JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Autorisierung & Authentifizierung

Systemtechnik Referat
5BHIT 2015/16

Andreas Ernhofer & Jakub Kopec

Version 1.0

Note:

Begonnen am 20. Oktober 2015

Betreuer: Markus Schabel & Michael Borko

Beendet am 18. November 2015

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Allgemein

1.1 Sichere Kommunikation

Damit eine sichere Kommunikation gewährleistet werden kann, ist eine Authentifizierung der kommunizierenden Parteien erforderlich. In vielen Fällen erfordert sie auch die Sicherstellung der Nachrichtenintegrität und unter Umständen auch der Vertraulichkeit.

Zum Schutz der Kommunikation zwischen Clients und Servern kann man an die Einrichtung eines **sicheren Kanals** denken. Ein sicherer Kanal schützt Sender und Empfänger vor Abfangen, Änderung und Fälschung von Nachrichten. Er schützt nicht zwangsläufig auch vor Störungen. Nachrichten werden vor Störungen über die Sicherstellung der Vertraulichkeit geschützt: Der sichere Kanal gewährleistet, dass seine Nachrichten nicht von Eindringlingen belauscht werden können. Der Schutz vor Änderungen und Fälschungen durch Eindringlinge wird über Protokolle zur **wechselseitigen Authentifizierung und Nachrichtenintegrität** gewährleistet.

[TANA] 9.2 Sichere Kanäle, S. 432

1.2 Allgemeine Sicherheitsanforderungen

In der Regel geht es in IT-Umgebungen immer um den Austausch und die Verarbeitung von Daten. Dabei kann es sich um die Nutzdaten der Anwender handeln, aber auch um Authentisierungs- und Autorisierungsdaten. In beiden Fällen gilt es, die Sicherheit dieser Daten zu gewährleisten. Folgende Anforderungen werden heutzutage typischerweise an die Sicherheit der Daten Gestellt:

1.2.1 Vertraulichkeit

Es gibt sensible Daten, die geheim sind und nur Einzelnen zugänglich sein sollen. Dazu gehören Passwörter oder vertrauliche E-Mails. Für solche Informationen muss man sicherstellen können, dass sie nicht in die Hände unbefugter Dritter gelangen. Man bezeichnet diese Anforderung als Vertraulichkeit.

1.2.2 Integrität

Kommt von dem lateinischen Wort *„integritas“* und bedeutet „Unversehrtheit“. Potenziellen Angreifern darf es nicht möglich sein, sensible Daten zu verändern. Zumindest muss die Möglichkeit bestehen solche Änderungen zu erkennen. Unter *Integrität* versteht man die Anforderung, Daten auf unerlaubte Manipulationen hin überprüfen zu können.

1.2.3 Authentizität

Darunter versteht man die Anforderung, die Echtheit der Daten feststellen zu können. Dazu gehört auch die Überprüfbarkeit der Identität des Senders m Sinne der *Authentifizierung*.

[PROM] 4.1.1 Allgemeine Sicherheitsüberlegungen, S. 43/44

2 Authentisierung

Unter Authentisierung versteht man den Nachweis der eigenen Identität. Dabei kann es sich um die Identität einer Person (eines Anwenders) oder auch um die eines Computerprogramms handeln. Wird dagegen eine angegebene Identität überprüft, so spricht man von *Authentifizierung*. Auch hier kann es um die Identität eines Menschen oder eines Programms gehen.

Ein Beispiel: Ein Anwender, der sich an einem Computer anmeldet, gibt dazu seinen Nutzernamen und zusätzliche Informationen (typischerweise ein Passwort) an. Damit *authentisiert* er sich gegenüber dem Anmeldeprogramm und *authentifiziert* den Anwender.
[PROM] 2.1 Authentisierung, S. 9

2.1 Authentisierungsmerkmale

Die eigene Identität zu belegen oder die Identität eines anderen zu überprüfen, stellt eine einfache Aufgabe dar, wenn es sich bei den Beteiligten um Menschen Handelt. Denn diese können sich anhand von Merkmalen wie ihrem Aussehen oder dem Klang ihrer Stimme erkennen. Darüber hinaus besitzen Menschen weitere eindeutige Merkmale wie den Fingerabdruck oder die Unterschrift. Solche Merkmale können ebenfalls für die Identitätsprüfung herangezogen werden, wenn Aussehen oder Stimme unbekannt sind. Auch wenn alle diese Merkmale unbekannt sein sollten, so können sich zwei Menschen immer noch anhand eines gemeinsamen Geheimnisses (Shares Secret) gegenseitig ihre Identität nachweisen. Solch ein gemeinsames Geheimnis kann bei zwei Personen beispielsweise ein Losungs- bzw. *Passwort* sein.

Computer besitzen zunächst keine vergleichbaren Merkmale, die sie derartig einmalig machen. Außerdem haben sie nur eingeschränkte Fähigkeiten, die oben beschriebenen Merkmale wie Aussehen oder Stimme auszuwerten. Diese Unfähigkeit gleichen sie wiederum dadurch aus, dass sie sich wesentlich längere und damit auch sicherere Geheimnisse merken können als Menschen. Anstelle von Codewörtern benutzen Computer für den Zweck sehr lange *kryptographische Schlüssel (Keys)*. Dabei handelt es sich um möglichst zufällig gewählte Folgen von Nullen und Einsen einer bestimmten Länge. Eine Schlüssellänge von 156 Bit ist heutzutage für symmetrische Kryptografie üblich.

Unsere menschlichen Gehirne sind kaum in der Lage, sich solche kryptografischen Schlüssel zu merken. Wir müssen und daher mit Passwörtern behelfen. Computer können aus diesen Passwörtern wieder um kryptografische Schlüssel erzeugen, mit denen sie besser umgehen können. Natürlich haben solche passwortbasierten Schlüssel lediglich die gleiche Sicherheit wie die zugrundeliegenden Passwörter.
[PROM] 2.1.1 Authentisierungsmerkmale, S. 10

2.2 Arten der Authentisierung

Passwörter und kryptografische Schlüssel sind nicht die einzigen möglichen Authentisierungsmerkmale. Im Allgemeinen kann man Authentisierungsmerkmale in folgende Kategorien einteilen:

- Authentisierung durch Wissen
- Authentisierung durch Besitz
- Authentisierung durch biometrische Merkmale

[PROM] 2.1.1 Authentisierungsmerkmale, Kategorien, S. 11

2.2.1 Authentisierung durch Wissen

Hier authentisiert sich der Benutzer durch vorgetragenes Wissen. Beispiele sind ein Passwort oder eine persönliche Identifikationsnummer (PIN). Diese Merkmale haben mehrere Nachteile: So eignen sie sich beispielsweise nicht für sehr vergessliche Menschen und können außerdem ausspioniert werden, auch ohne dass das Opfer das merkt.

[POGW] 3.1 Zugangskontrolle: Ansätze, S. 123

2.2.2 Authentisierung durch Besitz

Diese Art der Authentifikation basiert auf dem Besitz einer fälschungssicheren Marke, die Informationen in mechanischer, magnetischer oder elektrischer Form enthält. Dazu zählen Dinge wie der *Personalausweis*, die *EC-Karte* oder auch die *Transaktionsnummern* beim Internet-Banking *(TAN-Listen).* Im Computerumfeld zählen hierzu *Token-Karten*, *SSL-Zertifikate*, *Smartcards* oder auch Listen von *Einmalpasswörtern.* Der Nachteil hierbei ist, dass man sie entwenden kann, wobei man hier im Computerumfeld diesem Manko durch eine sehr kurze Gültigkeit dagegen steuern kann.

[POGW] 3.1 Zugangskontrolle: Ansätze, S. 124

2.2.3 Authentisierung durch biometrische Merkmale

Die biometrischen Merkmale gehören eigentlich auch zu den Dingen, die man besitzt. Sie haben aber den Vorteil, dass sie nicht so leicht zu entwenden sind, Zu diesen Merkmalen zählt man z.B. *Aussehen, Stimme, Fingerabdruck, Augenhintergrund oder Unterschrift.* Der Nachteil der biometrischen Merkmale liegt darin, dass Computer diese nicht besonders gut erkennen und verarbeiten können und dass die Authentisierung mittels dieser Merkmale daher fehleranfällig ist.

[PROM] 2.1.1 Authentisierungsmerkmale, Kategorien, S. 11

2.2.4 Conclusio

Alle drei Kategorien von Authentisierungsmerkmalen haben ihre Vorteile und Nachteile. Idealerweise Kombiniert man daher Merkmale verschiedener Kategorien, um die einzelnen Nachteile auszugleichen. Beispielsweise sind Geldgeschäfte mit einer EC-Karte (Authentisierung durch Besitz) nur durch zusätzliche Kenntnis einer PIN (Authentisierung durch Wissen) möglich.

[PROM] 2.1.1 Authentisierungsmerkmale, Kategorien, S. 12

3 Authentifizierung

Bevor wir in die Einzelheiten einsteigen wollen, lohnt es sich zu wissen, dass Authentifizierung und Nachrichtenintegrität aufeinander angewiesen sind. Dazu ein Kleines Gedankenexperiment. Betrachten wir zwei Systeme. Im ersten System wird nur die Authentifizierung gewährleistet und im zweiten nur die Integrität. Im ersten System weiß Bob vielleicht, dass die Nachricht (m) die er empfangen hat von Alice kommt, aber was nützt ihm das wenn er nicht weiß ob die Nachricht (m) am Weg zu ihm von jemand anderem manipuliert wurde? Gehen wir nun vom zweiten System aus. Wie glücklich kann Bob beim Erhalt einer Nachricht sein, er habe 1 Mio. Euro in der Lotterie gewonnen, wenn er nicht überprüfen kann, dass die Nachricht von den Organisatoren der Lotterie stammt? Demzufolge sollten Authentifizierung und Nachrichtenintegrität nicht getrennt werden.

Nehmen wir nun an, Alice ergreift die Initiative und richtet gemeinsam mit Bob einen Kanal ein. Ist der Kanal einmal eingerichtet worden, kann sich Alice sicher sein, dass sie mit Bob redet und vice versa.
Um in der Folge die Integrität der Datennachrichten sicherzustellen, die nach der Authentifizierung ausgetauscht werden, ist es gängige Praxis, Kryptografie mit geheimen Schlüsseln zu verwenden, indem zufällige Schlüssel (Sitzungsschlüssel) generiert werden. Ein Sitzungsschlüssel ist ein gemeinsamer (geheimer) Schlüssel, der aus Gründen der Integrität und möglicherweise auch der Vertraulichkeit zur Verschlüsselung von Nachrichten verwendet wird. Ein derartiger Schlüssel wird im Allgemeinen nur benutzt solange es den Kanal gibt. Bei der Schließung des Kanals wird sein zugehöriger Schlüssel verworfen (bzw. auf sichere Art zerstört). Wir kommen später auf Sitzungsschlüssel zurück.

[TANA] 9.2.1 Authentifizierung, S. 433

3.1 Authentifizierung auf der Grundlage eines gemeinsamen geheimen Schlüssels

Betrachten wir das Authentifizierungsprotokoll auf der Grundlage eines geheimen Schlüssels, den Alice und Bob bereits gemeinsam nutzen. Das Protokoll verfolgt einen gemeinsamen Ansatz, bei dem eine Seite die andere zu einer Antwort auffordert, die nur korrekt sein kann, wenn die andere Seite den gemeinsamen Schlüssel kennt. Solche Lösungen sind auch als **Challenge-Response-Verfahren** bekannt.

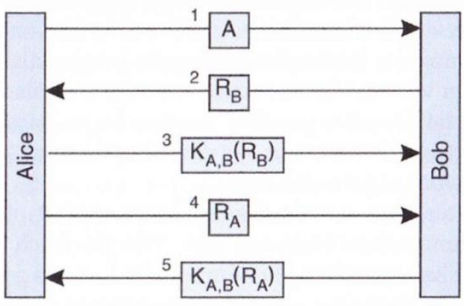

Abbildung 1: [TANA] Abbildung 9.12

Im Falle der Authentifizierung auf der Grundlage eines gemeinsamen geheimen Schlüssels geht das Protokoll wie in der obigen Abbildung vor. Zunächst sendet Alice Bob ihre Identität (Nachricht 1), womit sie anzeigt, dass sie einen Kommunikationskanal zwischen beiden einrichten möchte. Bob sendet Alice nachfolgend eine Aufgabe R_B, dargestellt als Nachricht 2. Eine solche Aufgabe kann in Form einer Zufallszahl erfolgen. Alice wird aufgefordert, diese

Aufgabe mit dem geheimen Schlüssel $K_{A,B}$ zu verschlüsseln, den sie mit Bob gemeinsam nutzt, und Bob die verschlüsselte Aufgabe zurückzusenden. Wenn Bob die Antwort $K_{A,B}$ (R_B) auf seine Aufgabe R_B erhält, kann er die Nachricht erneut unter Verwendung des gemeinsamen Schlüssels entschlüsseln, um zu sehen, ob die R_B enthält. Falls ja, dann weiß er, dass sich Alice auf der anderen Seite befindet, denn wer sonst hätte R_B mit $K_{A,B}$ verschlüsseln können? Bob hat also nun überprüft, dass er wirklich mit Alice redet. Beachten Sie aber, dass Alice noch nicht überprüft hat, dass wirklich Bob am anderen Ende des Kanals ist. Daher sendet sie eine Aufgabe R_A (Nachricht 4), auf die Bob mit $K_{A,B}$ (R_A) antwortet, was als Nachricht 5 dargestellt wird. Wenn Alice diese mit $K_{A,B}$ entschlüsselt und ihre R_A sieht, weiß sie, dass sie mit Bob redet.

[TANA] 9.2.1 Authentifizierung, S. 434

3.2 Authentifizierung über ein KDC (Key Distribution Center)

Eines der Probleme bei der Verwendung eines gemeinsamen geheimen Schlüssels zur Authentifizierung ist die Skalierbarkeit. Wenn ein verteiltes System N Hosts enthält und jeder Host gemeinsam mit jedem der anderen N-1 Hosts einen geheimen Schlüssel nutzen muss, erfordert dies, dass das System als als Ganzes N (N-1)/2 und jeder Host N-1 Schlüssel verwalten muss. Bei großen N führt das zu Problemen. Eine Alternative ist ein zentralisierter Ansatz wie ein *Key Distribution Center (KDC)*. Dieses KDC nutzt gemeinsam mit jedem der Hosts einen geheimen Schlüssel, aber die Hosts untereinander benötigen nicht mehr paarweise geheime Schlüssel. Die Verwendung eines KDC macht es also erforderlich, dass wir statt N (N-1)/2 nur N Schlüssel verwalten müssen, was offensichtlich eine Verbesserung darstellt.

Wenn Alice einen sicheren Kanal mit Bob einrichten will, kann sie das mithilfe eines (zuverlässigen) KDC tun. Die Idee dabei ist, dass das KDC einen Schlüssel sowohl an Alice als auch an Bob ausgibt, den sie dann zur Kommunikation nutzen können.

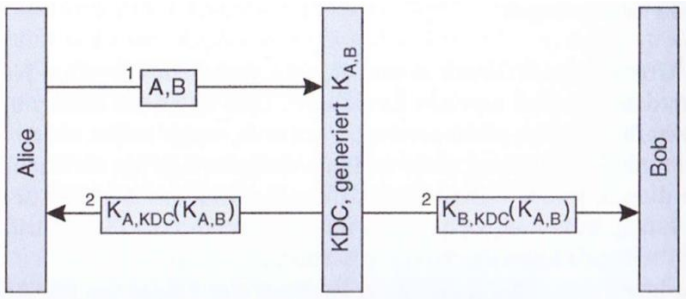

Abbildung 2: [TANA] Abbildung 9.15

Alice sendet zunächst eine Nachricht an das KDC mit der Mitteilung, dass sie mit Bob reden will. Das KDC sendet eine Nachricht zurück, die einen gemeinsamen geheimen Schlüssel $K_{A,B}$ enthält, dem sie verwenden kann. Die Nachricht wird mit dem geheimen Schlüssel $K_{A,KDC}$ verschlüsselt, den Alice mit dem KDC gemeinsam nutzt. Darüber hinaus sendet das KDC $K_{A,B}$ auch an Bob, nun aber verschlüsselt mit dem geheimen Schlüssel $K_{B,KDC}$ den es mit Bob gemeinsam nutzt.

Der Hauptnachteil dieses Ansatzes liegt darin, dass Alice möglicherweise einen Kanal mit Bob einrichten will, bevor der überhaupt den gemeinsamen geheimen Schlüssel vom KDC erhalten hat. Zudem muss das KDC Bob zum Handeln auffordern, indem es ihm den

Schlüssel weitergibt. Diese Probleme lassen sich umgehen indem man Bob's Nachricht ebenfalls an Alice sendet und ihr somit überlässt mit Bob in kontakt zu treten. Bob's Nachricht nennt man in diesem Fall auch *Ticket*. Es ist die Aufgabe von Alice dieses Ticket an Bob weiterzuleiten. Bob ist auch der einzige der das Ticket (also die für ihn bestimmte Nachricht) entschlüsseln kann, da nur er neben dem KDC weiß wie das Ticket zu entschlüsseln ist.

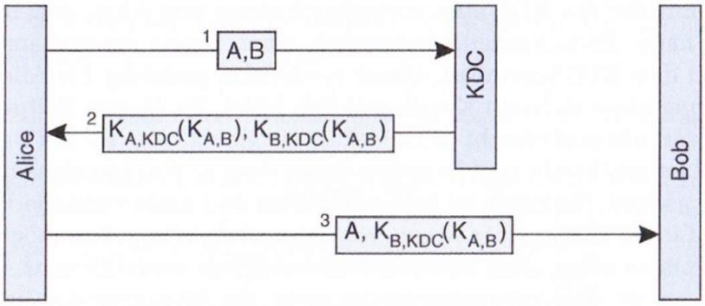

Abbildung 3: [TANA] Abbildung 9.16

Das in der Abbildung dargestellte Protokoll ist eine Variante des **Needham-Schroeder-Authentifizierungsprotokolls** unter Verwendung eines KDC.

Das Needham-Schroeder-Protokoll ist ein Mehrweg-Challenge-Response-Verfahren. Es funktioniert folgendermaßen: Wenn Alice einen sicheren Kanal mit Bob einrichten will, sendet sie eine Anfrage an das KDC mit einer Aufgabe R_A und ihrer Identität A und natürlich der von Bob. Das KDC antwortet ihr mit der Übersendung des Tickets $K_{B,KDC}(K_{A,B})$ zusammen mit dem geheimen Schlüssel $K_{A,B}$, den sie nachfolgend mit Bob gemeinsam nutzen kann. Die Aufgabe R_{A1}, die Alice zusammen mit ihrer Anfrage, einen Kanal zu Bob einzurichten, an das KDC sendet, wird auch als *Nonce* bezeichnet. Eine *Nonce* ist eine Zufallszahl, die nur einmal verwendet wird, zum Beispiel eine aus einer sehr großen Menge gewählte. Der Hauptzweck einer Nonce ist, zwei Nachrichten eindeutig miteinander in Beziehung zu setzen, in diesem Fall Nachricht 1 und Nachricht 2. Insbesondere dadurch, dass R_{A1} wieder in Nachricht 2 enthalten ist, ist sich Alice sicher, dass Nachricht 2 in Antwort auf Nachricht 1 gesendet worden ist und dass es sich zum Beispiel nicht um die Antwort auf eine ältere Nachricht handelt. Somit werden derartige Angriffe ausgeschlossen, da das erneute Einspielen einer alten Nachricht sofort entdeckt wird.

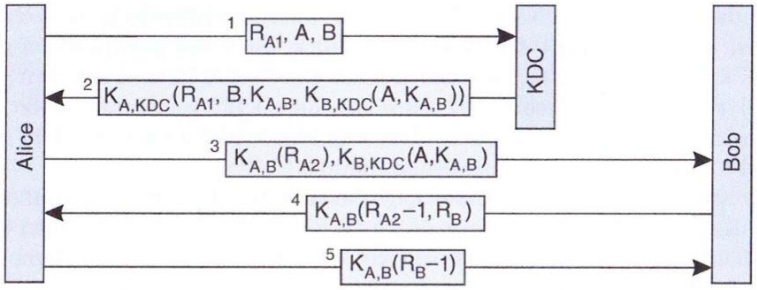

Abbildung 4: [TANA] Abbildung 9.17

Das Needham-Schroeder-Protokoll, wie wir es hier dargestellt haben, hat jedoch die Schwäche, dass Chuck Nachricht 3 wieder einspielen und Bob dazu bewegen könnte, einen Kanal einzurichten, wenn er je Zugriff auf einen alten Schlüssel $K_{A,B}$ erlangen würde. Bob würde dann glauben, dass er mit Alice spricht. Dementsprechend muss man Nachricht 3 zu Nachricht 1 in Beziehung setzen, d.h. den Schlüssel von der ursprünglichen Anfrage von Alice, einen Kanal mit Bob einzurichten, abhängig machen.

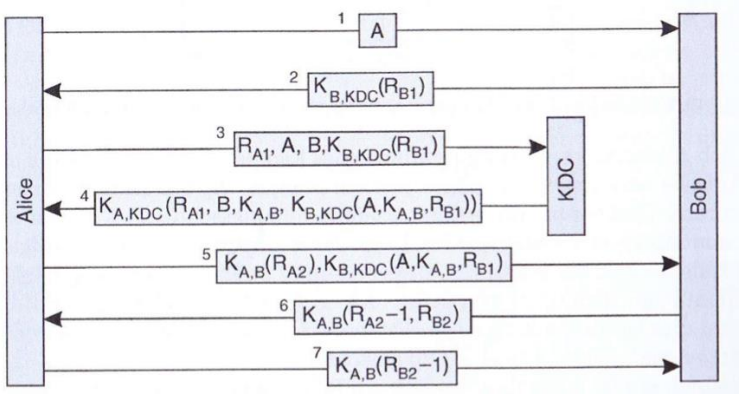

Abbildung 5: [TANA] Abbildung 9.18

Der Trick besteht darin, eine Nonce in der von Alice an das KDC gesandten Anfrage einzuschließen. Diese Nonce muss jedoch von Bob stammen: Das überzeugt Bob, dass wer auch immer es sei, der einen sicheren Kanal mit ihm einrichten möchte, die entsprechenden Informationen vom KDC erhalten hat. Daher bittet Alice zunächst Bob, ihr eine Nonce R_{B1} zu senden, verschlüsselt mit dem Schlüssel, den Bob gemeinsam mit dem KDC nutzt. Alice schließt diese Nonce in ihrer Anfrage an das KDC ein, das diese dann entschlüsselt und das Ergebnis in das erzeugte Ticket einfügt. Auf diese Art und Weise ist Bob sicher, dass der Sitzungsschlüssel an die ursprüngliche Anfrage von Alice, mit Bob zu reden, gebunden ist.

[TANA] 9.2.1 Authentifizierung über ein KDC, S. 436-439

3.3 Authentifizierung mit öffentlichen Schlüsseln

Hierbei brauchen die kommunizierenden Parteien keinen gemeinsamen Schlüssel zu kennen. Ein Benutzer erzeugt ein Schlüsselpaar, welches aus einem öffentlichen und einem privaten Schlüssel besteht. Der öffentliche dient zur Verschlüsselung und der private zur Entschlüsselung.

Nehmen wir an, dass Alice einen sicheren Kanal zu Bob einrichten will und beide im Besitz des öffentlichen Schlüssels des anderen sind. Alice beginnt, indem sie Bob eine Aufgabe R_A sendet, die sie mit seinem öffentlichen Schlüssel K_B^+ verschlüsselt. Es ist Bobs Aufgabe, die Nachricht zu entschlüsseln und die Aufgabe an Alice zurückzusenden. Da Bob der einzige ist, der die Nachricht (unter Verwendung des von Alice benutzten privaten Schlüssels) entschlüsseln kann, weiß Alice, dass sie es mit Bob zu tun hat.

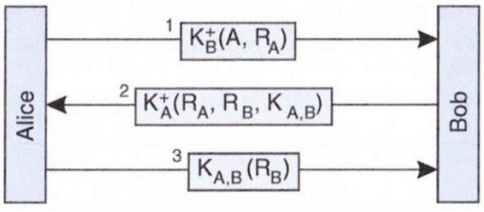

Abbildung 6: [TANA] Abbildung 9.19

Wenn Bob Alices Anfrage zur Einrichtung eines Kanals erhält, sendet er die entschlüsselte Aufgabe zusammen mit seiner eigenen Aufgabe R_B zur Authentifizierung von Alice zurück. Darüber hinaus erzeugt er einen Sitzungsschlüssel $K_{A,B}$, der zur weiteren Kommunikation genutzt werden kann. Bobs Antwort auf Alices Aufgabe, seine eigene Aufgabe und der Sitzungsschlüssel werden in eine Nachricht gepackt, die mit dem öffentlichen Schlüssel von Alice K_A^+ verschlüsselt wird. Nur Alice ist in der Lage, diese Nachricht mit dem zu K_A^+ zugehörigen privaten Schlüssel K_A^- zu entschlüsseln. Schließlich sendet Alice ihre Antwort auf

Bobs Aufgabe unter Verwendung des von Bob erzeugten Sitzungsschlüssels $K_{A,B}$ an Bob zurück. Auf diese Art beweist sie, dass die Nachricht 2 entschlüsseln konnte und dass es somit tatsächlich Alice ist, mit der Bob spricht.

[TANA] 9.2.1 Authentifizierung mit öffentlichen Schlüsseln, S. 440

3.4 Delegation

Häufig greifen Dienste auf andere Netzwerkdienste zu. Dabei greift der Client erstmals auf den ersten Dienst (Frontend) zu. Der muss seinerseits auf einen weiteren Netzwerkdienst (Backend) zurückgreifen, um die Anfrage des Clients zu beantworten. Das Frontend tritt gegenüber dem Nutzer in einer Service-Rolle, gegenüber dem Backend in einer Client-Rolle auf. Grob beschrieben ist Delegation das weitergeben von Tickets.

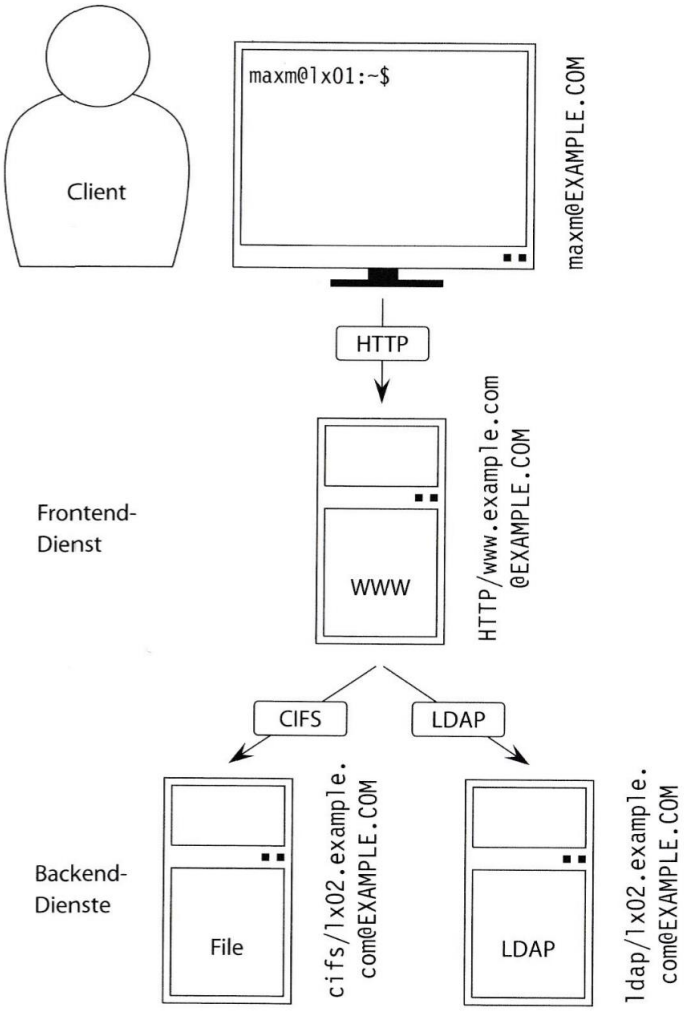

Abbildung 7: [PROM] S.32; Beispiel Delegation

4 Autorisierung

Neben der Authentisierung und Authentifizierung sind auch die Begriffe *Autorisierung* und Zugriffskontrolle wesentlich für jedes Sicherheitskonzept.

Beim Vorgang der *Autorisierung* wird festgelegt, mit welchen Berechtigungen Benutzer auf Ressourcen im Netzwerk zugreifen dürfen. Netzwerkdienste, die diese Ressourcen anbieten, führen im Allgemeinen eine *Zugriffskontrolle* durch. Dabei prüfen sie, ob der zugreifende

Benutzer autorisiert ist. Dementsprechend wird der Zugriff auf die Ressource erlaubt, verweigert oder nur eingeschränkt gewährt.

Damit das funktioniert, muss ein Dienst wissen, welchem Anwender er auf welches Objekt welche Aart von Zugriff erlauben kann. Welche *Autorisierungsinformationen* ein Dienst dafür benötigt und wo diese hinterlegt sind, hängt von dem betrachteten Dienst ab. Einige Beispiele:

- Dateidienste stellen Dateisysteme zur Verfügung. Die Dateien und Verzeichnisse darin sind mit sogenannten Access Control Lists (ACLs) versehen. Anhand dieser kann ein Datendienst entscheiden, ob ein bestimmter Anwender auf eine Datei oder ein Verzeichnis zugreifen darf und in welcher Form dieser Zugriff gegebenenfalls erfolgen kann (z.B. nur lesend, lesend und schreibend oder auch ausführend).
- Webanwendungen haben ebenfalls Zugriffsentscheidungen zu treffen. Beispielsweise lässt sich beim Apache-Webserver der Zugriff auf einzelne Verzeichnisse durch .htaccess-Dateien einschränken.
- Gerade bei Verzeichnisdiensten können die Zugriffsrechte sehr feingranular vergeben werden. Beispielsweise auf einem OpenLDAP-Server.

[PROM] 2.4 Autorisierung, Zugriffskontrolle und Namensdienste, S. 20/21

5 Nachrichtenintegrität & Vertraulichkeit

Neben der Authentifizierung sollte ein sicherer Kanal auch die Gewähr für Nachrichtenintegrität und Vertraulichkeit übernehmen. Vertraulichkeit lässt sich leicht herstellen, indem eine Nachricht einfach vor dem Senden verschlüsselt wird. Verschlüsselung kann entweder durch einen geheimen Schlüssel, der mit dem Empfänger gemeinsam genutzt wird, erfolgen oder alternativ durch Verwendung des öffentlichen Schlüssels des Empfängers. Eine Nachricht vor Änderungen zu schützen, ist allerdings etwas schwieriger.

5.1 Digitale Signaturen

Nachrichtenintegrität geht häufig über die eigentliche Übermittlung durch einen sicheren Kanal hinaus. Nehmen wir an, Bob verkauft Alice ein gewisses Objekt und das gesamte Geschäft verläuft über E-Mail-Verkehr und Alice bestätigt, dass sie das Objekt um den vereinbarten Preis kauft. Außer der Authentifizierung gibt es noch zwei Punkte die beachtet werden müssen.

- Alice muss zugesichert werden, dass Bob den vereinbarten Betrag nicht im Nachhinein böswillig ändert.
- Bob muss zugesichert werden, dass Alice nicht bestreiten kann, geschrieben zu haben das Objekt zu kaufen.

Diese Anforderungen lassen sich erfüllen, indem Alice die Nachricht digital so signiert, dass ihre Signatur eindeutig mit dem Inhalt verknüpft ist. Die eindeutige Verbindung zwischen einer Nachricht und ihrer Signatur verhindert, dass Änderungen an der Nachricht unbemerkt bleiben. Zudem kann Alice nicht zu einem späteren Zeitpunkt leugnen, dass sie die Nachricht signiert hat, wenn ihre Signatur als echt bestätigt werden kann.

Es gibt mehrere Arten des Anlegens digitaler Signaturen. Eine beliebte Form ist die Verwendung eines Verschlüsselungssystems mit öffentlichen Schlüsseln wie RSA.

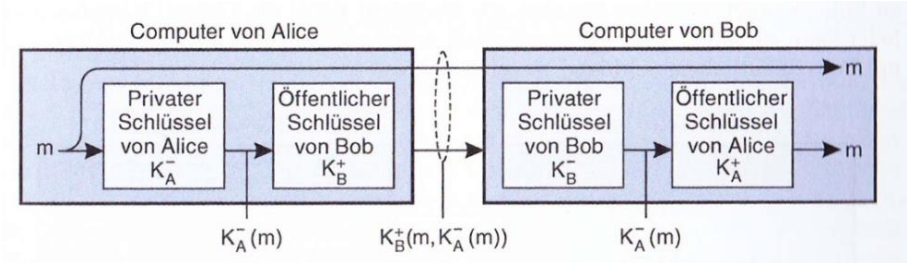

Abbildung 8: [TANA] Abbildung 9.20

Wenn Alice Bob eine Nachricht m sendet, verschlüsselt sie sie mit ihrem privaten Schlüssel K^-_A und schickt sie dann ab. Wenn sie will, dass der Inhalt der Nachricht geheim bleibt, kann sie Bobs öffentlichen Schlüssel verwenden und K^+_B (m. K^-_A (m)) senden. Dies kombiniert m und die von Alice signierte Version.

Wenn die Nachricht bei Bob ankommt, kann er sie entschlüsseln, indem er Alices öffentlichen Schlüssel verwendet. Wenn er sicher sein kann, dass der öffentliche Schlüssel tatsächlich Alice gehört, dann kann das Entschlüsseln der signierten Version von m und der gelungene Abgleich mit m nur bedeuten, dass sie von Alice kam. Alice ist vor böswilligen Änderungen von m durch Bob geschützt, weil Bob immer nachweisen müsste, dass die geänderte Version von m auch von Alice signiert wurde. Die entschlüsselte Nachricht allein zählt also nie im Eigentlichen als Beweis. Es ist auch in Bobs eigenem Interesse, die signierte Fassung von m aufzubewahren, um sich vor einem Rücktritt von Alice zu schützen.

[TANA] 9.2 Sichere Kanäle, S.437

5.2 Sitzungsschlüssel

Während der Einrichtung eines sicheren Kanals, nach Abschluss der Authentifizierungsphase benutzen die miteinander kommunizierenden Parteien im Allgemeinen der Vertraulichkeit willen einen eindeutigen, gemeinsamen Sitzungsschlüssel, welcher nach Vollendung der Kommunikation auf sichere Art verworfen wird.

Wieso nicht immer den selben Sitzungsschlüssel für alle Kommunikationen verwenden?

- Es wird leichter Schlüssel aufzudecken, je öfter sie vorkommen. Je mehr Daten man einem Eindringling mit dem selben Schlüssel liefert desto wahrscheinlicher ist es, dass er gewisse Eigenschaften des Schlüssels, die verschlüsselte Nachricht oder sogar den Schlüssel selbst herausfindet.
- Ein anderer Grund ist der Schutz vor Wiedereinspielungen. Wenn jedes Mal ein neuer Sitzungsschlüssel erzeugt wird, wird ein solcher Angriff sofort erkannt.
- Angenommen der Schlüssel ist nicht mehr sicher, kann der Eindringling alle früheren Nachrichten entschlüsseln. Wenn jedes Mal aber ein neuer Sitzungsschlüssel generiert wird, kann er im schlimmsten Fall auf eine einzige Sitzung zugreifen.

[TANA] 9.2 Sitzungsschlüssel, S.443

6 Sichere Gruppenkommunikation

Bisher haben wir uns auf die Einrichtung eines Kommunikationskanals zwischen zwei Parteien konzentriert. In verteilten Systemen ist es jedoch oft so, dass die Kommunikation zwischen mehr als nur zwei Parteien stattfindet.

6.1 Vertrauliche Kommunikation in Gruppen

Betrachten wir zunächst das Problem, die Kommunikation innerhalb einer Gruppe von N Benutzern vor Belauschen zu schützen.

Eine mögliche Maßnahme wäre einen gemeinsamen geheimen Schlüssel für alle Gruppenmitglieder zu verwenden. Das bringt jedoch das Problem mit sich, dass man bei dieser Maßnahme jedem Gruppenmitglied so weit vertrauen muss, dass man sich sicher sein kann, dass keines der Gruppenmitglieder den geheimen Schlüssel preisgibt.

Ein weiterer Ansatz wäre die Verwendung eines gemeinsamen geheimen Schlüssels für alle Paare von Gruppenmitgliedern. Das ruft jedoch das Problem hervor, dass man nun N(N-1)/2 Schlüssel verwalten müsste.

Der weitaus beste Ansatz ist die Verwendung von öffentlichen Schlüsseln. Jedes Gruppenmitglied hat sein eigenes Schlüsselpaar (öffentlicher Schlüssel und privater Schlüssel), von denen der öffentliche Schlüssel von allen Mitgliedern zum Versenden vertraulicher Nachrichten verwendet werden kann. Das bedeutet, dass insgesamt N Schlüsselpaare benötigt werden. Wenn ein Mitglied nicht mehr vertrauenswürdig ist, wird es einfach aus der Gruppe entfernt, ohne dass es die anderen Schlüssel preisgeben kann.

[TANA] 9.2.3 Sichere Gruppenkommunikation, S. 444

7 Angriffspotenzial

7.1 Netzwerkebene

7.1.1 Repay-Angriff

Siehe Ende einstufiges Kerberos Verfahren (kurze Erklärung)

7.2 Applikationsebene

7.2.1 Wörterbuch- bzw. Brute-Force-Attacke

Siehe Ende einstufiges Kerberos-Verfahren (kurze Erklärung)

8 RADIUS

8.1 Allgemein

RADIUS steht für **Remote Authentication Dial-In User Service** und ist ein Client-Server-Protokoll, welches zur Authentifizierung, Autorisierung und zum Accounting (AAA-System) von Benutzern bei Einwahlverbindungen in ein Computernetzwerk dient. RADIUS ist De-facto-Standard bei der zentralen Authentifizierung von Einwahlverbindungen über Modem, ISDN, VPN, WLAN und DSL.

8.2 Anwendung

8.2.1 IEEE 802.1x / WPA2

Im Zusammenhang mit WLAN wird die Authentifizierungsmethode IEEE 802.1x auch als WPA2-Enterprise, WPA2-1x oder WPA2/802.1x bezeichnet. Dieser Standard schreibt keinen RADIUS-Server vor, doch in der Regel wird beim Einsatz der Zugangskontrolle mit IEEE 802.1x auch ein RADIUS-Server eingesetzt.

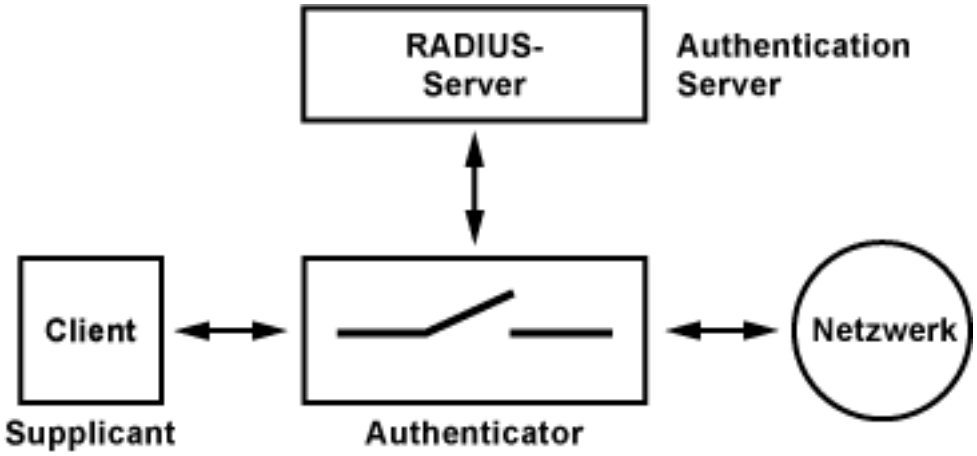

Abbildung 9: WPA2

Anmeldungen vom Supplicant (Client) werden vom Authenticator zuerst an den Authentication Server weitergeleitet. Der entscheidet, ob der Supplicant Zugang bekommt. In Abhängigkeit einer erfolgreichen Authentifizierung wird der Zugang zum Netzwerk über einen bestimmten Port freigeschaltet. Für IEEE 802.1x kann ein Port eine Buchse an einem Switch oder eine logische Assoziation sein. Denkbar ist hier die Zugangsmöglichkeit zum Netzwerk für einen WLAN-Client an einem WLAN-Access-Point. Mit IEEE 802.1x/EAP wird dem WLAN-Client zu Beginn einer Sitzung die dafür gültigen WPA2-Schlüssel mitgeteilt.

8.2.2 Funktionsweise von RADIUS

Beim Zugriff auf ein lokales Netzwerk eines Unternehmens über WLAN reicht die einfache Authentifizierung über ein gemeinsames Passwort (WPA2-PSK) nicht aus. Wenn das Passwort die Runde macht, dann ist das WLAN praktisch offen.
Mit RADIUS werden serverseitig Passwörter zugeteilt, was dem Administrator Arbeit erspart und für die Nutzer vergleichsweise einfach ist. In dieser Konstellation kommt WPA2-Enterprise zum Einsatz, bei dem die WLAN-Basisstation die Zugriffe der WLAN-Clients über das Protokoll IEEE 802.1x mit einem RADIUS-Server aushandeln.
Ein RADIUS-Server ist nicht immer zwingend erforderlich. Manche WLAN-Router enthalten bereits einen RADIUS-Server, der für kleine Netzwerke eine Alternative ist.

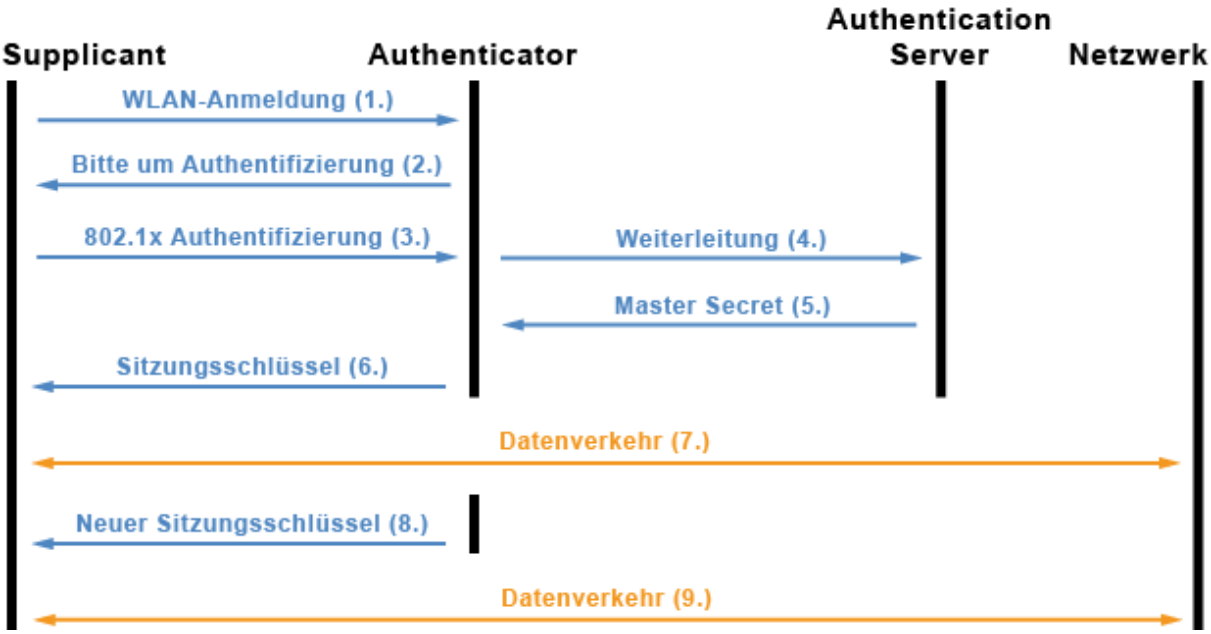

Abbildung 10: http://www.elektronik-kompendium.de/sites/net/1409281.htm

- Zuerst meldet sich der WLAN-Client (Supplicant) am WLAN-Access-Point (Authenticator) an. Beide Geräte sind entsprechend auf WPA2-Enterprise konfiguriert.
- Der Access-Point (Authenticator) fordert den Client (Supplicant) zur Authentifizierung auf. In der Regel folgt hier die Eingabe von Benutzername und Passwort durch den Nutzer.
- Der Client (Supplicant) authentisiert sich nach IEEE 802.1x.
- Der Access-Point (Authenticator) leitet die Authentifizierung an den RADIUS-Server (Authentication Server) weiter.
- Bei erfolgreicher Authentifizierung gibt der RADIUS-Server das Master Secret zurück.
- Der Access-Point generiert den Sitzungsschlüssel und teilt diesen dem Client mit.
- Durch den Sitzungsschlüssel bekommt der Client Zugriff auf das Netzwerk.
- In regelmäßigen Abständen bekommt der Client einen neuen Sitzungsschlüssel mitgeteilt.
- Damit ist weiterhin der Zugriff auf das Netzwerk durch den Client möglich.

http://www.elektronik-kompendium.de/sites/net/1409281.htm

8.3 Wie sicher ist RADIUS?

Die per RADIUS verwendeten Zugangsdaten (Benutzername und Passwort) sind normalerweise nicht sicherer als zum Beispiel ein WLAN-WPA2-Passwort. Eine höhere Sicherheit erreicht man nur durch den Einsatz zusätzlicher Zertifikate über EAP-TLS. Hierbei identifizieren sich RADIUS-Server und Client gegenseitig. Der dafür notwendige Einrichtungsaufwand sollte nicht unterschätzt werden. Selbst große Unternehmen betreiben diesen Aufwand nicht.

http://www.elektronik-kompendium.de/sites/net/1409281.htm

8.4 RADIUS vs. LDAP

Wieso sollte man RADIUS verwenden und nicht LDAP?

- RADIUS ist im vergleich zu LDAP sehr einfach aufzusetzen und zu konfigurieren.
- RADIUS ist erweiterbar
  - Man kann das Schema mit weiteren Attributen erweitern.
  - Man kann so gut wie jede Datenbank für die Authentifikation benutzen.
  - Man kann so gut wie jedes Authentifizierungsprotokoll benutzen.
- RADIUS inkludiert Accounting. Das ist nichts Anderes als eine Historie die zum Beispiel die Funktionalität bietet, einem Benutzer nur eine gewisse Zeit lang Zugang zum Netzwerk zu gewähren.

8.5 Implementierungen

Die zwei bekanntesten Implementierungen:

8.5.1 FreeRADIUS

http://freeradius.org

8.5.2 OpenRADIUS

http://www.openradius.net/

9 Einführung Kerberos

Kerberos ist in der griechischen Mythologie ein dreiköpfiger Hund, welcher den Eingang zum Reich der Toten bewacht. Seine Aufgabe ist es sicherzustellen, dass weder ein Toter herauskommt noch ein Lebender hereinkommt. Der Authentifizierungsdienst bzw. -protokoll hat seinen Ursprung in einem Projekt, welches 1983 am MIT startete. Das Athena-Projekt. Dabei ging es um die Vernetzung von Arbeitsplatzrechnern. Dieses Projekt war neben Kerberos auch noch die Grundlage für mehr als 100 Softwareprojekte. Das Athena-Netzwerk stellte auch ganz neue Herausforderungen an die Sicherheit. Anstatt sich “nur” mehr um die Sicherheit der wenigen Großrechner zu bemühen, mussten jetzt wesentlich mehr Systeme in die Sicherheitsanalyse einbezogen werden. Dafür wurde nun Kerberos entwickelt. Ein Dienst welcher sich in erster Linie mit der Autorisierung beschäftigt. Es deckt aber auch andere Sicherheitsaspekte ab welche später erläutert werden.
Es gibt 5 Versionen von Kerberos, wobei Kerberos v4 die erste Version war, welche auch außerhalb des MITs verwendet wurde.
Die wichtigsten Implementierungen sind derzeit MIT Kerberos, Heimdal und Active Directory. (Sind alle Kerberos v5 Implementierungen).

9.1 Lokale Anmeldung

Innerhalb einer Active-Directory-Domäne wird Kerberos für die Passwortüberprüfung bei einer lokalen Anmeldung an Windows-Rechnern verwendet.

9.2 Autorisierung

Kerberos wird zur Authentifizierung verwendet. Zur Autorisierung sollte man ein Verzeichnis- oder Namenssystem wie beispielsweise LDAP verwendet. Diese Trennung ist für ein sicheres Systemdesign zu befürworten.

10 Sicherheitsüberlegung

10.1 Anforderungen an Kerberos

Die Zielumgebung für den Einsatz von Kerberos besteht aus nicht vertrauenswürdigen Komponenten (Netzwerk, Arbeitsplatzrechner & Server). Die einzige Voraussetzung ist lediglich die Existenz von vertrauenswürdigen Maschinen, auf denen der Kerberos-Authentisierung läuft. Dadurch lassen sich die Anforderungen wie folgt zusammenfassen:

- **Anwenderauthentisierung** Kerberos soll Clients gegenüber Netzwerkdiensten authentisieren.
- **Gegenseitige Authentisierung** Kerberos soll die Authentisierung von Netzwerkdiensten gegenüber deren Clientprogrammen durchführen.
- **Single Sign-on** Anwender sollen nur zu Beginn ihrer Sitzung mit einem Passwort konfrontiert werden.
- **Keine Klartextpasswörter übers Netz** Passwörter und dazu äquivalente Informationen dürfen nicht über das Netzwerk übertragen werden. Auch die verschlüsselte Übertragung von Passwörtern über das Netzwerk ist im Rahmen von Kerberos-Authentisierung gängig nicht vorgesehen. Stattdessen führt Kerberos kurzlebige Authentisierungsmerkmale (Kerberos Tickets) ein.
- **Keine Anwenderpasswörter auf Workstations speichern** Es dürfen keine Anwenderpasswörter auf nicht vertrauenswürdigen Workstations gespeichert werden. Um dem Anwender das mehrmalige Eintippen von Passwörtern abzunehmen, ist die Versuchung sehr nahe. Nur auf stark abgesicherten und daher vertrauenswürdigen Authentisierungsdiensten werden solche Passwortinformationen gespeichert.
- **Keine Anwenderpasswörter auf Serversystemen** Da man auch die Serversysteme (mit der genannten Ausnahme der Authentifizierungsdienste) als nicht vertrauenswürdig einstufen muss, dürfen auch hier keine Anwenderpasswörter gespeichert werden.
- **Sicherheitsschicht** Um das Manko der unsicheren Netzwerkverbindungen auszugleichen, müssen sichere Datenkanäle bereitgestellt werden. Kerberos soll dazu die notwendige Infrastruktur für Datenintegrität und Datenvertraulichkeit zur Verfügung stellen.

10.2 Randbedingungen

Damit die Authentisierung über Kerberos auf eine sichere Art und Weise erfolgen kann, sind einige Randbedingungen zu erfüllen:

- Die Systeme auf denen der Authentifizierungsdienst läuft sind zu sichern. Dies stellt zunächst einen gewissen administrativen Aufwand dar, allerdings besteht darin auch der Vorteil, dass nur sehr wenige Systeme stark abgesichert werden müssen.
- Benutzerpasswörter dürfen nur dem Authentifizierungsdienst und dem Benutzer bekannt sein und auf keinen weiteren Maschinen zwischengespeichert werden.
- Anwender müssen mit ihren Passwörtern sorgfältig umgehen. Andernfalls bietet ein weitergegebenes bzw. gestohlenes Passwort ein großes Angriffspotential.

11 Begriffe – Kerberos

11.1 Realm

Kerberos Umgebungen werden aus administrativen Gründen in sogenannte Realms unterteilt. Diese entsprechen typischerweise einer Organisation oder Teilen einer Organisation. In diesen Realms gibt es KDCs, welche für die Authentisierungsvorgänge zuständig sind. Die Authentisierung ist aber auch zwischen unterschiedlichen Realms möglich (Cross-Realm-Authentisierung). Normalerweise entsprechen die Realm-Namen dem DNS-Namen in Großbuchstaben. Im Gegensatz zu den Regeln der Namensgebung bei DNS wird hier jedoch zwischen Klein- und Großschreibung unterschieden. Ansonsten gelten die gleichen Regeln.

11.2 Principals

Innerhalb eines Kerberos Realm werden Clients und Dienste mit sogenannten Kerberos Principals repräsentiert. Die Syntax eines für ein Principal sieht wie folgt aus:

Komponente1[/Komponente2/…/KomponenteN]@REALM

Bei einem allgemeinen Authentisierungsaustausch sind folgende Teilnehmer beteiligt:

- Client Principal:
  benutzername@REALM
  schueler@EXAMPLE.COM
- Dienst-Principal:
  dienst/server@REALM
  HTTP/www.example.com@EXAMPLE.COM
- Der Kerberos Dienst (als KDC)

12 Funktionsweiße Kerberos v5

12.1 Voraussetzung

Jedem Client und jedem Dienst muss eine Kerberos-Identität in Form eines Principals zugeordnet sein. Außerdem müssen Client und Dienst jeweils einen oder mehrere kryptographische Langzeitschlüssel besitzen (Client- /Service Key). Die nächste Voraussetzung ist, dass es einen Kerberos-Server (KDC) gibt. Eine wichtige Komponente eines KDCs ist die KDC-Datenbank, in der es zu jedem Kerberos Principal des Realms einen Eintrag geben muss. Außerdem müssen die Uhrzeiten der Systeme einigermaßen synchron sein. Weiter sollte die Hostnamenauflösung auf allen beteiligten Systemen identisch funktionieren.

12.2 Einstufiges Kerberos-Verfahren

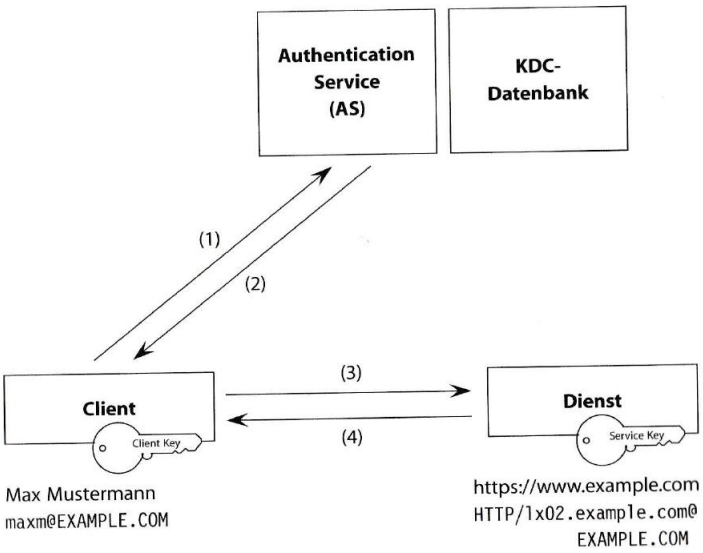

Abbildung 11: [PROM] S. 68; Das einstufige Kerberos Verfahren

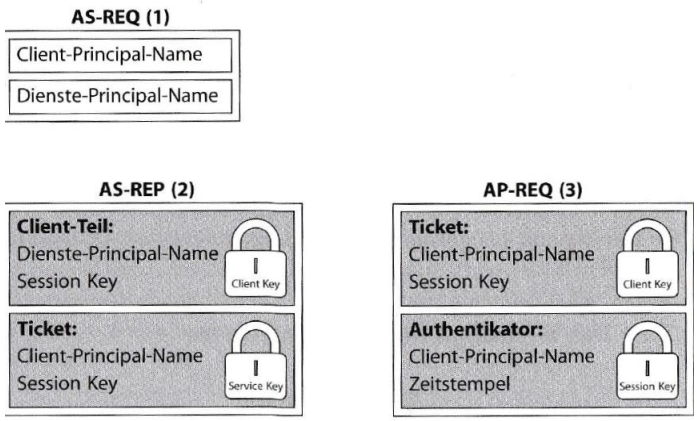

Abbildung 12: [PROM] S. 68; Inhalte der in Abbildung 2 ausgetauschten Nachrichten

Zuerst schickt der Client eine Anfrage an das KDC, den Authentication Service Request. Neben einigen anderen Informationen beinhaltet diese Anfrage im Wesentlichen:

- Den Namen des Client Principals
- Den Principal-Namen Dienstes auf den er zugreifen will

Auf diese Anfrage reagiert der AS nun folgendermaßen:
Zunächst erzeugt er einen zufälligen Sitzungsschlüssel (Session Key) für die beiden Principals. Dann sucht er nach deren beiden Langzeitschlüsseln in der KDC-Datenbank. Aus diesen Informationen konstruiert er die Nachricht 2 (Authentication Service Reply). Der Wesentliche Anteil dieser Nachricht besteht aus zwei verschlüsselten Teilen:

- Der erste Teil besteht aus dem Principalnamen des Dienstes und dem Session Key, beides verschlüsselt mit dem Langzeitschlüssel des Clients. In der Abbildung 3 als Client-Teil definiert.
- Der zweite Teil beinhaltet den Principalnamen des Clients und den Session Key, beides verschlüsselt mit dem Langzeitschlüssel des Dienstes. Dieser Teil ist das Ticket für den Dienst.

Wenn der Client die Nachricht 2 erhält entschlüsselt er den Client-Teil mit dem Client Key. Da er diesen Langzeitschlüssel nicht gespeichert hat fragt er den Benutzer nach dem Passwort um dieses mittels string2key-Funktion in den Client Key umzuwandeln. Nach dem Entschlüsseln ist ein wichtiger Schritt getan: der Client kennt jetzt den Session Key.
Mit diesem Session Key kann der Client nun einen Authentikator konstruieren. Dieser besteht aus einem aktuellen Zeitstempel des Clientsystems und dem Principalnamen des Clients. Der Client verschlüsselt diese beiden Informationen mit dem Session Key und erhält so einen Authentikator.
Nun hat der Client alles vorbereitet, um auf den Dienst mittels Application Server Request (Nachricht 3) zuzugreifen. Der AP-REQ enthält das Ticket und den Authentikator.

Erhält der Dienst die Nachricht 3 so kann er zuerst das Ticket entschlüsseln und kennt dadurch ebenfalls den Session Key. Außer dem Dienst, Client und dem Vertrauenswürdigen KDC kennt niemand den Session Key!
Die Authentifizierung des Clients erfolgt über den Authentikator. Mit dem Session Key kann der Dienst diesen entschlüsseln. Findet er darin den Principalnamen des Clients und einen gültigen Zeitstempel so kann er den Client als authentisch betrachten, denn ein dritter hätte den Authentikator ohne Session Key nicht erzeugen können.
Falls der Client auf gegenseitige Authentifizierung besteht so geschieht dies mittels Nachricht 4, dem Application Service Reply.

Angreifer können Nachricht 3 auf dem Netzwerk mithören und das Erlauschte später selbst verwenden. Gelingt dies innerhalb der Clock Skew (Gültiger Zeitraum des Zeitstempels), so wäre ein sogenannter Replay-Angriff erfolgreich. Damit das nicht geschieht muss sich der Dienst die bereits akzeptierten Authentikatoren im sogenannten Replay Cache speichern. [PROM] S 67 „Das einstufige Kerberos-Verfahren“

Um zu verhindern, dass Angreifer durch das Empfangen der AS-REP das verschlüsselte Client Passwort erhalten gibt es die Möglichkeit der Pre-Authentification. Dies bedeutet, dass der Client seinen Key mit der AS-REQ mitschicken muss um eine AS-REP zu erhalten. Andernfalls könnten Angreifer mittels Wörterbuch- oder Brute-Force-Attacke das Anwenderpasswort entschlüsseln.

12.3 Zweistufiges Kerberos-Verfahren

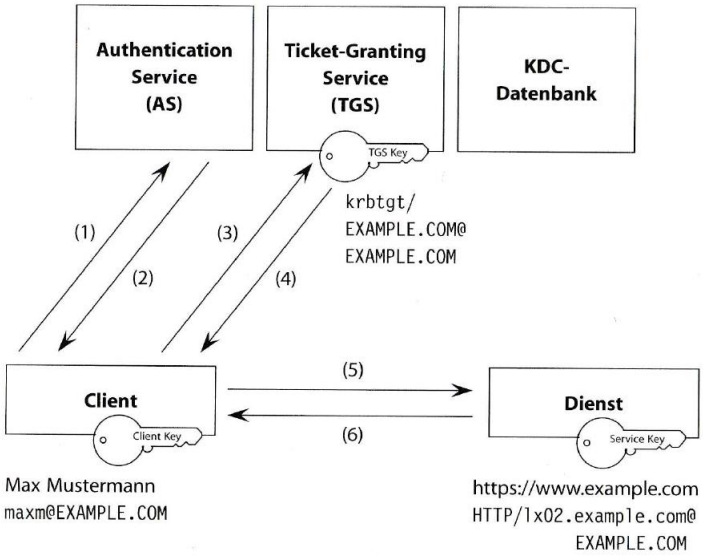

Abbildung 13: [PROM] S.71; Das zweistufige Kerberos-Verfahren

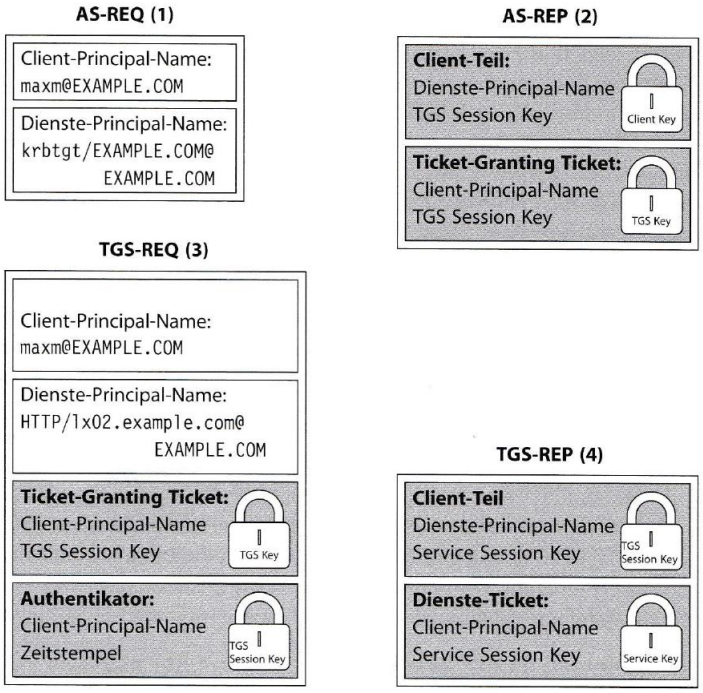

Abbildung 14: [PROM] S.73;Inhalte der in Abbildung 4 ausgetauschten Nachrichten

Im Gegensatz zum einstufigen Kerberos-Verfahren enthält das KDC hier auch einen Ticket Granting Service (TGS). Das Ticket für den TGS wird TGT genannt. Der TGS hat, wie auch der AS, Zugriff auf die KDC-Datenbank und übernimmt somit die gleichen Aufgaben wie der AS.

- Der TGS beantwortet Clientanfragen
- Der TGS erzeugt zufällige Session Keys
- Der TGS erzeugt Kerberos Tickets

Der einzige Unterschied zum AS ist, dass Clients zum Zugriff ein TGT benötigen, während sie für den AS den Client Key (Passwort) benötigen.

Um Single Sign-On zu realisieren holen sich Clients zunächst vom AS ein TGT – wozu einmal das Passwort notwendig ist. Weitere Tickets bekommen sie dann vom TGS - ohne weitere Anwenderinteraktionen.
Der Ablauf stellt sich nun wie folgt dar: Der Client beantragt vom AS ein TGT. In der ersten Nachricht befindet sich der Pricipalname des Clients sowie der Principalname des Dienstes. Der AS sendet dann eine Nachricht zurück (2). Diese kann in zwei Teile geteilt werden. Der Client-Teil besteht aus dem Prinicipalnamen des TGS und dem TGS Session Key. Beides Verschlüsselt mit dem Langzeitschlüssel des Clients. Der zweite Teil (TGT) setzt sich aus dem Principalnamen des Clients und dem TGS Session Key. Beides verschlüsselt mit dem Langzeitschlüssel des Dienstes (TGS Key).
Der Client entschlüsselt diese Nachricht nun mit dem Client Key (Passworteingabe vom Benutzer) und gelangt so an den Session Key. Den TGS Session Key und das TGT speichert der Client nun in seinem Credential Cache.
Bis her war alles gleich wie beim einstufigen Verfahren, mit dem kleinen Unterschied, dass es sich hier um den TGS anstelle des AS gehandelt hat.
Um auf einen Dienst zugreifen zu können muss der Client zunächst einen Authentikator erzeugen. Dazu verschlüsselt er seinen Prinzipalnamen und einen aktuellen Zeitstempel mit dem TGS Session Key. Diese Anfrage heißt Ticket-Granting-Server Request (TGS-REQ). Analog zum AS-REQ erhält sie den Namen des Client Principals, sowie den Namen des Dienstes, für den der Client das Ticket ausgestellt haben möchte.
Erhält der TGS den TGS-TEQ, so prüft er das TGT und den Authentikator. Passt alles, dann hat der TGS den Client authentifiziert. Er stellt dann einen neuen Session Key für Client und Dienst. Dieser soll hier Service Session Key genannt werden. Außerdem entnimmt der TGS der KDC-Datenbank den Langzeitschlüssel des Dienstes (den Service Key).
Die Antwort des TGS an den Client heißt Ticket-Granting Service Reply (TGS-REP). Der Inhalt ähnelt dem des AS-REP. Auch der TGS-REP besteht aus zwei verschlüsselten Teilen. Dem Clienteil, also dem Principalnamen des Dienstes und dem Service Session Key. Dieser Inhalt wird mit dem TGS Session Key verschlüsselt. Der zweite Teil (Service Ticket) besteht aus dem Principalnamen des Dienstes und dem Service Session Key. Dieser Teil ist mit dem Langzeitschlüssel des Dienstes verschlüsselt.
Durch das Entschlüsseln gelangt der Client an den Service Session Key. Anschließend kann er gleich wie beim einstufigen Kerberos Verfahren auf den Dienst zugreifen.
Gegenüber dem einstufigen Kerberos-Verfahren hat der Client nun den Vorteil, dass er auf weitere Dienste zugreifen kann ohne dabei Schritt 1 und 2 erneut durchlaufen zu müssen. Für 3 und 4 ist keine Anwenderinteraktion nötig, das einmal beim Erhalten der Nachricht 2 eingegebene Passwort genügt. Single Sign-on ist also möglich.
[PROM] S71; „Das zweistufige Kerberos-Verfahren“

12.4 Ticket Flags

- **(R) Renewable**
  Kennzeichnet das Ticket als erneuerbar. Der Client darf daher ein neues TGT mit einem späteren Ablaufdatum anfordern.
- **(D) Allow-Postdate**
  Ist Voraussetzung dafür, dass der TGS Tickets mit einem in der Zukunft liegenden Startdatum ausstellen kann.
- **(F) Forwardable**
  Gibt die Erlaubnis, dass der TGS ein neues TGT an ein anderes Zielsystem ausstellen darf.
- **(f) Forwarded**
  Markiert, dass das Ticket auf Basis eines weitergegebenen Tickets ausgestellt wurde.
- **(P) Proxiable**
  Gibt die Erlaubnis, dass der TGS ein neues TGT an ein anderes Zielsystem ausstellen darf.
- **(p) Proxy**
  Markiert, dass das Ticket auf Basis eines weitergegebenen Tickets ausgestellt wurde.
- **(0) Ok-As-Delegate**
  Wird in Dienst-Tickets gesetzt, wenn die Dienste so vertrauenswürdig sind, dass man ihnen Tickets weiterleiten kann.
- **(I) Initial**
  Tickets vom AS werden damit gekennzeichnet, Tickets vom TGS nicht. Dadurch kann man nachvollziehen von wo der Client das Ticket bezogen hat.
- **(A) Pre-Authenticated**
  Dieses Flag wird gesetzt, wenn der AS-REQ mit einer Pre-Authentication verbunden war.
- **(T) Transited-Policy-Checked**
  Diese Ticket Flag spielt eine Rolle, wenn es um Cross-Realm-Authentisierung geht.
- **(H) HW-Authent**
  Wiese Flag wird bei einem hardwarebasierten Authentisierungsvorgang gesetzt.

12.5 Tickets automatisiert erneuern

Es besteht die Möglichkeit ein Ticket ohne Anwenderinteraktion erneuern zu lassen. Dazu muss Client initial ein erneuerbares Ticket anfordern. Das KDC muss dem Client dann ein TGT mit gesetzter R Flag ausgeben. Das Ticket kann dann erneuert werden solange das Ticket Gültig ist und das Ticket Renew Lifetime auch noch nicht abgelaufen ist. Weiter kann das Ticket nie über die Renew Lifetime hinaus gültig sein. So bringt es nichts das Ticket kurz davor zu erneuern, da es dann nicht nochmal die eigentliche Lifetime des Tickets erhält, sondern nur mehr die Zeit welche von der Renew Lifetime über ist.

12.6 Tickets für die Zukunft

Eine weitere Option für Tickets ist es Tickets anzufordern, welche erst in der Zukunft gültig sein solle. Dazu wird vom KDC ein Postdated Ticket mit einer gesetzten invalid Flag ausgegeben. So hat der Client quasi ein Ticket, kann es jedoch (noch) nicht benutzen. Um dieses Ticket jetzt zu aktivieren muss es mittels Validate Request gegen ein gültiges Ticket eingetauscht werden. Dies kann jedoch nur erfolgen, wenn der Zeitpunkt der Gültigkeit eingetreten ist.

12.7 Delegation unter Kerberos

12.7.1 Ticket Forwarding

Die TGT-Weiterleitung beim Ticket Forwarding ist der einfachste Delegationsmechanismus bei Kerberos v5. Dabei schickt der Client dem Dienst mit dem AP-REQ ein TGT und den zugehörigen Session Key. Mit diesen Credentials kann der Dienst vom TGS beliebige Backend-Dienste-Tickets im Namen des Clients beziehen und sich damit jedem weiteren Dienst der Kerberos Umgebung als dieser Client ausweisen. Dem Dienst steht durch das weitergeleitete TGT die Identität des Clients also uneingeschränkt zur Verfügung.
Der Client fordert vom TGS ein weiteres TGT an. Dieses leitet er an den Dienst weiter. Dieser kann nun mittels diesem TGTs weitere Tickets zum Zugreifen auf Dienste vom TGS anfordern.

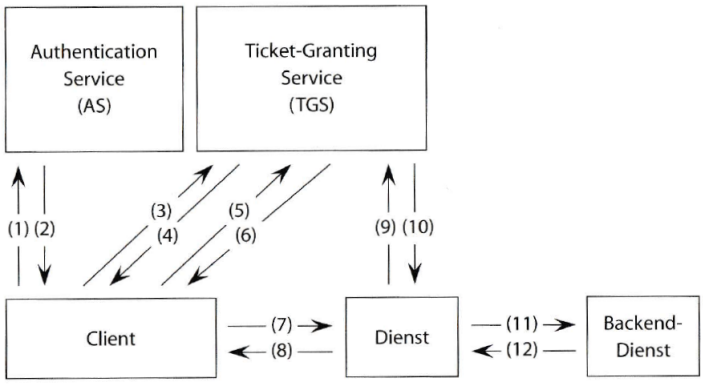

Abbildung 15: [PROM] S.71; Ticket Forwarding

12.7.2 Ticket Proxying

Ticket Proxying funktioniert ganz ähnlich wie das Forwarding. Allerding werden dabei TGTs weitergeleitet. Stattdessen übergibt der Client dem Zielsystem direkt das Ticket für den Backend-Dienst.
Im Gegensatz zum Forwarding schränkt Proxying die Verwendung der Clientidentität ein, da der Dienst nicht auf beliebige Backends zugreifen kann, sondern nur auf diejenigen, für die ihm der Client Tickets weiterleitet. Der Nachteil davon ist es, dass der Client genau wissen muss, auf welche Backend-Dienste der Dienst zugreift. Aus diesem Grund gibt es kaum Anwendungen, die Proxying verwenden.

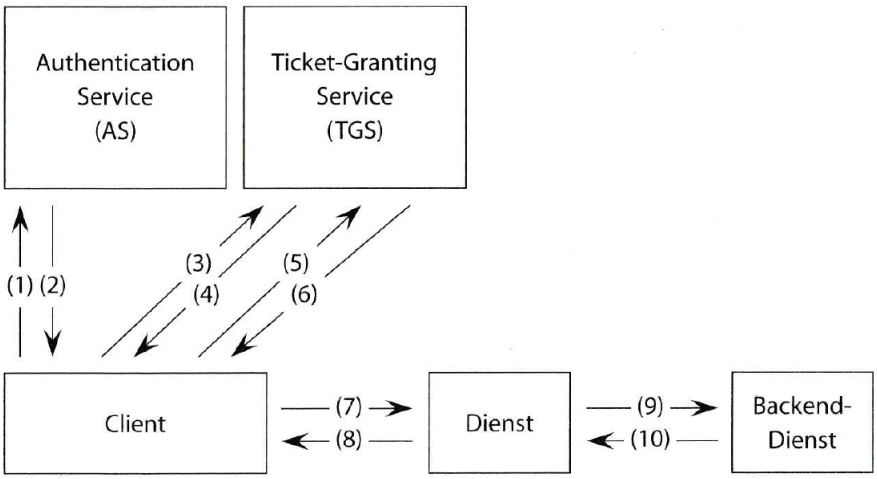

Abbildung 16: [PROM] S.71; Ticket Proxying

12.7.3 Constrained Delegation

Bei der Constrained Delegation greift der Client ganz „normal“ mir seinen Creditionals auf den Dienst zu. Dem Dienst liegt so lediglich das Dienste-Ticket des Clients vor, an dem er den Client authentifizieren kann.

Der Dienst greift nun auf das KDC zu und fordert das Ticket für den Backend-Dienst an. Diese TGS-REQ enthält das TGT des Dienstes, mit dem er sich authentifiziert, und das Dienste-Ticket des Clients. Wenn nun beide berechtigt sind erhält der Dienst das Backendticket für die Identität des Clients.
Der Backend-Dienst registriert bei einem Zugriff dabei dann einen Zugriff durch den Client.

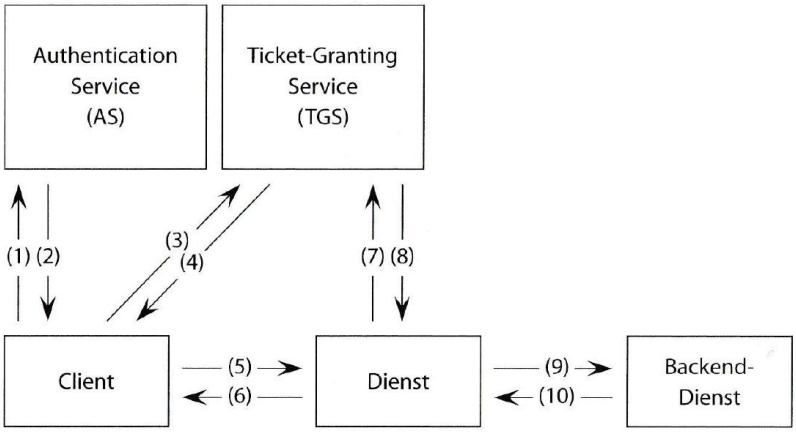

Abbildung 17: [PROM] S.71; Constrained Delegation

12.7.4 Protocol Transition

Bei der Protocol Transition authentifiziert sich der Client gegenüber dem Dienst über ein beliebig anderes Authentisierungsprotokoll als Kerberos. Anders als vorhin liegt dem dienst nun kein Client-Ticket vor.

Nachdem der Dienst den Client authentifiziert hat, stellt er eine Anfrage an den TGS, authentisiert sich und fragt nach einem Dienste-Ticket für sich selbst, das aber in der Identität des Clients ausgestellt werden soll. Danach entspricht die Ausgangslage wieder derjenigen bei der Constrained Delegation, nur, dass das Dienste-Ticket nicht vom Client, sondern vom KDC stammt.

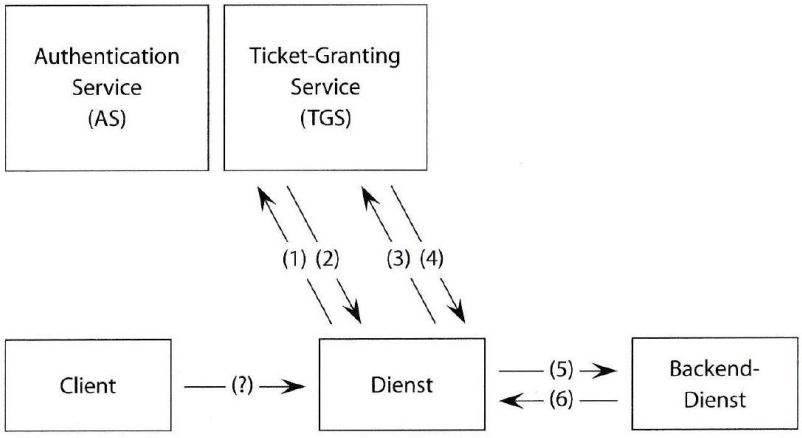

Abbildung 18: [PROM] S 72; Protocol Transition

12.8 Authentisierung zwischen Realms

Um eine Authentifizierung zwischen mehreren Realms zu ermöglichen muss ein Vertrauensverhältnis, bezeichnet als Cross-Realm Trust, bestehen. Der Dienste-Realm muss also darauf vertrauen können, dass der Realm des Clients die Authentifizierung ordentlich durchführt. Man bezeichent den Realm des Dienstes auch als vertrauenden Realm und den des Clinets als vertrauter Realm. Symbolisch dargestellt werden Trusts durch Pfeile. (sehe Abb. 17)
Die Vertrauensverhältnisse zwischen Realms werden mit kryptografischen Mitteln abgesichert. Dazu muss man zwischen zwei vertrauten Realms symetrische Schlüssel, sogenannte Inter-Realm-Keys, einrichten.

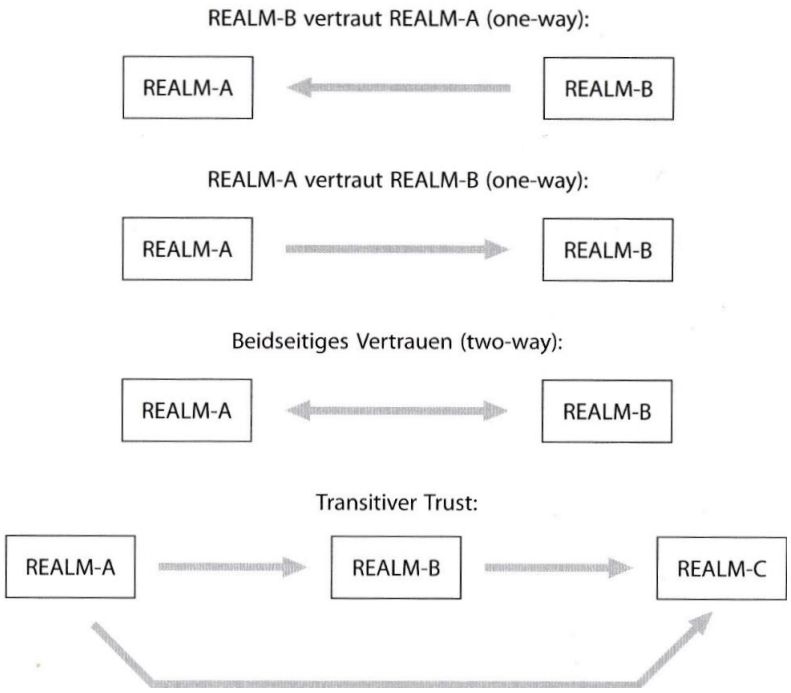

Abbildung 19: [PROM] S. 107; Trust zwischen Kerberos Realms

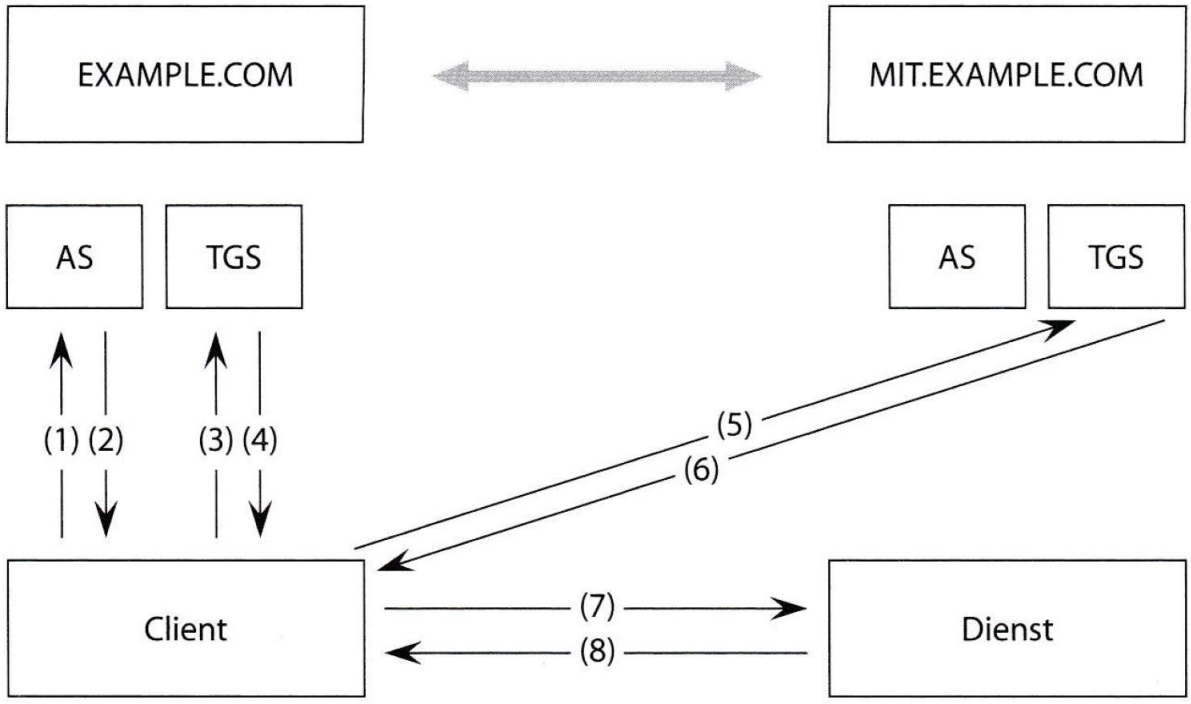

Abbildung 20: [PROM] S. 108; Cross-Realm-Authentisierung zwischen zwei Kerberos Realms

Ablauf Abbildung 20:
Der Client authentifiziert sich gegenüber dem AS
Fordert Ticket für den anderen Realm an
TGS sendet cross-Realm Ticket
Client fordert ticket bei zweitem TGS an mittels eben erhaltenen Ticket
Zweites TGS schickt Dienst-Tickt an Client
Client kann mit dem Dienst kommunizieren

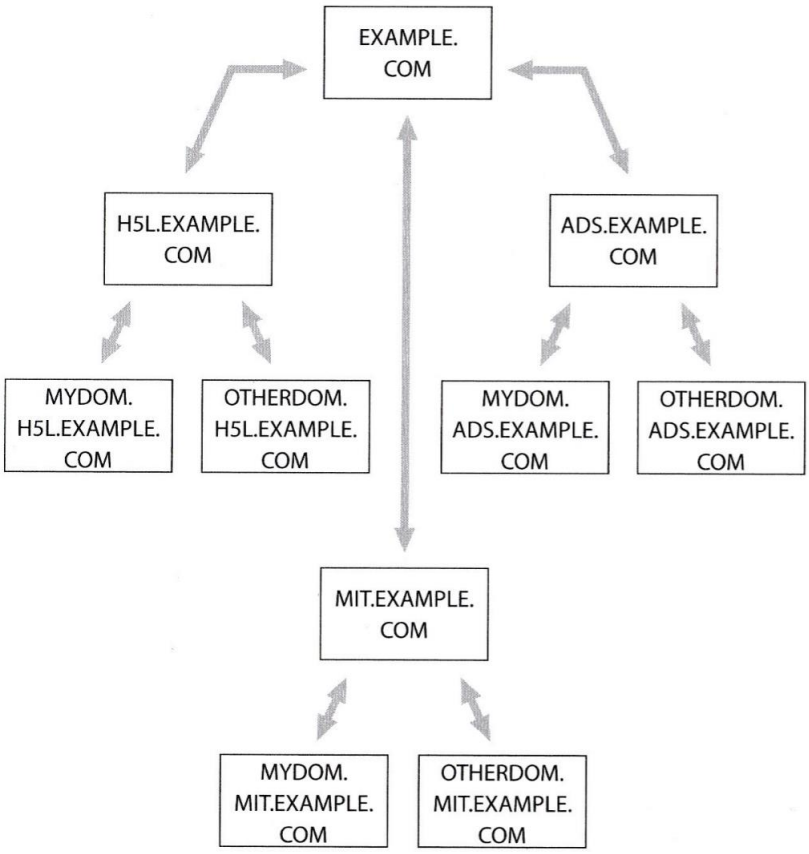

Abbildung 21: [PROM] S. 109; Cross-Realm Trusts zwischen Keberos Realms

Bei mehr als 2 Realms wird der Authentificaton Path länger und es werden mehr TGTs für die weiteren Realms ausgeteilt.

13 Einfaches Beispiel

Anmeldung am Rechner → erhält TGT (Ticket Granting Ticket)
Im Credential Cach werden die Tickets abgelegt → mit klist [-f] auslesbar
Anmelden an einem Netzwerkdienst, zum Beispiel LDAP oder SSH, braucht kein Passwort mehr → bekommt dafür jeweils ein eigenes Ticket
→So hat man bei ssh aber auf dem neuen Rechner kein Ticket mehr → Delegation wäre super → Lösung: Ticket forwarding welches das TGT übergibt (Sicherheitsproblem… man muss dem dienst und dem Server auf welchem er läuft vertrauen können!) SSH –K erlaubt –k verbietet (kann aber auch in der ~/.ssh/config fix eingestellt werden.

Webbrowser (in diesem Beispiel Firefox) können auch kerberisiert werden. Ohne Zusatzeinstellungen muss man jedoch erneut bname und passwort eingeben. In den Einstellungen kann man das automatisieren indem man unter *network.negotiate-auth.trusted-uris* alle URLs einträgt bei denen Kerberos Authentifizierung erfolgen soll.
Um Delegation und damit Ticket Forwarding an Browser zu ermöglich muss man bei Firefox unter *network.negotiate-auth.delegation-uris* die URLs eingeben.
[PROM] S28 Beispiel einer Kerberos Umgebung

14 Quellen

[PROM] Kerberos von Mark Pröhl 2011, Dpunkt.verlag
[POGW] Basiswissen IT-Sicherheit von Werner Poguntke 2010, W3L-Verlag - Herdecke – Witten, 2. Auflage
[TANA] Verteilte Systeme von Andrew S. Tanenbaum Maarten van Steen 2008, Pearson Plc. Verlag, 2. Auflage
[BERE] Identity Management von Elisa Bertino & Kenji Takahashi Artech House Verlag
[ZIEP] Netzwerkangriffe von innen von Paul Sebastian Ziegler O'Relly Verlag