес, с которого приходит первоначальный запрос. Теперь билет действителен только в том случае, если он передан с той же самой рабочей станции, с которой первоначально запрашивался.

## Более безопасный аутентификационный протокол

- Хотя описанный сценарий и решает часть проблем аутентификации в открытых сетевых окружениях, многие проблемы все еще остаются. В частности, следует решить следующие две задачи. Во-первых, сделать так, чтобы пользователю приходилось вводить пароль минимальное количество раз. Пока предполагается, что каждый билет может использоваться только один раз. Если пользователь С хочет проверить свою почту на почтовом сервере, он должен предоставить пароль для получения билета на почтовый сервер. Если С хочет проверить почту несколько раз в течение дня, каждое обращение к почтовому серверу требует повторного ввода пароля. Эту процедуру можно усовершенствовать, разрешив переиспользовать билеты. При первой входной сессии рабочая станция может запомнить полученный билет сервера и использовать его от имени пользователя в дальнейшем при доступе к этому серверу.
- Однако при такой схеме пользователю необходим новый билет для каждого нового сервера. Если пользователь хочет получить доступ к серверу печати, почтовому серверу, файловому серверу и т.д., то при первом доступе к каждому серверу будет требоваться ввод пароля.

## Более безопасный аутентификационный протокол

---

- Вторая проблема состоит в том, что ранее рассмотренный сценарий включает незашифрованную передачу пароля в первом сообщении. Оппонент может перехватить пароль и использовать любой доступный данному пользователю сервис.
- Для решения этих проблем рассмотрим схему, которая не допускает незашифрованных паролей, и введем новый сервер, называемый сервером предоставления билетов (ticket-granting server – TGS). Этот сценарий состоит в следующем:

# Более безопасный аутентификационный протокол

## ■ Один раз при входе пользователя:

1.  $C \rightarrow AS: ID_C, ID_{TGS}$
2.  $AS \rightarrow C: E_{K_c} [Ticket_{TGS}]$

## ■ Один раз для каждого типа сервиса:

1.  $C \rightarrow TGS: ID_C, ID_S, Ticket_{TGS}$
2.  $TGS \rightarrow C: Ticket_S$

## ■ Один раз для каждого доступа к сервису:

1.  $C \rightarrow S: ID_C, Ticket_S$

$$Ticket_{TGS} = E_{K_{tgs}} [ID_C, AD_C, ID_{TGS}, TS_1, LT_1]$$

$$Ticket_S = E_{K_s} [ID_C, AD_C, ID_S, TS_2, LT_2]$$

# Более безопасный аутентификационный протокол

- Пользователь первым делом получает билет, гарантирующий билет,  $\text{Ticket}_{\text{tgs}}$  от AS. Этот билет хранится в модуле клиента на рабочей станции пользователя. Сервер TGS выдает билеты пользователям, которые перед этим были аутентифицированы AS. Каждый раз, когда пользователю требуется новый сервис, клиентский модуль обращается к TGS, используя предоставленный AS билет, и TGS выдает билет для конкретного сервиса. Клиентский модуль хранит каждый гарантирующий сервис билет и использует его для аутентификации на сервере всякий раз, когда требуется конкретный сервис. Рассмотрим данную схему подробнее:
- 1. Клиентский модуль запрашивает билет, гарантирующий билет, посылая идентификатор пользователя к AS вместе с идентификатором TGS, который будет в дальнейшем использоваться для получения билета, гарантирующего сервис.
- 2. AS в ответ присылает билет, зашифрованный ключом, полученным из пароля пользователя. Когда этот ответ поступает в клиентский модуль, он просит пользователя ввести свой пароль, создает ключ и пытается расшифровать полученное сообщение. Если используется корректный пароль, то билет успешно ~~2023~~влекается.

## Более безопасный аутентификационный протокол

Так как только законный пользователь должен знать пароль, только законный пользователь и может получить билет. Таким образом, пароль используется для получения доверительной грамоты от Kerberos без передачи пароля в незашифрованном виде. Сам билет включает идентификатор, сетевой адрес пользователя и идентификатор TGS. Это соответствует первому сценарию. Необходимо, чтобы такой билет мог использоваться клиентским модулем для запроса нескольких билетов, гарантирующих предоставление сервиса. Следовательно, билет, гарантирующий билет, с одной стороны, должен быть переиспользуемым. Но, с другой стороны, необходимо добиться того, чтобы оппонент не мог перехватывать этот билет и использовать его. Рассмотрим следующий сценарий: оппонент перехватывает билет и ждет до тех пор, пока пользователь не завершит регистрацию на своей рабочей станции. Тогда оппонент пытается получить доступ к этой рабочей станции или сконфигурировать свою рабочую станцию с тем же сетевым адресом, что и у законного пользователя. После этого оппонент будет иметь возможность переиспользовать билет для обмана TGS. Чтобы этого не произошло, билет включает отметку времени, определяющую дату и время, когда был получен билет, и время жизни, определяющую величину времени, в течение которого билет является действительным (например, 8 часов). Таким образом, теперь клиентский модуль имеет переиспользуемый билет, и нет необходимости требовать от пользователя ввода пароля для получения нового сервиса. В заключении заметим, что билет, гарантирующий билет, шифруется секретным ключом, известным только AS и TGS. Это предотвращает модификацию билета. Билет повторно зашифровывается ключом, основанным на пароле пользователя. Это гарантирует, что билет может быть восстановлен только законным пользователем, прошедшим аутентификацию.

# Более безопасный аутентификационный протокол

---

- Теперь, когда клиентский модуль имеет билет, гарантирующий билет, доступ к любому серверу можно получить, выполнив шаги (3) и (4):
- 3. Клиентский модуль запрашивает билет, гарантирующий сервис. Для этой цели клиентский модуль передает TGS сообщение, содержащее идентификатор пользователя, идентификатор требуемого сервиса и билет, гарантирующий билет.
- 4. TGS расшифровывает входящий билет и проверяет успешность дешифрования по наличию своего идентификатора. Также необходимо убедиться, что время жизни данного билета не истекло. Затем TGS сравнивает идентификатор пользователя и его сетевой адрес со значениями, полученными из билета. После этого TGS выдает билет, гарантирующий доступ к нужному сервису.

# Более безопасный аутентификационный протокол

- Билет, гарантирующий сервис, имеет ту же структуру, что и билет, гарантирующий билет. Этот билет также содержит отметку времени и время жизни. Если пользователь захочет получить доступ к тому же самому сервису позднее, клиентский модуль может просто использовать ранее полученный билет, гарантирующий сервис, и нет необходимости повторно запрашивать пароль пользователя. Заметим, что билет зашифрован с помощью секретного ключа  $E_{K_S}$ , известного только TGS и серверу, что предотвращает его изменение.
  - Наконец, с билетом, гарантирующим сервис, клиентский модуль может получить доступ к соответствующему сервису:
5. Клиентский модуль запрашивает доступ к сервису от имени пользователя. С этой целью клиентский модуль передает сообщение серверу, содержащее идентификатор пользователя и билет, гарантирующий сервис. Сервер аутентифицирует пользователя, используя содержимое билета.

# *Более безопасный аутентификационный протокол*

---

- Этот новый сценарий позволяет сделать так, чтобы за время работы пользователя пароль запрашивался только один раз, и обеспечивает защиту пароля пользователя.

## Аутентификационный протокол версии 4

- Хотя рассмотренный сценарий усиливает безопасность по сравнению с первым сценарием, остаются еще две проблемы. Первая проблема состоит в том, что время жизни связано с билетом, гарантирующим билет. Если это время жизни очень короткое (т.е. минуты), то пароль у пользователя будет запрашиваться повторно. Если время жизни большое (т.е. часы), то оппонент имеет больше возможностей для совершения различных replay-атак. Злоумышленник должен просматривать сеть и перехватить билет, гарантирующий билет, и затем ждать, когда законный пользователь выйдет. Затем оппонент может подделать сетевой адрес законного пользователя и послать сообщение шага (3) к TGS. Это откроет ему неограниченный доступ к ресурсам и файлам законного пользователя.

## Аутентификационный протокол версии 4

---

- Аналогично, если оппонент перехватил билет, гарантирующий сервис, и использует его прежде, чем истечет время его действия, он имеет доступ к соответствующему сервису.
- Таким образом, можно сформулировать следующее дополнительное требование. Сетевой сервис (т.е. TGS или прикладной сервис) должны иметь возможность убедиться в том, что билет использует тот, кто получил его.
- Вторая проблема состоит в том, что должна быть возможность аутентификации самих серверов для пользователей. Без подобной аутентификации оппонент может изменить конфигурацию таким образом, чтобы сообщение к серверу перенаправлялось по другому адресу. Этот ложный сервер будет затем выполнять действия в качестве реального сервера, перехватывать любую информацию от пользователя или не предоставлять ему необходимый сервис.

## Аутентификационный протокол версии 4

- Сначала рассмотрим проблему перехвата билетов, гарантирующих билеты, и необходимость гарантировать, что представленный билет является тем же самым билетом, который был выдан клиенту. Для решения этой проблемы AS должен безопасным способом обеспечить как клиентский модуль, так и TGS некоторой секретной информацией. После этого клиентский модуль может доказать TGS свою идентичность, предоставляя безопасным способом эту секретную информацию. Здесь может использоваться некий разделяемый секрет, который в дальнейшем будет применяться в качестве ключа шифрования. Этот разделяемый секрет называется ключом сессии.
- Рассмотрим технологию распределения ключа сессии.

# Аутентификационный протокол версии 4

**Обмен с аутентификационным сервисом для получения билета, гарантирующего билет**

**1.  $C \rightarrow AS$ :**  $ID_C, ID_{TGS}, TS_1$  - клиентский модуль запрашивает билет, гарантирующий билет

$ID_C$ : идентификатор пользователя.

$ID_{TGS}$ : идентификатор TGS.

$TS_1$ : отметка времени, которая позволяет AS проверить, синхронизированы ли часы клиента с часами AS.

## Аутентификационный протокол версии 4

2.  $AS \rightarrow C$ :  $E_{K_C} [K_{C,TGS}, ID_{TGS}, TS_2, LT_2, Ticket_{TGS}]$  - AS возвращает билет, гарантирующий билет

$$Ticket_{TGS} = E_{K_{AS,TGS}} [K_{C,TGS}, ID_C, AD_C, ID_{TGS}, TS_2, LT_2]$$

$Ticket_{TGS}$ : билет, используемый клиентом для доступа к TGS.

$K_C$ : ключ шифрования, основанный на пользовательском пароле, применение которого позволяет AS и клиентскому модулю аутентифицировать пользователя и защитить содержимое сообщения (2).

$K_{C,TGS}$ : ключ сессии, созданный AS для обеспечения безопасного обмена между клиентским модулем и TGS.

$K_{AS,TGS}$ : ключ, которым зашифрован билет и который известен только AS и TGS.

$ID_{TGS}$ : идентификатор TGS, подтверждение того, что данный билет предназначен для TGS.

$AD_C$ : адрес пользователя, предотвращает использование билета с любой рабочей станции, кроме той, с которой он был первоначально запрошен.

$TS_2$ : время создания билета.

$LT_2$ : время жизни билета.

# Аутентификационный протокол версии 4

Обмен с сервисом, гарантирующим билет, для получения билета, гарантирующего сервис

3.  $C \rightarrow TGS$ :  $ID_S$ ,  $Ticket_{TGS}$ ,  $Authenticator_C$  - клиентский модуль запрашивает билет, гарантирующий сервис

$ID_S$ : идентификатор сервера S.

$Ticket_{TGS}$ : гарантирует TGS, что данный пользователь аутентифицирован AS.

$Authenticator_C$ : создается клиентом для подтверждения законности ключа.

$$Authenticator_C = E_{Kc, tgs} [ID_C, AD_C, TS_3]$$

$TS_3$ : время создания аутентификатора.

## Аутентификационный протокол версии 4

4. **TGS  $\rightarrow$  C:**  $E_{K_{C,tgs}} [K_{C,S}, ID_S, TS_4, LT_4, Ticket_S]$  - TGS возвращает билет, гарантирующий сервис

$$Ticket_S = E_{K_{tgs,s}} [K_{C,S}, ID_C, AD_C, ID_S, TS_4, LT_4]$$

**Ticket<sub>S</sub>:** билет, используемый клиентом для доступа к серверу S.

**K<sub>tgs,s</sub>:** ключ, разделяемый S и TGS.

**K<sub>C,S</sub>:** ключ сессии, который создается TGS для обеспечения безопасного обмена между клиентским модулем и сервером без необходимости разделения ими постоянного ключа.

**ID<sub>S</sub>:** включено в качестве доказательства того, что этот ключ предназначен для сервера S.

**TS<sub>4</sub>:** время создания билета.

**LT<sub>4</sub>:** время жизни билета.

# Аутентификационный протокол версии 4

Клиент/серверный аутентификационный обмен для получения сервиса

5.  $C \rightarrow S$ :  $\text{Ticket}_S, \text{Authenticator}_C$  - клиент запрашивает сервис

$$\text{Ticket}_S = E_{K_{ts,s}} [K_{C,s}, ID_C, AD_C, ID_S, TS_4, LT_4]$$

$$\text{Authenticator}_C = E_{K_{c,s}} [ID_C, AD_C, TS_5]$$

$\text{Ticket}_S$ : гарантирует серверу, что данный клиент аутентифицирован AS.

$\text{Authenticator}_C$ : создается клиентом для подтверждения законности ключа.

$TS_5$ : время создания аутентификатора.

6.  $S \rightarrow C$ :  $E_{K_{c,s}} [TS_5 + 1]$  - дополнительная аутентификация сервера для клиента, гарантирует C, что сообщение получено от S.

■  $TS_5 + 1$ : гарантирует C, что не было replay-атак.

## Аутентификационный протокол версии 4

- Прежде всего, клиентский модуль посылает сообщение к AS с требованием доступа к TGS. AS отвечает сообщением, зашифрованным ключом, полученным из пароля пользователя  $K_C$ , которое содержит билет. Зашифрованное сообщение также содержит ключ сессии  $K_{C, tgs}$ , где индексы определяют, что это ключ сессии для C и TGS. Таким образом, ключ сессии безопасно передан как C, так и TGS.
- К первой фазе сценария добавлено несколько дополнительных элементов информации. Сообщение (1) включает отметку времени, так что AS знает, что сообщение своевременно. Сообщение (2) включает несколько элементов билета в форме, доступной C. Это необходимо C для подтверждения того, что данный билет предназначен для TGS и для определения момента истечения срока его действия.

## Аутентификационный протокол версии 4

- Теперь, имея билет и ключ сессии, С может обратиться к TGS. Как и раньше, С посылает TGS сообщение, которое включает билет и идентификатор требуемого сервиса (сообщение (3)). Дополнительно С передает аутентификатор, который включает идентификатор и адрес пользователя С, а также отметку времени. В отличие от билета, который является переиспользуемым, аутентификатор применяется только один раз и не имеет времени жизни (LT). Теперь TGS расшифровывает билет с помощью ключа, который он разделяет с AS. Этот билет содержит ключ сессии  $K_{C,tgs}$ . В действительности билет устанавливает, что любой, кто использует  $K_{C,tgs}$ , должен быть С. TGS задействует ключ сессии для дешифрования аутентификатора. TGS может затем сравнить имя и адрес из аутентификатора с тем, которое содержится в билете, и с сетевым адресом входящего сообщения. Если все совпадает, то TGS может быть уверен, что отправитель билета является настоящим его собственником. В действительности аутентификатор устанавливает, что до времени  $TS_3$  возможно использование  $K_{C,tgs}$ . Заметим, что билет не доказывает чью-либо идентичность, а является способом безопасного распределения ключей. Аутентификатор является доказательством идентификации клиента. Так как аутентификатор может быть использован только один раз, опасности, что оппонент украдет аутентификатор для пересылки его позднее, не существует.

## Аутентификационный протокол версии 4

- Сообщение (4) от TGS имеет вид сообщения (2). Сообщение зашифровано ключом сообщения, разделяемым TGS и C, и включает ключ сессии, который разделяется C и сервером S, идентификатор S и отметку времени билета. Билет включает тот же самый ключ сессии.
- C теперь имеет переиспользуемый билет, гарантирующий сервис S. Когда C представляет этот билет, как и в сообщении (5), он также посылает аутентификатор. Сервер может расшифровать билет, получить ключ сессии и расшифровать аутентификатор.
- Если требуется взаимная аутентификация, сервер посылает сообщение (6), в котором возвращает значение отметки времени из аутентификатора, увеличенное на единицу и зашифрованное ключом сессии. C может расшифровать это сообщение для получения увеличенной отметки времени. Так как сообщение может быть расшифровано ключом сессии, C уверен, что оно может быть создано только S. Это гарантирует C, что replay-атаки не было.

# Аутентификационный протокол версии 4

---

- Наконец, в завершение клиент и сервер разделяют секретный ключ. Этот ключ может быть использован для шифрования будущих сообщений между ними или для обмена новым случайным ключом сессии.

# Области Kerberos

- Полнофункциональное окружение Kerberos, состоящее из сервера Kerberos, некоторого числа клиентов и некоторого числа прикладных серверов, требует следующего:
  1. Сервер Kerberos должен иметь в своей базе данных идентификаторы и хэшированные пароли всех зарегистрированных пользователей, т.е. все пользователи регистрируются на сервере Kerberos.
  2. Сервер Kerberos должен разделять секретный ключ с каждым сервером, т.е. все серверы регистрируются на сервере Kerberos.
- Такое окружение называется **областью (realm)**. Сети из клиентов и серверов в различных административных организациях обычно образуют различные области. Однако пользователи из одной области могут нуждаться в доступе к серверам из других областей, и некоторые серверы готовы предоставлять сервисы пользователям из других областей, если эти пользователи будут аутентифицированы.

# Области Kerberos

- Kerberos предоставляет механизм для поддержки такой аутентификации между областями. Для двух областей, поддерживающих межобластную аутентификацию, добавлено следующее требование:
  1. Сервер Kerberos для каждой из взаимодействующих областей разделяет секретный ключ с сервером Kerberos в другой области. Другими словами, два сервера Kerberos регистрируют друг друга.
- Схема требует, чтобы сервер Kerberos в одной области доверял серверу Kerberos в другой области аутентифицировать своих пользователей. Более того, серверы во второй области также должны быть согласны доверять серверу Kerberos в первой области.
- Если эти основополагающие условия выполняются, можно ввести следующий механизм доступа пользователей к серверам из других областей. Пользователю, который хочет получить сервис на сервере из другой области, необходим билет для этого сервера. Клиентский модуль следует обычным процедурам получения доступа к локальному TGS и затем запрашивает билет, гарантирующий билет, для удаленного TGS (TGS в другой области). Затем клиентский модуль вызывает удаленный TGS для получения билета, гарантирующего сервис, на требуемый сервер в области, которую обслуживает удаленный TGS.

## Области Kerberos

1.  $C \rightarrow AS$ :  $ID_C, ID_{TGS}, TS_1$
2.  $AS \rightarrow C$ :  $E_{K_c} [K_{C, tgs}, ID_{TGS}, TS_2, LT_2, Ticket_{TGS}]$
3.  $C \rightarrow TGS$ :  $ID_{TGSrem}, Ticket_{TGS}, Authenticator_C$
4.  $TGS \rightarrow C$ :  $E_{K_{c, tgs}} [K_{C, TGSrem}, ID_{TGSrem}, TS_4, Ticket_{TGSrem}]$
5.  $C \rightarrow TGS_{rem}$ :  $ID_{rem}, Ticket_{TGSrem}, Authenticator_C$
6.  $TGS_{rem} \rightarrow C$ :  $E_{K_{c, TGSrem}} [K_{C, Srem}, ID_{Srem}, TS_6, Ticket_{Srem}]$
7.  $C \rightarrow S_{rem}$ :  $Ticket_{Srem} || Authenticator_C$

## Области Kerberos

- Билет, предоставляемый удаленному серверу  $S_{rem}$ , кроме всего остального, содержит идентификатор области, в которой пользователь был аутентифицирован. Сервер определяет, является ли данный удаленный запрос законным.
- При таком подходе традиционная проблема состоит в том, что если существует  $N$  областей, то должно быть  $[N(N - 1)]/2$  безопасных обменов ключей, чтобы каждый Kerberos области мог взаимодействовать со всеми остальными Kerberos.

# *Kerberos версии 5*

---

- Версия 5 Kerberos описана в RFC 1510 и предоставляет ряд улучшений по сравнению с версией 4. Сначала сделаем общий обзор изменений в версии 5 относительно версии 4, а затем рассмотрим протокол версии 5.

## Различия между версиями 4 и 5

- Версия 5 предназначена для преодоления недостатков проектирования и технических недоработок версии 4.
- Версия 4 Kerberos была разработана для использования в окружении проекта Athena и, следовательно, не предназначалась для использования в общих целях. Это привело к следующим **недостаткам проектирования**:
  1. *Зависимая система шифрования*: версия 4 требует использования DES. Сначала существовали экспортные ограничения DES, в настоящий момент основным недостатком является сомнение в силе DES. В версии 5 зашифрованный текст помечен типом шифрования, что позволяет задействовать любой алгоритм симметричного шифрования. Ключ шифрования помечен типом и длиной, что также позволяет применять различные алгоритмы.

## Различия между версиями 4 и 5

2. *Зависимость от интернет-протоколов:* версия 4 требует использования IP-адресации. Другие типы адресов, такие как сетевые адреса ISO, не поддерживаются. В версии 5 сетевой адрес помечен типом и длиной, что позволяет задействовать любой тип сетевого адреса.
3. *Упорядочивание байтов сообщения:* в версии 4 отправитель сообщения использует упорядочивание байтов по своему выбору и помечает сообщение, чтобы определить, левый или правый байт расположен в младшем адресе. В версии 5 структура всех сообщений определяется на основании ASN.1 и DER, что обеспечивает однозначную последовательность байтов.

## Различия между версиями 4 и 5

4. *Время жизни билета*: в версии 4 значения времени жизни хранятся в 8-битовых блоках по 5 минут. Таким образом, максимальное время жизни, которое может быть получено, есть  $2^8 * 5 = 1280$  минут или меньше 21 часа. Для некоторых приложений этого может быть недостаточно. В версии 5 билет включает явное время начала и время конца, допуская произвольное время жизни билета.
5. *Перенаправление аутентификации*: версия 4 не позволяет, чтобы доверительные грамоты, полученные для одного клиента, были перенаправлены другому хосту и использовались другим клиентом. Эта особенность не дает клиенту возможность получить доступ к серверу, чтобы затем сервер получил доступ к другому серверу от имени данного клиента. Например, клиент сделал запрос на сервер печати, после чего необходим доступ к файл-серверу за файлом клиента, с помощью доверительной грамоты клиента. Версия 5 обеспечивает такую возможность.

## Различия между версиями 4 и 5

6. *Аутентификация между областями:* в версии 4 взаимосвязь  $N$  областей требует последовательности из  $N^2$  взаимосвязей Kerberos-to-Kerberos. Версия 5 поддерживает метод, который требует меньшего числа взаимосвязей.

Существуют также следующие **технические недостатки** в самом протоколе версии 4:

1. *Двойное шифрование:* сообщения 2 и 4, которые поставляют клиентам билеты, зашифрованы дважды, один раз секретным ключом сервера назначения, а затем опять секретным ключом, известным клиенту. Второе шифрование не является обязательным и вычислительно расточительно.

## Различия между версиями 4 и 5

2. *Ключи сессии:* каждый билет включает ключ сессии, который используется клиентом для шифрования аутентификатора, посылаемого серверу. Дополнительно ключ сессии может впоследствии использоваться клиентом и сервером для защиты сообщений, передающихся в течение данной сессии. В версии 5 появилась возможность вести переговоры о ключе подсессии, который используется только для одного соединения. Для нового соединения будет применяться новый ключ шифрования.
3. *Атаки на пароль:* одним из слабых мест, присущих обеим версиям, являются атаки на пароль. Сообщение от AS клиенту включает нечто, зашифрованное ключом, основанным на пароле клиента. Оппонент может перехватить это сообщение и попытаться расшифровать его, используя различные пароли. Если результат дешифрования будет иметь корректный формат, то это означает, что оппонент раскрыл пароль клиента и может последовательно использовать его для получения доверительной грамоты от Kerberos. Версия 5 обеспечивает механизм, называемый предаутентификацией, который затрудняет атаки на пароли, но не предотвращает их.

# Аутентификационный протокол версии 5

## Получение билета, гарантирующего билет

1.  $C \rightarrow AS$ : Options,  $ID_C$ ,  $Realm_C$ ,  $ID_{TGS}$ , Times,  $Nonce_1$
2.  $AS \rightarrow C$ :  $Realm_C$ ,  $ID_C$ ,  $Ticket_{TGS}$ ,  $E_{K_C} [K_{C, TGS}, Times, Nonce_1, Realm_{TGS}, ID_{TGS}]$

$$Ticket_{TGS} = E_{K_{tgs}} [Flags, K_{C, TGS}, Realm_C, ID_C, AD_C, Times]$$

## Получение билета, гарантирующего сервис

3.  $C \rightarrow TGS$ : Options,  $ID_V$ , Times,  $Nonce_2$ ,  $Ticket_{TGS}$ ,  $Authenticator_C$
4.  $TGS \rightarrow C$ :  $Realm_C$ ,  $ID_C$ ,  $Ticket_S$ ,  $E_{K_C, TGS} [K_{C, S}, Times, Nonce_2, Realm_S, ID_S]$

$$Ticket_{TGS} = E_{K_{tgs}} [Flags, K_{C, TGS}, Realm_C, ID_C, AD_C, Times]$$

$$Ticket_S = E_{K_S} [Flags, K_{C, S}, Realm_C, ID_C, AD_C, Times]$$

$$Authenticator_C = E_{K_{C, tgs}} [ID_C, Realm_C, TS_1]$$

# Аутентификационный протокол версии 5

## Получение сервиса

5.  $C \rightarrow S$ : Options, Ticket<sub>S</sub>, Authenticator<sub>C</sub>

6.  $S \rightarrow C$ : E<sub>C,S</sub> [TS<sub>2</sub>, Subkey, Seq#]

$\text{Ticket}_S = E_{K_S} [\text{Flags}, K_{C,S}, \text{Realm}_C, \text{ID}_C, \text{AD}_C, \text{Times}]$

$\text{Authenticator}_C = E_{K_{C,S}} [\text{ID}_C, \text{Realm}_C, \text{TS}_1, \text{Subkey}, \text{Seq\#}]$

# Аутентификационный протокол версии 5

- Рассмотрим получение билета, гарантирующего билет. Сообщение (1) является запросом клиентского модуля на билет, гарантирующий билет. Как и прежде, оно включает идентификаторы пользователя и TGS. Добавлены следующие новые элементы:

*Realm*: определяет область пользователя.

*Options*: используется для запроса основных флагов, которые должны быть установлены в возвращаемом билете.

*Times*: используется клиентом для запроса следующих установок времени в билете:

from: требуемое начальное время для запрашиваемого билета.

till: требуемое время окончания для запрашиваемого билета.

rtime: требуемое время обновления.

*Nonce*: случайное число, повторяемое в сообщении 2, гарантирующее, что ответ своевременный и повтором оппонента не является.

# Аутентификационный протокол версии 5

- Сообщение (2) возвращает билет, гарантирующий билет, который содержит информацию для клиента, и блок, зашифрованный с использованием ключа шифрования, основанного на пользовательском пароле. Этот блок включает ключ сессии, который будет использоваться между клиентом и TGS, время, указанное в сообщении 1, nonce из сообщения 1 и определяемую TGS информацию. Сам билет включает ключ сессии, идентифицирующую информацию клиента, требуемое значение времени и флаги, которые отражают статус данного билета и требуемые опции. Эти флаги вводят важные новые функциональности в версии 5.
- Сравним получение билета, гарантирующего сервис, в версиях 4 и 5. Сообщение (3) в обеих версиях включает аутентификатор, билет и имя требуемого сервиса. Дополнительно версия 5 включает требуемое время, опции билета и nonce. Аутентификатор тот же самый, что используется в версии 4.
- Сообщение (4) имеет ту же структуру, что и сообщение (2); оно возвращает билет и информацию, необходимую клиентскому модулю, зашифрованную ключом сессии, разделяемым к настоящему времени клиентским модулем и TGS.

# Аутентификационный протокол версии 5

- Наконец, для получения сервиса в версии 5 появилось несколько новых возможностей. В сообщении (5) клиентский модуль может запросить опцию, которая требует взаимной аутентификации. Аутентификатор включает несколько следующих новых полей:
- *Подключ:* выбор клиентом ключа шифрования, который используется для защиты данной конкретной прикладной сессии. Если данное поле опущено, используется ключ сессии из билета ( $K_{Cs}$ ).
- *Sequence number:* дополнительное поле, которое определяет начальный номер последовательности, используемый сервером в сообщениях, посылаемых клиенту в течение данной сессии. Сообщения могут быть пронумерованы для предотвращения replay-атак.

## Аутентификационный протокол версии 5

- Если требуется взаимная аутентификация, сервер отвечает сообщением (6). Это сообщение включает отметку времени из аутентификатора. Заметим, что в версии 4 отметка времени возрастала на единицу. Это не является необходимым в версии 5, так как формат сообщения такой, что злоумышленник не может создать сообщение (6), не зная соответствующих ключей шифрования. Поле подключа, если оно присутствует, переопределяет поле подключа, если оно присутствует, в сообщении (5). Дополнительное поле номера последовательности определяет стартовый номер последовательности, используемый клиентом.

# Аутентификационный протокол версии 5

## Флаги билета

Поле флагов, введенное в билеты в версии 5, поддерживает расширенную функциональность по сравнению с версией 4. Рассмотрим флаги, которые могут быть определены в билете.

INITIAL	Данный билет получен с использованием AS-протокола и не получен на основе билета, гарантирующего билет
PRE-AUTHENT	При начальной аутентификации клиент был аутентифицирован прежде, чем был выдан билет
HW-AUTHENT	Протокол, используемый для начальной аутентификации, требует использования аппаратуры, ожидая ввода исключительно имени клиента
RENEWABLE	Говорит TGS о том, что данный используемый билет получен взамен билета, время действия которого истекло.
MAY-POSTDATE	Говорит TGS о том, что просроченный билет мог быть получен на основании данного билета, гарантирующего билет
POSTDATED	Определяет, что данный билет является просроченным; конечный сервер может проверить поле authtime, чтобы посмотреть, когда произошла первоначальная аутентификация.
INVALID	Определяет, что данный билет является недействительным и что прежде, чем он будет использоваться, его действительность должна быть подтверждена у TGS.
PROXIABLE	Говорит о том, что новый билет, гарантирующий сервис, с другим сетевым адресом может быть получен на основе существующего билета
PROXY	Определяет, что данный билет является агентом на другой сервис (проху)
FORWARDABLE	Говорит TGS, что новый билет, гарантирующий билет, может быть получен на основе данного билета, гарантирующего билет
FORWARDED	Определяет, что данный билет является либо forwarded, либо получен на основе аутентификации, включающей forwarded билет, гарантирующий билет

# Аутентификационный протокол версии 5

- Флаг INITIAL определяет, что данный билет получен от AS, а не от TGS. Когда клиент требует билет, гарантирующий сервис, от TGS, он предоставляет билет, гарантирующий билет, полученный от AS. В версии 4 это был способ, в конечном счете, получить билет, гарантирующий сервис. Версия 5 предоставляет дополнительную возможность, чтобы клиент мог получить билет, гарантирующий сервис, непосредственно от AS. Это применяется в таких, например, случаях, когда сервер изменения пароля хочет убедиться, что пароль клиента был только что проверен.
- Флаг PRE-AUTHENT, если установлен, определяет, что когда AS получит первоначальный запрос (сообщение 1), он аутентифицирует клиента, прежде чем выдать билет. Строгая форма этой предаутентификации остается неспецифицированной. Например, реализация MIT версии 5 имеет предаутентификацию в виде зашифрованной отметки времени. В этом случае клиентский модуль посылает AS предаутентификационный блок, содержащий случайное число, номер версии и отметку времени и зашифрованный с использованием пароля пользователя.

## Аутентификационный протокол версии 5

- AS расшифровывает блок и посылает билет, гарантирующий билет, если отметка времени находится в допустимом диапазоне. Другая возможность применения данного флага состоит в использовании смарт-карт, создаваемых с постоянно меняющимся паролем, который включается в предаутентификационное сообщение. Пароли, создаваемые картой, могут быть основаны на пользовательских паролях, но затем быть преобразованы смарт-картой так, чтобы в действительности использовались одноразовые пароли. Это предотвращает атаки, основанные на легко вскрываемых паролях. Если используется смарт-карта или аналогичное устройство, это определяется флагом HW-AUTHENT.

## Аутентификационный протокол версии 5

- Когда билет имеет долгое время жизни, существует опасность его кражи и последующего использования оппонентом в допустимый период. Если используется короткое время жизни для уменьшения подобной угрозы, то может возникнуть потребность в получении новых билетов. В случае билета, гарантирующего билет, клиент может хранить секретный ключ пользователя, который не подвержен риску, или повторно запрашивать у пользователя пароль. Компромисс состоит в использовании возобновляемых билетов. Билет с установленным флагом RENEWABLE включает два срока истечения: один для данного билета и один является самым поздним допустимым значением для истекаемого времени. Если новое время находится в пределах самого позднего допустимого значения, TGS может выдать новый билет с новым временем сессии и определить время его истечения. Преимущество данного механизма состоит в том, что TGS может отказаться обновлять билет, помечая его как украденный.

## Аутентификационный протокол версии 5

- Клиент может выдать запрос на предоставление AS билета, гарантирующего билет, с установленным флагом MAY-POSTDATE. Клиент может затем использовать этот билет для запроса билета от TGS с установленными флагами POSTDATED и INVALID. Впоследствии клиент может подать подтверждение на просроченный билет, чтобы сделать его действительным. Эта схема может использоваться для выполнения долгих пакетных заданий на сервере, который периодически требует билет. Клиент может один раз получить некоторое число билетов для данной сессии с несколькими значениями времени. Все, кроме первого билета, первоначально являются недопустимыми. При наступлении момента, когда требуется новый билет, клиент может сделать соответствующий билет действительным. При таком подходе клиент не будет повторно использовать свой билет, гарантирующий билет, для получения билета, гарантирующего сервис.

## Аутентификационный протокол версии 5

- В версии 5 стало возможным, чтобы сервер являлся проху для клиента, в результате чего устанавливаются верительные грамоты и привилегии клиента при запросе сервиса от другого сервера. Если клиент хочет использовать данный механизм, он требует билет, гарантирующий билет, с установленным флагом PROXIABLE. Когда такой билет предоставляется от TGS, TGS разрешает получать билет, гарантирующий сервис, с различных сетевых адресов; этот последний билет имеет установленный флаг PROXY. Приложение, получившее такой билет, может принять его или требовать дополнительной аутентификации с тем, чтобы обеспечить след аудита.

## Аутентификационный протокол версии 5

- Концепция проху является ограниченным случаем более сильной процедуры перенаправления. Если в билете установлен флаг FORWARDABLE, TGS может выдать запрашивающему билет, гарантирующий билет с различными сетевыми адресами, и установить флаг FORWARDED. Этот билет может затем быть представлен удаленному TGS. Такая возможность позволяет клиенту получить доступ к серверу из другой области без требования того, чтобы каждый Kerberos поддерживал секретный ключ с серверами Kerberos во всех других областях. Например, области могут быть структурированы иерархически. Тогда клиент может просматривать дерево вверх до общего узла и затем спускаться обратно до нужной целевой области. Каждый шаг прохода включает перенаправление билета, гарантирующего билет, к следующему TGS в пути.