Grundlagen der entfernten Authentifizierung und Autorisierung: Kerberos

Ausarbeitung im Seminar
Konzepte von Betriebssystem-Komponenten
Sommersemester 2010

Florian Lukas

21. Juli 2010

Inhaltsverzeichnis

1 Einleitung

Eine grundlegende Herausforderung jeder größeren Institution mit vielen Benutzern und Diensten, wie z. B. einer Universität oder einer Firma, ist die sichere und nachvollziehbare Kontrolle des Zugriffs auf ihre Dienste.

Dabei müssen sowohl die berechtigten Benutzer korrekt identifiziert, als auch der Zugriff auf einen Dienst technisch manipulationssicher realisiert werden. Zur Reduzierung des Administrationsaufwands sollen Benutzer- und Berechtigungsinformationen zentral gespeichert und verwaltet werden. Als weitere Erleichterung für die Benutzer kann eine solche Infrastruktur eingesetzt werden, um *Single-Sign-On (SSO)* zu ermöglichen, das einmalige Anmelden eines Benutzers an der Infrastruktur um danach alle verfügbaren Dienste ohne weitere Anmeldung nutzen zu können.

In dieser Arbeit sollen zunächst die Grundlagen der sicheren Kommunikation mit symmetrischen Schlüsseln dargestellt werden. Danach wird *Kerberos* vorgestellt, ein weit verbreitetes System, das aufbauend auf diesen Grundlagen alle oben genannten Forderungen erfüllt.

2 Grundlagen

Um eine sichere Dienstnutzung zu ermöglichen, ist eine Authentifizierung und Autorisierung der Benutzer notwendig. Im Folgenden sollen diese beiden unterschiedlichen Vorgänge erläutert werden.

2.1 Authentifizierung

Der erste Schritt zur sicheren Nutzung von Diensten ist die *Authentifizierung*, die Feststellung und Überprüfung der Identität des Benutzers. Dies kann durch verschiedenste Methoden erfolgen. Das verbreitetste Verfahren ist die Eingabe eines geheimen Passworts, aber es gibt auch Verfahren die auf dem Besitz eines Objekts, wie einer Smart-Card oder eines Zertifikats, beruhen. Weniger verbreitet, aber auch möglich, ist die Authentifizierung über biometrische Merkmale wie Fingerabdrücke oder Iris-Scans.

2.2 Autorisierung

Nach der erfolgreichen Authentifizierung eines Benutzers muss das System feststellen, ob der Benutzer auch berechtigt ist, die gewünschte Operation auszuführen. Diese Überprüfung, als auch die anschließende Berechtigung selbst, ist die *Autorisierung*. Die Art der Autorisierung ist meist sehr dienstspezifisch, da verschiedene Dienste ganz unterschiedliche Operationen anbieten. So kann ein Dateiserver Berechtigungen beispielsweise mithilfe einer ACL[1] überprüfen, während ein Datenbankadministrator die Ausführung einzelner SQL-Befehle mittels dem `GRANT` Befehl kontrollieren kann.

[1] *Access Control List:* Beliebig erweiterbare Liste von Benutzern und Gruppen mit ihren jeweiligen Berechtigungen.

3 Sichere Kommunikation mit symmetrischen Schlüsseln

Die sichere Nutzung von Diensten setzt neben Methoden zur Authentifizierung und Autorisierung auch ein sicheres Verfahren zur Kommunikation bzw. Nutzung der Dienste voraus, da sonst ein Angreifer nach erfolgter Authentifizierung und Autorisation die Anfragen eines berechtigten Benutzers abfangen und seine eigenen abschicken könnte.

In diesem Kapitel soll zunächst ein grundlegender Ansatz und dann ein konkretes Authentifizierungsprotokoll für ein solches Verfahren dargestellt werden.

3.1 Ansatz

Ein erster, simpler Ansatz ist die Verteilung von symmetrischen Schlüsseln zwischen allen möglichen Teilnehmerpaaren und die anschließende Verschlüsselung aller ausgetauschten Nachrichten eines Teilnehmerpaars mit dem jeweiligen Schlüssel. Dieses einfache Vorgehen hat allerdings einige schwere Nachteile.

Ein großes administratives Problem ist die Anzahl an benötigten Schlüsseln. Bei N Teilnehmern benötigt man $\frac{N^2-N}{2}$ geheim verteile Schlüssel, wenn jeder Teilnehmer mit jedem kommunizieren können soll, was bei steigender Anzahl an Benutzern kaum handhabbar ist. [7]

Darüber hinaus ist dieser Ansatz ohne vorherige Autorisation anfällig für *Replay*-Attacken, bei deinen ein Angreifer eine abgefangene, verschlüsselte Nachricht wiederholt oder verzögert. Solch eine gefälschte Nachricht kann nicht von einer normalen Unterschieden werden und führt so zur Wiederholung oder verzögerten Ausführung der ursprünglichen Operation. Dies kann z. B. eine Bestellung wiederholt absenden oder einen Aktienkauf verzögern, was unbedingt verhindert werden muss.

3.2 Key-Distribution-Center

Abbildung 1: Schlüsselverteilung bei Verwendung eines Key-Distribution-Centers mit mehreren Benutzern (A) und Diensten (B).

Um die Anzahl an Schlüsseln, die verteilt werden müssen, gering zu halten wird ein *Key-Distribution-Center* eingesetzt. Dies ist eine vertrauenswürdige Einheit, die mit jedem anderen Teilnehmer einen geheimen Schlüssel besitzt. Da im Zusammenspiel mit einem Authentifizierungsprotokoll dann keine weiteren Schlüssel mehr verteilt werden

müssen, reduziert sich die Schlüsselanzahl bei N Teilnehmern auf N Schlüssel, was praktisch handhabbar ist. [7]

In Abbildung 1 ist ein KDC dargestellt, wobei aufgrund der üblichen Verwendung zwischen Benutzern und Diensten unterschieden wird. Es ist dabei jedoch auch möglich, dass Benutzer oder Dienste untereinander kommunizieren, wenn das KDC ihnen das erlaubt.

3.3 Needham-Schroeder-Protokoll

Um die Kommunikations-Teilnehmer gegenseitig zu authentifizieren und danach einen neuen Sitzungsschlüssel zwischen beiden einzurichten wird ein *Authentifizierungsprotokoll* verwendet. Der neu eingerichtete Schlüssel wird dann für die Verschlüsselung aller nachfolgenden Nachrichten verwendet. Da der Sitzungsschlüssel nach dem Ende der Kommunikation verworfen wird, ist dieser Ansatz nicht anfällig für *Replay*-Attacken, die nach Beendigung der Kommunikation gestartet werden.

Das *Needham-Schroeder-Protokoll* ist ein solches Protokoll, das auch die Grundlage für das Authentifizierungsprotokoll von Kerberos bildet. [7]

Das Protokoll nutzt ein Key-Distribution-Center, das einen gemeinsamen Schlüssel mit jedem Teilnehmer besitzt und läuft nach folgendem Schema ab (siehe Abbildung 2):

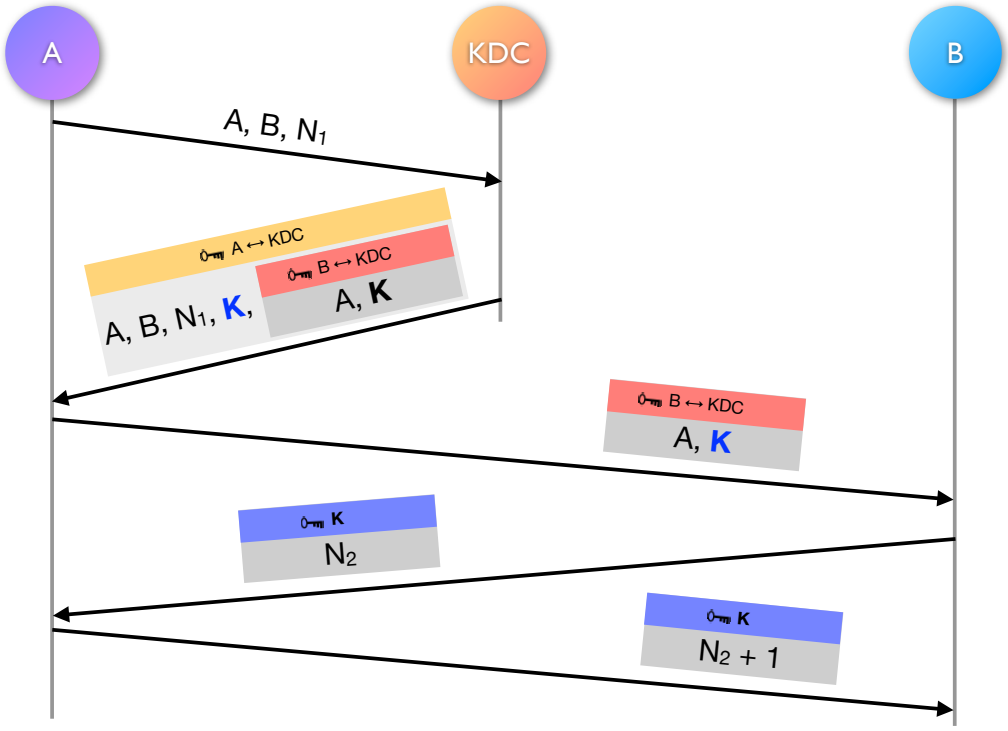

Abbildung 2: Protokollablauf des Needham-Schroeder-Protokolls. (Inhalt der Kästchen ist mit dem Schlüssel im Kopfbalken des Kästchens verschlüsselt.) Nach [5]

1. A sendet eine (unverschlüsselte) Nachricht mit seinem Namen, dem Namen des gewünschten Dienstes (B) und einer Zufallszahl (*Nonce*, N_1) an das KDC.

2. Das KDC erzeugt einen neuen Sitzungsschlüssel K für die Sicherung der Kommunikation zwischen A und B. Dieser Schlüssel K, der Name von B und die Nonce N_1 werden an A mit dem Schlüssel zwischen A und dem KDC verschlüsselt zurückgesendet. Außerdem ist in dieser verschlüsselten Nachricht der Name von A zusammen mit dem Schlüssel K verschlüsselt mit dem Schlüssel zwischen B und dem KDC enthalten.

3. A entschlüsselt diese Nachricht mit seinem Schlüssel, überprüft ob die Nonce N_1 mit der ersten Nachricht übereinstimmt und sendet dann den mit dem Schlüssel zwischen B und dem KDC verschlüsselten Teil an B.

   In diesem Schritt stellt A sicher, mit dem KDC zu kommunizieren, da nur das KDC die Antwort mit der Nonce N_1 mit dem geheimen Schlüssel zwischen A und dem KDC verschlüsseln kann.

4. B entschlüsselt diese Nachricht mit seinem Schlüssel, erzeugt eine neue Nonce N_2, verschlüsselt diese mit dem Sitzungsschlüssel K und sendet diese Nachricht an A.

5. A entschlüsselt die Nachricht mit K, und erniedrigt diese um eins und sendet das Ergebnis wieder verschlüsselt mit K an B zurück.

6. B entschlüsselt die Antwort von A und überprüft ob sie wirklich die erniedrigte Nonce N_2 enthält.

Mit den letzten beiden Nachrichten weißt A gegenüber B nach im Besitz des Schlüssels K zu sein, da A sonst die verschlüsselte Antwort nicht berechnen könnte. B weiß, dass das KDC den Sitzungsschlüssel K in seiner Antwort mit dem Schlüssel zwischen A und dem KDC verschlüsselt hat. Da nur A diesen Schlüssel besitzt, geht B davon aus wirklich mit A zu kommunizieren.

Das Needham-Schroeder-Protokoll besitzt eine Sicherheitslücke, falls der Sitzungsschlüssel von einem Angreifer herausgefunden wird (z. B. durch Zugriff auf den Hauptspeicher des Clients) und dieser die dritte Nachricht im Protokollablauf abgehört hat. Dann kann der Angreifer diese Nachricht einfach neu absenden und die Antwort-Nonce vom Dienst mit dem Sitzungsschlüssel verschlüsselt erhöht zurückschicken. Der Dienst kann nicht unterscheiden, ob die dritte Nachricht im Protokollablauf bereits einmal versandt wurde. Die einfachste Lösung für dieses Problem ist die Verwendung von Zeitstempeln statt Noncen im Protokoll. So kann ein Dienst feststellen, dass eine Nachricht nicht aktuell ist.

4 Kerberos

Kerberos ist ein offenes, weit verbreitetes System zur Authentifizierung (nicht aber zur Autorisierung) von Benutzern und Diensten, das auf dem Needham-Schroeder-Protokoll basiert. Es wurde am MIT entwickelt und die aktuelle Version 5 ist von der IETF standardisiert [4]. Kerberos-Implementierungen sind sowohl kommerziell als auch als Open-Source verfügbar.

Im Folgenden soll zuerst der Aufbau und die Funktionsweise des Kerberos-Systems dargestellt werden. Danach werden einige Erweiterungen sowie die Nachteile und Verwendung von Kerberos behandelt.

4.1 Aufbau und Schlüsselverteilung

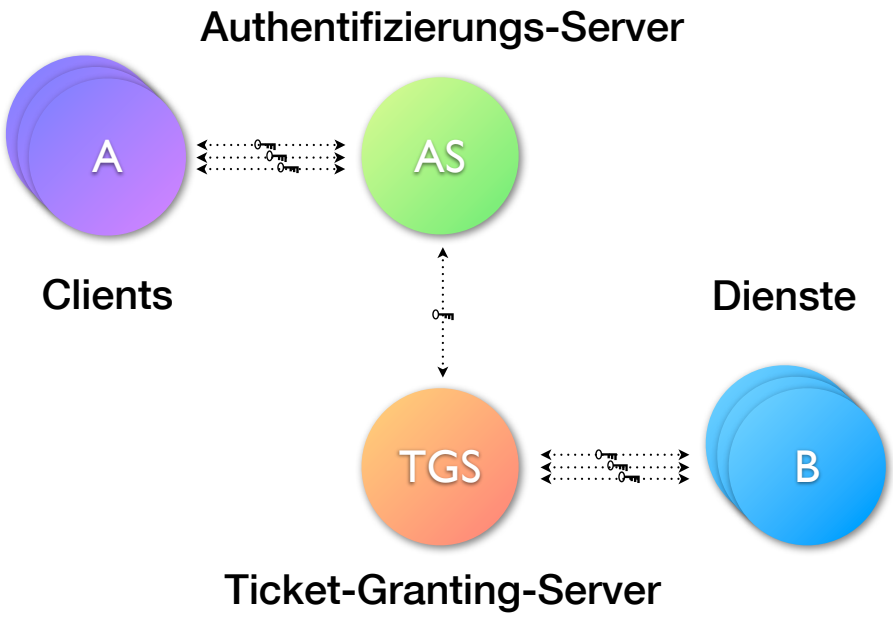

Abbildung 3: Aufbau und Schlüsselverteilung eines Kerberos-Systems.

Wie in Abbildung 3 dargestellt, besteht eine Kerberos-Installation aus vier verschiedenen Arten von Teilnehmern:

Benutzer (A) Die Benutzer verwenden die Dienste und melden sich dazu mit lokal installierter Kerberos-Client-Software (einmal pro Sitzung) beim Kerberos-System an.

Dienste (B) Die Dienste sind so konfiguriert, dass sie zur Authentifizierung ihrer Benutzer das Kerberos-System verwenden. Dienste, die eine Authentifizierung mittels Kerberos unterstützen, werden im Englischen manchmal auch als *kerberized* bezeichnet.

Authentifizierungs-Server (AS) Der Authentifizierungs-Server besitzt einen geheimen Schlüssel mit jedem Benutzer. Dieser Schlüssel wird normalerweise aus dem Passwort generiert, dass der Benutzer für die Anmeldung am Kerberos-System benutzt. Ohne diesen Schlüssel können die Antworten des Authentifizierungs-Server nicht entschlüsselt und weiter verwendet werden.

Ticket-Granting-Server (TGS) Der Ticket-Granting-Server besitzt einen Schlüssel mit jedem Dienst und einen Schlüssel mit dem Authentifizierungs-Server. Er stellt berechtigen Benutzern Kerberos-Tickets aus, die dem Benutzer ermöglichen einen Dienst zu benutzen.

Zur besseren Skalierbarkeit und Zuverlässigkeit können mehrere Authentifizierungs-Server und Ticket-Granting-Server verwendet werden. Dadurch verteilen sich viele Anfragen auf mehrere Server und bei Ausfall eines Servers kann das Kerberos-System weiter benutzt werden.

4.2 Protokollablauf

Um einen Dienst im Kerberos-System zu nutzen, erhält ein berechtigter Benutzer einen neu generierten Sitzungsschlüssel und ein sogenanntes *Kerberos-Ticket* für den Zugriff auf den gewünschten Dienst. Ein solches Ticket enthält neben Zusatzinformationen wie z. B. der Gültigkeitsdauer und der IP-Adressen der Kommunikationspartner den gleichen Sitzungsschlüssel, ist aber komplett verschlüsselt mit dem Schlüssel zwischen dem Aussteller und dem Dienst. Dadurch kann der Benutzer A, der ein solches Ticket erhält, dies weder lesen noch verändern.

Der genaue Protokollablauf basiert auf dem Needham-Schroeder-Protokoll (siehe Abschnitt 3.3) und läuft wie folgt ab (siehe Abbildung 4): [7]

1. Der Benutzer A möchte auf einen Dienst B zugreifen. Die Kerberos-Client-Software schickt eine Nachricht mit dem Namen von A, einem Zeitstempel und dem Namen eines geeigneten Ticket-Granting-Servers TGS an den Authentifizierungsserver.

2. Der Authentifizierungs-Server sucht A in seiner Benutzerdatenbank und erzeugt aus dessen Passwort den Schlüssel A↔AS. Er generiert auch einen neuen Sitzungsschlüssel K[A↔TGS]. Dieser Sitzungsschlüssel wird in ein sogenanntes *Ticket-Granting-Ticket* gepackt, ein Ticket, mit dem A vom Ticket-Granting-Server weitere Tickets anfordern kann. Das Ticket-Granting-Ticket und der neue Sitzungsschlüssel K[A↔TGS] werden an A zurückgeschickt. Diese Antwort ist komplett mit dem Schlüssel A↔AS verschlüsselt.

3. Nach Erhalt der Antwort muss A sein Passwort eingeben, damit der Kerberos-Client aus diesem den gleichen Schlüssel A↔AS erzeugen kann. Damit wird die Antwort des Authentifizierungs-Server entschlüsselt, falls das Passwort stimmt, ansonsten schlagen alle weiteren Schritte fehl. Das enthaltene Ticket-Granting-Ticket

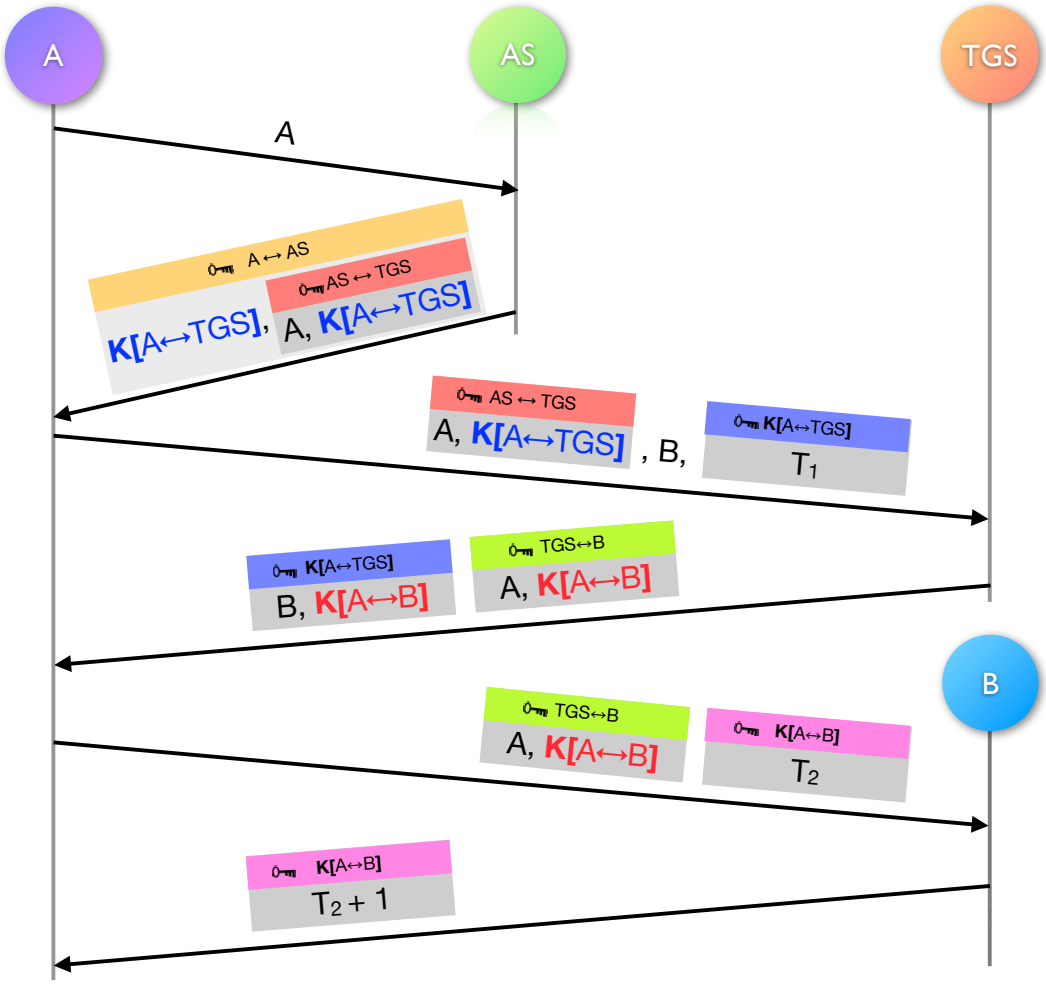

Abbildung 4: Protokollablauf der Kerberos-Authentifizierung (Zeitstempel und Zusatzinformationen zur Übersichtlichkeit weggelassen; K[X↔Y] ist ein generierter Sitzungsschlüssel zwischen X und Y) Nach [5]

wird zusammen mit dem Namen des gewünschten Dienstes und einem *Authenticator* an den Ticket-Granting-Server gesendet. Der Authenticator besteht aus einem Zeitstempel sowie dem Namen und der Adresse von A, wobei all dies mit dem Sitzungsschlüssel K[A↔TGS] verschlüsselt ist.

4. Der Ticket-Granting-Server entschlüsselt das Ticket-Granting-Ticket, entnimmt daraus den Sitzungsschlüssel K[A↔TGS] und entschlüsselt damit den Authenticator. Stimmen die Informationen darin mit denen des Tickets überein und ist der Zeitstempel noch aktuell genug, überprüft der Server, ob A berechtigt ist, auf den Dienst B zuzugreifen. Falls ja, wird ein neuer Sitzungsschlüssel K[A↔B] generiert. Dieser wird mit Zeitstempel, Namen und Adresse von A in ein neues *Service-Granting-Ticket* zusammengefasst, das mit dem Schlüssel TGS↔B verschlüsselt ist. Dieses Ticket und der Sitzungsschlüssel K[A↔B] wird an A zurückgesandt.

5. A entschlüsselt die Nachricht und erstellt einen neuen Authenticator, der mit dem Sitzungsschlüssel K[A↔B] verschlüsselt ist. Dieser und das erhaltene Ticket wird dann an B geschickt.

6. B entschlüsselt das Ticket mit dem Schlüssel TGS↔B und erhält daraus den Sitzungsschlüssel K[A↔B]. Damit kann B den Authenticator überprüfen und den darin enthaltenen Zeitstempel um eins erhöht zurücksenden, wobei diese Antwort wieder mit K[A↔B] verschlüsselt ist.

7. A entschlüsselt die Antwort und prüft ob der Zeitstempel korrekt erhöht wurde. Falls ja, geht A davon aus auch wirklich mit B zu kommunizieren, da nur B den Schlüssel TGS↔B kennt, mit dem man aus dem Ticket den Sitzungsschlüssel K[A↔B] erhalten kann. Nun haben sich A und B gegenseitig authentifiziert und können die eigentliche Dienst-Nutzung mit dem Sitzungsschlüssel K[A↔B] beginnen.

4.3 Erweiterungen

4.3.1 Cross-Realm-Trust

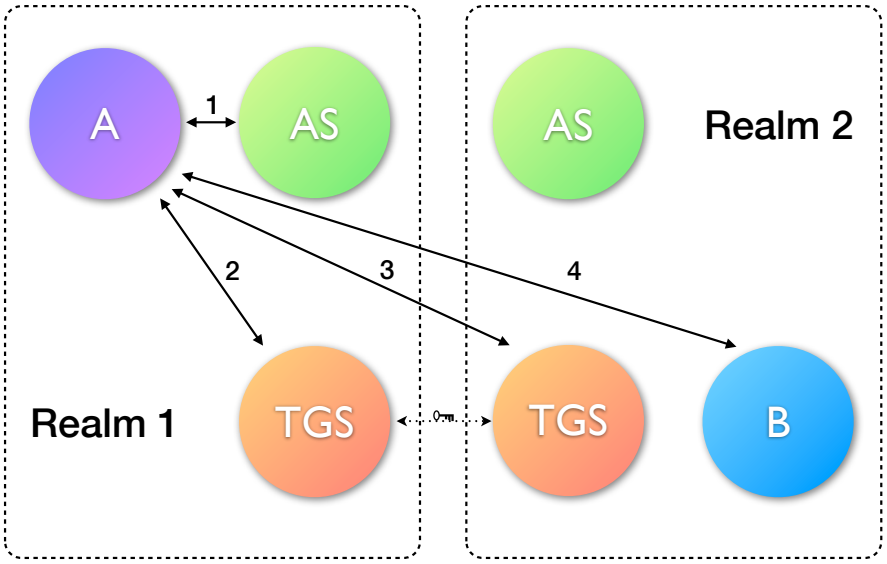

Abbildung 5: Aufbau und Protokollablauf bei Cross-Realm-Trust.

Falls eine große Institution mit mehreren Teilen, wie z. B. eine Firma mit mehreren Standorten, Kerberos nutzen will, ist es vorteilhaft, wenn die einzelnen Teile eigenständige Kerberos-Installationen, sogenannte *Realms*, betreiben. Dadurch laufen nicht alle Authentifizierungsvorgänge der gesamten Institution über eine zentrale, entfernte Kerberos-Installation, wobei z. B. beim Ausfall der Intranet-Verbindung zu dieser alle Dienste

unverfügbar wären obwohl die Dienste und die entfernte Kerberos-Installation noch korrekt arbeiten. Um bei diesen eigenständigen Installationen trotzdem alle Dienste aller Realms mit einmaliger Anmeldung nutzen zu können kann ein *Cross-Realm-Trust* aufgebaut werden.

Hierfür muss nur ein geheimer Schlüssel zwischen den Ticket-Granting-Servern der beteiligten Realms vereinbart werden. Beim Zugriff auf einen Dienst in einem entfernten Realm (siehe Abbildung 5) wird dann zunächst der entfernte Ticket-Granting-Server wie ein normaler Dienst behandelt, d. h. der lokale TGS stellt ein Ticket für den entfernten TGS aus. Danach kann mit diesem Ticket beim entfernten TGS ein Ticket zur Benutzung des gewünschten Dienstes angefordert werden. [8]

Damit bei einer größeren Anzahl an Realms nicht Schlüssel zwischen allen beteiligten Ticket-Granting-Servern verteilt werden müssen, können Kerberos-Realms wie im DNS-System hierarchisch angeordnet werden. Dann müssen nur noch Schlüssel zum hierarchisch höheren Knoten verteilt werden.

4.3.2 Pre-Authentication

Im ursprünglichen Kerberos-Protokollablauf antwortet der Authentifizierungs-Server ohne vorherige Authentifizierung mit einem gültiges Ticket-Granting-Ticket. Da dieses Ticket symmetrisch verschlüsselt ist, kann eigentlich nur der Benutzer, für den dieses Ticket bestimmt ist, dieses Ticket auch verwenden. Allerdings kann ein Angreifer sich auf diesem Weg ein gültiges Ticket besorgen und dann versuchen den Schlüssel zu rekonstruieren, z. B. durch Wörterbuchattacken.

Um dies zu verhindern wurde die optionale *Pre-Authentication* in Kerberos eingeführt. Dabei muss ein Client erst einen Zeitstempel mit seinem (aus dem Passwort generierten) Schlüssel verschlüsselt an den Authentifizierungs-Server senden, bevor er das Ticket-Granting-Ticket erhält. So werden diese Offline-Passwort-Attacken verhindert, da ein Angreifer ohne Kenntnis des Passworts den Zeitstempel nicht verschlüsseln kann. [8]

4.3.3 Spezielle Tickets

Kerberos-Tickets haben ein begrenzte Gültigkeitsdauer, die für jedes Ticket einzeln gesetzt werden kann. Für manche Anwendungen, wie z. B. sehr lange Simulationen, ist es jedoch unerwünscht, Tickets mit solch langer Gültigkeitsdauer auszustellen, da Kerberos keinen Mechanismus enthält, ein kompromittiertes Ticket wie bei SSL-Zertifikaten zurückzurufen bzw. ungültig zu machen. Bei manchen Anwendungen ist vorher auch nicht bekannt, wie lang ein Ticket benötigt wird. Deshalb wurden *erneuerbare Tickets* eingeführt, welche neben ihrem eigentlichen Ablaufdatum noch ein späteres Datum enthalten, bis zu dem sie zum Ticket-Granting-Server zurückgeschickt werden können. Dieser schickt daraufhin ein neues Ticket zurück, das identisch ist, aber mit neuer Gültigkeitsdauer und neuem Datum der letzten Möglichkeit der Erneuerung. Wurde ein Ticket mittlerweile kompromittiert und ist dies dem Kerberos-System bekannt, weigert sich der Ticket-Granting-Server, ein neues Ticket zurückzusenden, und der Angreifer kann den Dienst nicht weiter nutzen.

Ganz ähnlich funktionieren *vordatierte Tickets*, welche z. B. für zeitlich geplante oder in Warteschlangen eingereihte Aufgaben verwendet werden, die erst in der Zukunft gestartet werden und dann ein gültiges Ticket benötigen. Statt ein Ticket auszustellen, das so lange gilt, bis es benötigt wird, wird ein vordatiertes Ticket ausgestellt, dass bei Erhalt ungültig ist. Wie die erneuerbaren Tickets enthält es zusätzlich ein Datum, ab dem es dem Ticket-Granting-Server erneut zugesandt werden kann, damit dieser es gültig zurücksendet. Danach kann es zur Nutzung des Dienstes verwendet werden.

Gelegentlich ist es nötig, dass ein Dienst im Auftrag des Benutzers eine Aktion bei einem anderen Dienst ausführt, wie z. B. ein Druckdienst, der von einem Dateiserver eine Datei drucken soll. Für diese Fälle gibt es *weiterleitbare Tickets*. Diese Tickets können dem Ticket-Granting-Server zurückgesandt werden, der dabei allerdings eine andere IP-Adresse als die des Clients einfügt. So kann die IP-Adresse des Dienstes eingefügt werden, der dann dieses Ticket benutzen kann um auf einen anderen Dienst zuzugreifen. [6]

4.4 Nachteile

Fällt der Authentifizierungs-Server oder der Ticket-Granting-Server aus kann kein Benutzer mehr irgendeinen Dienst benutzen, da der Benutzer kein Ticket-Granting-Ticket bzw. keine Dienst-Tickets mehr erhält. Dieses Problem kann durch redundante Authentifizierungs-Server und Ticket-Granting-Server gelöst werden, wodurch weiterhin Tickets ausgestellt werden können.

Des Weiteren erfordert der Einsatz von Zeitstempeln im Kerberos-Protokoll, dass die Systemuhren aller Rechner im Kerberos-System synchron laufen. Üblich ist die Voreinstellung, dass eine Kerberos-Anfrage fehlschlägt, wenn die Zeitstempel mehr als fünf Minuten auseinander liegen. Dieses Problem ist jedoch durch den Einsatz von Zeitsynchronisationsprotokollen wie NTP leicht behoben. [7]

Der größte Nachteil von Kerberos ist, dass die Sicherheit des gesamten Systems an der Sicherheit der Authentifizierungs- und Ticket-Granting-Server hängt. Wird einer dieser Server kompromittiert, z. B. durch einen Angreifer oder einen ehemaligen Mitarbeiter, der noch Zugriff besitzt, kann dieser sich als jeder Benutzer ausgeben und jeden Dienst nutzen. Diese Schwachstelle ist durch den Aufbau des Kerberos-Systems bedingt und kann technisch nicht einfach gelöst werden. Deshalb ist unbedingt darauf zu achten, dass diese Server so gut wie möglich abgesichert sind.

4.5 Verwendung

Kerberos ist kommerziell verfügbar und weit verbreitet als zentraler Bestandteil von Microsoft Active Directory [8] und Apple Open Directory [3]. Auch für Unix/Linux Umgebungen sind verschiedene Implementierungen vorhanden.

In lokalen Netzwerken ohne zentralen Kerberos-Server benutzen Mac OS X Betriebssystemdienste (wie z. B. Datei oder Remote-Desktop-Dienste) ihren eigenen Kerberos-Server, damit sich ein Benutzer nur einmal pro entfernten Computer anmelden muss. [1] Bei Verwendung von *Back to my Mac* entfällt diese Anmeldung sogar komplett, da dabei der Benutzer über digitale Zertifikate seines *MobileMe*-Kontos authentifiziert wird. [2]

Auch Microsofts *Xbox Live* Dienst nutzt intern Kerberos zur Authentifizierung der momentan 23 Millionen Nutzer. Dabei werden sowohl ein Ticket-Granting-Ticket für den Xbox Live Benutzer als auch für die Xbox selbst von Microsofts `PASSPORT.NET` Realm angefordert[2].

5 Zusammenfassung

Kerberos bietet, basierend auf dem Needham-Schroeder-Protokoll, alle notwendigen Eigenschaften eines Authentifizierungssystems für größere Institutionen. Es bietet Sicherheit, indem es sowohl bei aktiven als auch passiven Angreifern verhindert, dass sich diese als ein anderer Benutzer ausgeben können. Durch mehrere redundante Kerberos-Server kann das System sowohl zuverlässig als auch skalierbar ausgelegt werden, was besonders wichtig ist, da sonst keinerlei Dienste mehr benutzt werden können. Darüber hinaus bietet Kerberos Transparenz für die Benutzer, die nur einmalig ihr Passwort eingeben müssen und ab dann bei allen Diensten automatisch authentifiziert werden.

Literatur

[1] Apple Inc. *About Kerberos in Mac OS X 10.5 clients*, Oct. 2007. `http://support.apple.com/kb/TA24992`.

[2] Apple Inc. *Back to My Mac User's Guide*, Sept. 2009. `http://images.apple.com/mobileme/docs/L358808A_BackMac_UG_v3.pdf`.

[3] Apple Inc. *Open Directory Administration*, 2009. `http://images.apple.com/server/macosx/docs/Open_Directory_Admin_v10.6.pdf`.

[4] Neuman, C., Yu, T., Hartman, S., and Raeburn, K. The Kerberos Network Authentication Service (V5). RFC 4120, IETF, July 2005. `http://www.ietf.org/rfc/rfc4120.txt`.

[5] Peterson, L. L., and Davie, B. S. *Computernetze: eine systemorientierte Einführung*. dpunkt-Lehrbuch. dpunkt, Heidelberg, 2007.

[6] Ricciardi, F. Kerberos Protocol Tutorial, Nov. 2007. `http://kerberos.org/software/tutorial.html`.

[7] Schäfer, G. *Netzsicherheit: algorithmische Grundlagen und Protokolle*. dpunkt-Lehrbuch. dpunkt.verlag, Heidelberg, 2003.

[8] Walla, M. Kerberos explained, May 2000. `http://technet.microsoft.com/en-us/library/bb742516.aspx`.

[2] Paketmitschnitt im tcpdump-Format: `http://wwwcip.informatik.uni-erlangen.de/~siflluka/xbox-kerberos.cap`