

**Nils Hoier**

# **Authentifikation mit Kerberos**



**Betreuer: Dr. Hans-Joachim Mück**

Universität der Freien und Hansestadt Hamburg  
Fachbereich 18 – Informatik

**Nils Hoier**

0hoier@informatik.uni-hamburg.de

# Authentifikation mit Kerberos

Seminararbeit im Rahmen des Seminars „18.415 Sicherheit in vernetzten Systemen“ im Wintersemester 2002/2003 mit Bezug auf den Vortrag „Authentifikation“, gehalten von Benjamin Schleinzer, Lutz Leitner und Nils Hoier.

18.415 Sicherheit in vernetzten Systemen (Foliensammlung):  
<http://www.informatik.uni-hamburg.de/RZ/lehre/18.415/>

Universität der Freien und Hansestadt Hamburg.  
Fachbereich 18 – Informatik.

# I. Inhaltsverzeichnis

<b>1. EINLEITUNG .....</b>	<b>4</b>
1.1. INTENTION DER AUTHENTIFIKATION .....	4
1.2. KERBEROS – BEWACHER DER UNTERWELT .....	4
1.3. DIE ENTWICKLUNGSGESCHICHTE VON KERBEROS .....	4
1.4. MOTIVATION .....	5
<b>2. KERBEROS SYSTEMARCHITEKTUR .....</b>	<b>6</b>
2.1. DESIGNASPEKTE DER VERSIONEN 4 UND 5 .....	6
2.2. REALM & PRINCIPAL .....	7
2.3. KEY DISTRIBUTION CENTER .....	8
2.3.1. <i>Authentication Service</i> .....	9
2.3.2. <i>Ticket Granting Service</i> .....	9
2.4. REPLIZIERTE KDCs .....	10
<b>3. KERBEROS PROTOKOLL VERSION 4.....</b>	<b>11</b>
3.1. LOGIN .....	12
3.1.1. <i>Authentication Server Request</i> .....	12
3.1.2. <i>Authentication Server Reply</i> .....	12
3.2. REMOTEZUGRIFF .....	13
3.2.1. <i>Ticket Granting Server Request</i> .....	13
3.2.2. <i>Ticket Granting Server Reply</i> .....	13
3.2.3. <i>Application Request</i> .....	14
3.2.4. <i>Application Reply</i> .....	14
3.3. INTERREALM-AUTHENTIFIKATION - DIREKT .....	15
<b>4. KERBEROS PROTOKOLL VERSION 5.....</b>	<b>16</b>
4.1. LOGIN .....	17
4.2. REMOTEZUGRIFF .....	17
4.3. TICKETTYPEN .....	18
4.3.2. <i>Gültigkeitszeitraum: Renewable &amp; Postdated</i> .....	18
4.3.1. <i>Delegation von Rechten: Forwardable &amp; Proxiable</i> .....	18
4.4. ASN.1 .....	19
4.5. INTERREALM-AUTHENTIFIKATION - INDIREKT .....	19
<b>5. ALLGEMEINE SYSTEMSCHWÄCHEN .....</b>	<b>21</b>

# 1. Einleitung

## 1.1. Intention der Authentifikation

Grundlegende Intention der informatisch betrachteten Authentifikation ist der eindeutige Nachweis der Identität eines Benutzers (einer Instanz) gegenüber einer anderen Kommunikationsinstanz, und der sich durch diese Verifikation implizit ergebenden Restriktionen für diesen Benutzer in einem betrachteten Nutzungskontext.

Durch diese eindeutige Identitätsprüfung soll somit gewährleistet werden, dass zum einem dem Benutzer eine identitätsbezogene Kommunikationsbeziehung zugesichert werden kann, und zum anderen nur die dem Benutzer anvertrauten Zugriffsrechte in Anspruch genommen werden können.

## 1.2. Kerberos - Bewacher der Unterwelt

Die MIT-Entwickler bewiesen über ihre fachliche Kompetenz hinaus auch ein Gespür für einfallreiche Namen. Eine kleine Anekdote zum Namen Kerberos.

Der Name Kerberos stammt aus der griechischen Mythologie und symbolisiert den Namen eines dreiköpfigen Höllenhundes, dem Bewacher der Unterwelt. Überliefert ist des Weiteren, dass dieser Hund dem Ersuchenden Eingang gewährt, jedoch jegliche Rückkehr aus seinem Reich zu verhindern weiß.

In wieweit diese Symbolik dem Authentifikationskonzept von Kerberos gerecht wird mag bezweifelt werden, jedenfalls sollen demnach die drei Köpfe des Hundes die drei beteiligte Schlüsselparteien Client, Server und KDC symbolisieren.

Das Deckblatt dieser Ausarbeitung zeigt das offizielle Logo von Kerberos, dem dreiköpfigen Höllenhund.

## 1.3. Die Entwicklungsgeschichte von Kerberos

Kerberos war ursprünglich nicht als eigenständige Entwicklung geplant. Das Massachusetts Institute of Technology (MIT) verfolgte 1983 die Implementation des Projektsystems „Athena“, eine softwaretechnische Umsetzung einer Client-Server-Landschaft grafischer Workstations im eigenen akademischen Umfeld.

Eine Komponente dieser Implementation war die Entwicklung eines adäquaten Authentifikationssystems zur zentralen Authentifikation. Die Geburtsstunde von Kerberos. Als Basis für die Umsetzung diente ein von Needham-Schroeder bereits entworfenes Authentifikationsprotokoll mit geheimen Schlüsseln, welches von S. Miller und C. Neuman zum Kerberos Erstentwurf erweitert wurde.

In den Folgejahren diente Kerberos nur MIT-internen Zwecken und wurde bis Version 3 als Entwicklungsversion verbessert und schließlich 1978/88 in Version 4 erstmals als freiverfügbare Implementation vertrieben.

Version 4 wurde bis 1990 fortlaufend weiterentwickelt und verbessert, jedoch schon im Jahre 1989 wurde mit der Spezifikation einer Nachfolgerversion begonnen, der Fünften. Im September des Jahres 1993 erging die Endspezifikation als Internet-Standard in der RFC 1510.

Kerberos ist heute in einer Vielzahl von freien und kommerziellen Distributionen verfügbar. So finden sich unter anderem Umsetzungen unter Unix, Solaris, SunOS, Linux und Windows.

Aufgrund der restriktiven Exportpolitik der Vereinigten Staaten von Amerika im Bereich der digitalen Kryptographie ist Kerberos teilweise diesbezüglichen Beschränkungen unterlegen, was die Entstehung einer verschlüsselungsgekapselten, ebenso freien Umsetzung, genannt „Bones“, als Pardon zu Kerberos Version 4 begünstigt hat. Die modulare Konzeption von Kerberos Version 5, gerade im Hinblick auf die Wahl beliebiger Verschlüsselungsalgorithmen, entschärfte diese Problematik.

## 1.4. Motivation

Sicherheit in vernetzten Systemen, eine Begrifflichkeit, die in einer Zeit rasant wachsender globaler Vernetzung zusehends immer mehr an Bedeutung gewinnt.

Die Motivation der Kerberosentwicklung ist geleitet von der Maxime, dass die innere Sicherheit eines Netzes ebenso relevant ist, wie die gängige Gefahr von Außen.

Verbindungen sind unsicher. Sie werden abgehört, umgeleitet oder einfach nur unterbrochen. Ziel muss es also sein, den Übermittlungsverkehr von inhaltlich sensitiven Daten, wie den Passwörtern, auf ein geringes Maß zu reduzieren.

Kerberos setzt genau an diesem Punkt an, der Trennung von Authentisierung und Autorisierung.

Konventionell findet diese Trennung nicht statt, sodass für jeden verwendeten Dienst ein eigenes Passwort und somit eine eigene Authentisierung vonnöten ist.

Ziel von Kerberos ist die einmalige zentrale Authentisierung eines Benutzers bei einer vertrauenswürdigen Instanz, und die dezentrale Autorisierung bei den jeweils verwendeten Diensten, ohne weitere Passwortübermittlung.

Die Reduktion der Passwortübermittlung schließt nicht als solches die weit klaffenden Sicherheitslücken bestehender Internetstandards wie etwa HTTP, FTP, Telnet, SMTP oder POP.

Des Weiteren bedarf es geeigneter Verschlüsselungs- und Integritätssicherungsverfahren, die auch Bestandteil von Kerberos sind, und im Folgenden bei der Vorstellung der Kerberos Systemarchitektur und Protokollbeschaffenheit Erwähnung finden.

# 2. Kerberos Systemarchitektur

## 2.1. Designaspekte der Versionen 4 und 5

Die Implementation des Kerberos Systems in ein Kommunikationsnetz erfolgt zumeist auf der Ebene der Anwendungsschicht, bezogen auf die Schichten des ISO-OSI Referenzmodells somit auf der End-zu-End-Ebene 7. Die End-zu-End Verbindungsbeziehung der Kommunikationsinstanzen bedingt eine diesbezügliche Authentifikationssicherheit. Eine Implementation auf ISO-OSI Ebene 6 z.B. mit RPC, oder etwa aber auch die Integration auf tieferen Protokollebenen von IP, TCP oder UDP ist mit Nutzen für die Host-zu-Host Sicherheit möglich, wenn auch seltener in der Anwendung.

Die konkrete Implementation auf Ebene der Anwendungsschicht setzt die spezielle Anpassung der verwendeten Programme an das Kerberosystem voraus. Es muss ein sogenanntes Kerberising stattfinden, wobei hierfür vom Kerberosystem benötigte Bibliotheken zur softwaretechnischen Umsetzung bereitgestellt werden. Die Umsetzung bleibt dabei transparent, sodass z.B. Änderungen des Passworts wie gewohnt erfolgen, jedoch im Hintergrund eine Weiterleitung an das Kerberosystem stattfindet.

Kerberos in Version 4 und 5 basiert konzeptionell auf dem gleichen Fundament. Version 4 ist aufgrund ihrer Einfachheit und Schnelligkeit z.Z. weiter verbreitet als ihre Nachfolgerversion, die jedoch funktionell um eine Vielzahl an Neuerungen bereichert wurde. Version 5 wurde konsequenter modularisiert und ermöglicht unter anderem die Benutzung von beliebigen Verschlüsselungsroutinen. In Version 4 ist nur die Verschlüsselung mittels veraltetem DES unter Verwendung einer kerberos-spezifischen Plaintext-Cipher-Block-Changing Methodik möglich<sup>1</sup>. Modular wählbar in Version 5 ist ebenso der zu verwendende Protokollstapel. Die Vorgängerversion ermöglichte diesbezüglich, trotz größter Relevanz, nur TCP/IP. Weitere Neuerungen in Version 5 sind die Typisierung von verschiedenen Netzwerkadressformaten und die Erweiterung und Änderung beim Nachrichtenformat und Protokoll. Nachteilig ist die begrenzte Kompatibilität der verschiedenen Versionen, wobei Version 5 abwärtskompatibel zu Version 4 ist, das Gegenteil – wie so häufig – jedoch nicht gegeben ist.

Aufgrund der konzeptionellen Gleichheit der Versionen, werden im Folgenden die gemeinsamen Systemparadigmen und -Komponenten, wie etwa Principal, Realm und KDC, dargestellt und im Anschluss zweigleisig der Protokollablauf und entschiedene Erweiterungen in Version 5 erläutert.

---

<sup>1</sup> Im Rahmen des Seminars „18.415 Sicherheit in vernetzten Systemen“ wurde auch das Thema „Kryptographie“ behandelt. Die Folien des Vortrags „Kryptographische Verfahren - Theorie“ finden Sie unter: <http://www.informatik.uni-hamburg.de/RZ/lehre/18.415/>

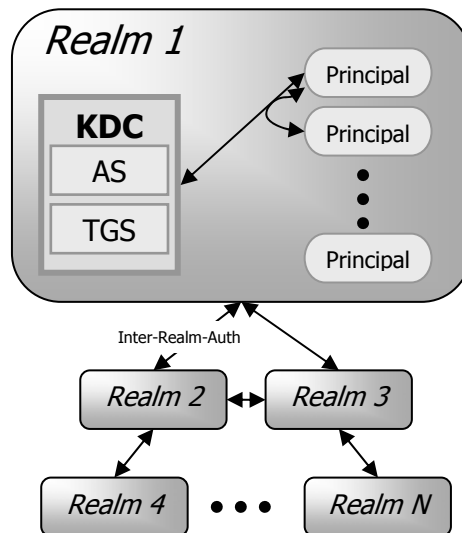


Abbildung 2: Kerberos Systemarchitektur. Realm, Principal & KDC (basierend auf [Eis99] S. 6)

## 2.2. Realm & Principal

Der Vorteil verteilter Systeme gegenüber einer zentralen Konzeption ist die gewonnene Redundanz und somit die Verhinderung von Ausfällen einer einzelnen Fehlerquelle. Die Verwendung von Verteilte Systemen ist auch in entschiedenem Maße an die Bedürfnisse und administrativen Pflichten bzw. Bedenken der benutzenden Organisationen gebunden. Organisationskritische Daten, wie die Passwörter, von externen Administratoren verwalten zu lassen, ist kaum denkbar. Um dieser sogenannten Single-Point-of-Failure Problematik Rechnung zu tragen und um eine bessere Repräsentation und Administration von Organisationen zu ermöglichen, gibt es in der Kerberosystemarchitektur eine Aufspaltung in verschiedene Domänen, den sogenannten Realms.

Jede Realm repräsentiert ein komplettes Kerberosystem, mit realmeigenem Key Distribution Center (KDC) als vertrauenswürdige Vermittlungspartei zwischen den einzelnen Principals der Realm.

Principals sind in Kerberos Benutzer, Clients, Clientprozesse und -programme, also jene Objekte, die im Netzwerk miteinander unter Verwendung des Kerberosystems kommunizieren, jedoch nicht selber Komponenten des Systems sind.

Principals haben dabei immer einen realmweit eindeutig identifizierbaren dreiteiligen Bezeichner und sind immer nur Teil einer Realm.

Der Bezeichner eines Principals setzt sich dabei in Version 4 aus einem Namen, einer Instanz und der benutzten Realm zusammen. In Version 5 wird der Instanzbezeichner durch eine Mehrteilung des Name-Strings ersetzt, wodurch die gewohnte Instanzbenennung weiterverwendet werden kann. Es wird deutlich, dass ein gleicher Name aber eine unterschiedliche Instanz zwei unterschiedliche Principals hervorbringt, die z.B. in Hinblick auf Zugriffsrechte grundverschieden sein können.

Principalsyntax Beispiele:

```
name[/instance]@REALM
donald/admin@pentagon.mil
donald@pentagon.mil
```

Generell gilt, dass Realms untereinander nur eingeschränkt vertrauen.

Eine Vertrauensbeziehung zwischen verschiedenen Realms wird, der Kerberos Philosophie folgend, dadurch etabliert, dass ein Realm zusätzlich auch als Principal bei einem anderen Realm eingetragen wird. In Version 4 unterliegt man hierbei der Beschränkung, dass explizit

mindestens eine auch Principal bei der anderen sein muss. Version 5 ermöglicht es des Weiteren, dass auch indirekt über andere Realm-Principal Beziehungen, quasi transitiv, Vertraulichkeitsprüfungen erfolgen. Dieses führt zu einer erheblichen Komplexitätsreduktion und diesbezüglich geringerem Administrationsaufwand. Mehr hierzu in den Abschnitten 3.3./4.5. Interrealm-Authentifikation: Direkt/Indirekt.

Eine Realm, in Version 4 durch ein 40-Zeichen String und in Version 5 erweitert auch mit hierarchischer Benennungsstruktur nach X.500 eindeutig identifizierbar, kann ohne weiteres Zehntausende von Principals in seiner KDC-Datenbank verwalten. In einer Realm können auch durchaus mehrere KDCs in einer Realm verwendet werden, wobei dann jedoch grundsätzlich die gleichen KDC Master Keys und die gleiche Datenbank an Principal Master Keys verwendet werden. In diesen Fällen handelt es sich um replizierte KDCs, die in Abschnitt 2.4. detaillierter erläutert werden.

## 2.3. Key Distribution Center

Herzstück der Kerberos Architektur ist das Key Distribution Center, kurz KDC.

Das KDC stellt die Trusted-Third-Party dar, also jenen vertrauensvollen Vermittler der hinzugezogen wird, um die Echtheit der Kommunikationspartner zu verifizieren.

Das Prinzip der Trusted-Third-Party besteht darin, dass jeder Principal ein Geheimnis sein Eigen nennt, das ihn gegenüber der Trusted-Third-Party eindeutig authentifiziert, da nur die Trusted-Third-Party ein Mitwisser dieses Geheimnisses ist.

Jedes KDC einer Realm kennt alle Geheimnisse seiner Principals, und andererseits trauen die Principals auch nur dem realmeigenem KDC. Zwei Principals derselben Realm authentifizieren sich somit indirekt über ihr Key Distribution Center.

Das KDC verwaltet in seiner Datenbank die Namen und Schlüssel (Master Keys) all seiner Principals. Darüber hinaus werden optionale Attribute und administrative Daten archiviert. Die Master Keys der Principals werden in den meisten Implementationen aus dem Passwort der Principals abgeleitet und zusätzlich mittels DES in Version 4 bzw. einer beliebigen Methodik in Version 5 mit dem Master Key des KDCs verschlüsselt. Trotz verwendeter Verschlüsselung ist es obsolet, dass für das KDC eine erweiterte Absicherung gegenüber jedweden Angreifern erforderlich ist, weshalb der KDC Service in aller Regel auf speziell hierfür hergerichteten physikalisch sicheren Servern läuft.

Theoretisch betrachtet könnte man sich vorstellen, dass, wenn ein Client einen bestimmten Dienst nutzen möchte, eine entsprechende Anfrage an das KDC erfolgt und daraufhin das KDC eindeutige Session Keys an die beiden beteiligten Parteien zur gegenseitigen Authentifikation vergibt, wobei diese jeweils mit dem Master Keys der Parteien verschlüsselt sind. Die nachfolgende Grafik illustriert diese Vorstellung.

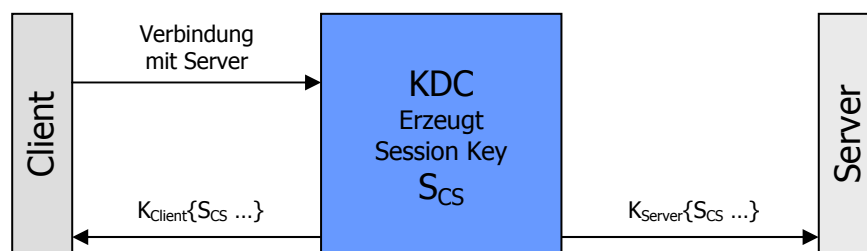


Abbildung 3: Theoretisches KDC mit Keyvergabe an beide Parteien (basierend auf [Mic99] S. 7)

In der Praxis ist eine Implementation in dieser Hinsicht kaum möglich, da für jeden neuen Client, der diesen Service nutzen möchte, eine Archivierung des Session Keys erfolgen müsste und nicht sichergestellt ist, dass der Session Key für den Server rechtzeitig vor der Anfrage des Clients versendet werden konnte. Als Folge daraus würden sich enorme Zwischenspeicherkapazitäten seitens des KDCs ergeben müssen, was eine effiziente Nutzung ausschließt.

Kerberos verfolgt einen anderen Weg. Wenn der Client das KDC informiert, eine Verbindung zu einem bestimmten Server aufzubauen, dann generiert das KDC einen gemeinsamen Session Key für beide Parteien und verschlüsselt diesem zum einen mit dem Master Key des Clients und zum anderen mit dem des Servers, was bis zu diesem Zeitpunkt der bereits oben genannten Variante entspricht. Nun sendet das KDC allerdings beide Informationen an den Client, welcher natürlich nur den Teil der Mitteilung entschlüsseln kann, der mit dem eigenen Master Key verschlüsselt ist. Die mit dem Master Key des Servers verschlüsselten Daten können vom Client nun als Ticket für eine Verbindung zum Server verwendet werden. Der Server kann nach dem Entschlüsseln des von dem Client erhaltenen Tickets eindeutig verifizieren, dass dieser mittels des KDC identifiziert wurde. Somit wird eine gegenseitige Authentifikation ermöglicht. Der Session Key und das Ticket zum Server werden als die Credentials des Clients für den Server bezeichnet.

Das Key Distribution Center ist in zwei zeitlich kausal gebundene Aufgabenbereiche unterteilt, dem Authentication Service (AS) und dem Ticket Granting Service (TGS). Beide Services verwenden trotz funktionaler Trennung die gleiche Datenbasis und werden in der gängigen Literatur häufig zur besseren Anschauung separierter abgehandelt, als das dieses im technischen Sinne sinnvoll erscheinen mag.

### **2.3.1. Authentication Service**

Der Authentication Service findet nur einmalig, während des ersten Sessionlogins, Anwendung. Er repräsentiert die Authentisierungsphase bzw. Authentifikationsphase und damit die Kapselung von der Autorisierungsphase. Der Authentication Service, auch häufig Authentication Server genannt, vergibt an den Client, nach dessen Anfrage, einen mit dem Client Master Key verschlüsselten Session Key und ein Ticket Granting Ticket (TGT), welches wiederum den Session Key und erweiternde TGT Informationen, wie etwa den Namen des Clients und die Gültigkeitsdauer des TGT, enthält. Das dienstgewährende TGT dient als Zugangsberechtigung zum Ticket Granting Service und ist deshalb mit dem Master Key des KDCs verschlüsselt. Nach dem Erhalt des Session Keys und des TGTs werden die temporär im Clientspeicher abgelegten Passwörter verworfen, da zu einem späteren Zeitpunkt mittels Übermittlung des TGTs und des Authenticators Authentifikationssicherheit zum KDC und anschließend zum Remotehost ermöglicht wird. Der Authenticator besteht aus einem mit dem Session Key verschlüsselten Zeitstempel.

### **2.3.2. Ticket Granting Service**

Wie bereits im vorherigen Abschnitt erwähnt, benötigt ein Client zur Nutzung einer Remotekommunikation nur noch das vom KDC zugesandte Ticket Granting Ticket (TGT) und einen Authenticator. Der vom Client generierte Authenticator und das TGT werden in diesem Fall mit Angabe der gewünschten Remoteinstanz an das KDC zurücksendet. Das mit dem KDC Master Key verschlüsselte TGT liefert dem KDC den Session Key des Clients und lässt die Entschlüsselung des Zeitstempels des Authenticators zu. Damit ist die Grundlage zur Generierung eines Client-Remote-Tickets gegeben, welches wiederum mit dem Session Key des Clients, und nun gerade nicht mehr mit dem Master Key des Clients, verschlüsselt zum Client zurückgesendet wird.

Detaillierte Erläuterungen zum Ablauf des Protokolls und zur Beschaffenheit einzelner Tickettypen folgen in Abschnitt 3. Kerberos Protokoll Version 4, sowie in Abschnitt 4. Kerberos Protokoll Version 5.

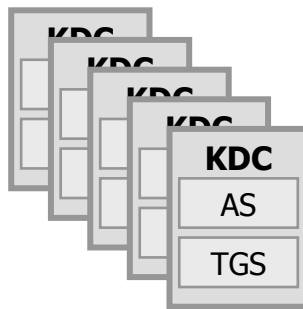


Abbildung 4: Replizierte KDCs

## 2.4. Replizierte KDCs

Wie zum Ende von Abschnitt 2.2 Realm & Principal eingeführt, können mehrere KDCs innerhalb einer Realm als verteilte Umgebung zu Vermeidung von Ausfallsituationen und zur Steigerung der Performance verwendet werden.

Ein Ausfall, sei es z.B. durch Überbeanspruchung oder einer Verbindungstrennung eines zentralen KDCs bedeutet zwangsläufig immer den kompletten Ausfall sämtlicher Einwahl- und Kommunikationsverbindungen eines kerberosgestützten Systems.

Ausweg aus solchem unhaltbaren Zustand ist die Einrichtung replizierter KDCs, die die Nutzlast teilen, und im Falle des Falles für Kompensation sorgen.

Replizierte KDCs sind inhaltlich äquivalente, gegenseitig austauschbare Kopien einer KDC Master Copy. Sie weisen demnach alle den gleichen Master Key und die gleichen Principalnamen und Principal Master Keys auf.

Das Einpflegen von Änderungen, beispielweise das Hinzufügen oder Löschen von Principals, kann nur zentral bei der KDC Master Copy erfolgen.

Die in regelmäßigen Abständen bzw. auf Benutzerkommando gezogenen Abbilder, Slaves genannt, ermöglichen nur Lese-, jedoch keine Schreibzugriffe. Diese Master-Slave Konzeption birgt jedoch auch Gefahren. Gleichzeitige bzw. widersprüchliche Änderungen seitens der Slaves können zu Konfliktsituationen führen.

Der Ausfall des Master Copy KDCs führt zwangsläufig zu einer Single-Point-of-Failure Problematik, die jedoch nicht einen Komplettausfall des Systems bedeutet, jedoch im Auffangbetrieb, mittels der replizierten KDCs, jegliche Änderungen an der bestehenden KDC Datenbank ausschließt.

Dieses Manko scheint hinnehmbar bis sinnvoll, da im Falle eines Master Copy Ausfalles die Wiederherstellung Vorrang hat, und Änderungswünsche nachstehen müssen, da gegebenenfalls ein Zurückspielen einer Slave-Kopie nötig sein könnte, und zwischenzeitige Änderungen dann verloren gingen.

Da die KDC Datenbank nicht komplett als Paket beim Ziehen der Slaveskopien verschlüsselt wird, ist prinzipiell die Gefahr gegeben, dass die Master Keys der Principals und der Master Key des KDCs von Angreifern in Folge eines Sniffings ermittelt werden können, da einzelne Principals und deren Einstellungen einzeln, jedoch mit dem Master Key des KDC verschlüsselt, separiert vorhanden sind. Unter der Annahme der Kenntnis eines Master Keys eines oder mehrerer Principals könnte mittels einer Brute Force Methodik der Master Key ermittelt und anschließend andere Principal Passwörter als Folge der Verwendung von DES in Kerberos Version 4 aus deren Principal Master Keys entschlüsselt werden.

# 3. Kerberos Protokoll Version 4

Die nachfolgenden Abschnitte 3. und 4. illustrieren detailliert den Protokollablauf der beiden verschiedenen Versionen, und vertiefen damit das bereits in Abschnitt 2.3 Key Distribution Center eingeführte Wissen.

Im Kerberos-Protokollablauf können drei wesentliche Phasen differenziert werden. In der ersten Phase erfragt der Benutzer entsprechende Credentials, um später überhaupt Anfragen für andere Services stellen zu können, was die Authentifikation des Benutzers mit dem KDC impliziert. In der zweiten Phase erfragt der Benutzer Credentials für die Authentifikation mit einem gewünschten Service. In der abschließenden dritten Phase kann sich der Benutzer mit den aus Phase Zwei gewonnenen Credentials nun beim gewünschten Service authentifizieren.

In Kerberos werden zwei Arten von Credentials unterschieden: Das Ticket eines Benutzers zu einem Service und der Authenticator eines Benutzers/Clients. Das Ticket ist jeweils mit dem Master Key des Services verschlüsselt und gibt anhand seiner Inhalte exakt vor, wer der Inhaber dieses Tickets ist – sich also beim KDC/AS rechtmäßig authentifiziert hat – und somit Zugang zu dem spezifizierten Service hat. Des Weiteren ist ein Ticket ab Erstellungszeitpunkt (Timestamp) bis zum Ablauf seiner Gültigkeitsdauer mehrfach benutzbar. In Version 4 ist ein Ticket immer sofort nutzbar, da es sich explizit um den Erstellungszeitpunkt eines Tickets handelt und nicht um den Ausstellungszeitpunkt, welcher vordatierbar wäre. Jedes Ticket – Ausnahme: Das TGT enthält Session Key  $S_{User}$  – von Kerberos Version 4 besteht aus folgenden Informationen:

$Ticket_{User \leftrightarrow Service} : \{User, Client, S_{User \leftrightarrow Service}, Lifetime, Timestamp, Service\}K_{Service}$

Für den Service ist nach der Entschlüsselung des erhaltenen Tickets in erster Stufe verifizierbar, welchem Benutzer das Ticket ausgestellt wurde und ob das erhaltene Ticket noch gültig ist. Der Session Key  $S_{User \leftrightarrow Service}$  ist nun auch dem Service bekannt und dient in zweiter Instanz zur Überprüfung des Authenticators.

Jeder Authenticator – Ausnahme: Der Authenticator im Verbund mit TGT ist mit  $S_{User}$  verschlüsselt – von Kerberos Version 4 besteht aus folgenden Informationen:

$Authenticator_{User \leftrightarrow Service} : \{User, Timestamp\}S_{User \leftrightarrow Service}$

Der Authenticator wird direkt auf dem jeweiligen Clients für jede Anfrage neu erzeugt und wird immer im Verbund mit einem Ticket versandt. Mit dem aus dem Ticket bekannten Session Key kann der Authenticator entschlüsselt werden und liefert dem Service die Gewissheit, dass der anfragende Benutzer auch wirklich im Besitz des korrekten Session Keys ist, und somit ausgeschlossen werden kann, dass das Ticket von jemand anderen versendet wurde. Mittels eines Vergleiches des mitgesendeten Timestamps mit der lokalen Zeit des Services kann überprüft werden, ob es sich um eine zeitnahe Anfrage handelt, sodass Antworten auf ältere – möglicherweise gestohlene – Anfragen unterbunden werden.

Zur Verdeutlichung der Abläufe folgt nun ein beispielhafter Kommunikationsablauf zwischen der Benutzerin Alice und der Serverinstanz Bob, wobei folgende Abkürzungen verwendet werden:

A	Alice [Name (Primary), Instanz, Realm]
B	Bob [Name (Primary), Instanz, Realm]
$C_X$	Client (IP) von X, im Sinne von Workstation
$K_X$	Master Key von X, $X = \{Alice, Bob, KDC\}$
L	Lifetime
T	Timestamp
$\{123\}K_X$	123 verschlüsselt mit Master Key von X
$\{123\}S_X$	123 verschlüsselt mit Session Key von X
$S_A$	Session Key von Alice
$S_{A \leftrightarrow B}$	Session Key zwischen Alice und Bob
TGS	Ticket Granting Server
TGT	Ticket Granting Ticket
	$TGT = Ticket_{A \rightarrow TGS} = \{A, C_X, S_A, L, T, TGS\}K_{TGS}$
$Ticket_{X \rightarrow Y}$	Ticket für X nach Y
	$\{X, C_X, S_{X \rightarrow Y}, L, T, Y\}K_Y$
$Auth_{A \rightarrow TGS}$	Authenticator von Alice zum TGS
	$\{A, Timestamp\}S_A$
$Auth_{A \rightarrow B}$	Authenticator von Alice zu Bob
	$\{A, Timestamp\}S_{AB}$

### 3.1. Login

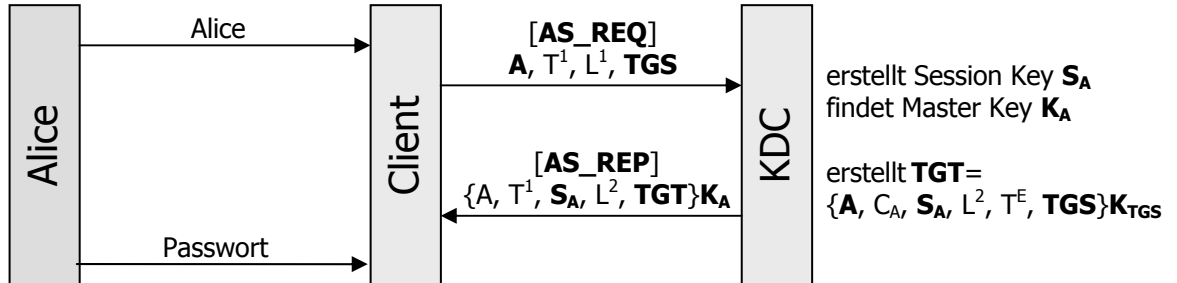


Abbildung 5: Login in Version 4, AS\_REQ, AS\_REP (basierend auf [KPS02] S. 311)

#### 3.1.1. Authentication Server Request

Alice loggt sich mit ihrem Benutzernamen an ihrem Client an. Der Client generiert nun einen in der Kerberos Dokumentation als Kerberos Authentication Request (KRB\_AS\_REQ) bezeichnete Nachricht, welche dem Authentication Server des KDCs den Namen von Alices  $A$ , einen Timestamp  $T^1$ , eine voraussichtliche Logindauer  $L^1$  und eine Aufforderung für ein Ticket zu dem Ticket Granting Server TGS im Klartext übergibt.

#### 3.1.2. Authentication Server Reply

Der KDC Authentication Server überprüft seine Datenbank nach dem Principal Alice und findet den Master Key  $K_A$ , welcher aus dem Passwort von Alice abgeleitet ist. Des Weiteren wird ein TGT generiert, welches den Principal Alice  $A$ , die Netzwerkadresse  $C_A$  des Clients von Alice, einen neu erstellten Session Key  $S_A$  für Alice, die Gültigkeitsdauer  $L^2$  und den

Erstellungszeitpunkt  $T^E$  des mitgelieferten Tickets zum TGS, verschlüsselt mit dem Master Key  $K_{TGS}$  des KDC, enthält. Somit ist das TGT nur vom KDC selber lesbar.

In der Authentication Server Reply Nachricht wird nun das TGT, mit Angabe seiner Gültigkeitsdauer  $L^2$ , im Beisein des Session Keys  $S_A$ , verschlüsselt mit dem Master Key von Alice  $K_A$ , an Alice A versandt. Die Beigabe von  $T^1$  ermöglicht dem Client die zeitlich korrekte Kontrolle zusammenhängender Requests und Replys.

Erst nach dem Erhalt der Request-Nachricht wird Alice um die Angabe ihres Passwortes gebeten, welches dann in einen DES Schlüssel konvertiert, und somit zum Master Key von Alice überführt wird. Nach der einmaligen Anwendung kann der Key  $K_A$  wieder gelöscht werden, da im weiteren Verlauf nur noch der Session Key  $S_A$  benötigt wird. Dieses verzögerte Verfahren soll die Phase der eigentlichen Passwort Verarbeitung auf ein Minimum reduzieren. Andererseits ist es möglich, nur indem man eine Benutzerkennung an einen KDC sendet, die Credentials eines Benutzers zu erfragen, welche natürlich trotzdem mit dem Benutzer Master Key verschlüsselt bleiben. Dennoch könnte man nun mit geeigneten Offline-Hacking Methoden eine Entschlüsselung herbeiführen, was die Entwickler von Version 5 bewogen hat, das Passwort mit dem Benutzernamen abzufragen. Da nur Alice die Reply-Nachricht entschlüsseln kann und nun mit den Credentials weitere Remotetickets angefordert werden können, ist die Authentifikation im ersten Schritt erfolgt.

## 3.2. Remotezugriff

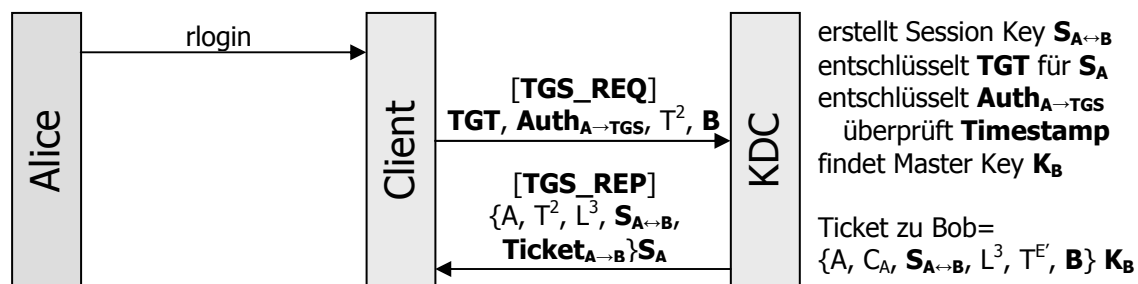


Abbildung 6: Remotezugriff in Version 4, TGS\_REQ, TGS\_REP (basierend auf [KPS02] S. 313)

### 3.2.1. Ticket Granting Server Request

Nachdem Alice nun im Besitz eines TGTs ist, kann sie mit diesem Ticket, als Zugangsberechtigung zum TGS, ein Ticket zu ihrem gewünschten Remotehost, nämlich Bob, vom KDC/TGS anfordern. Mittels des Ticket Granting Server Requests (**TGS\_REQ**) sendet der Client das **TGT**, einen Authenticator  $Auth_{A \rightarrow TGS}$ , einen Timestamp  $T^2$  und das Ziel Bob **B** unverschlüsselt zum KDC. Der Authenticator  $Auth_{A \rightarrow TGS}$  beinhaltet Alice's aktuelle Systemzeit und ist mit dem Session Key  $S_A$  verschlüsselt, was dem KDC in der Folge signalisiert, dass der Client den korrekten Session Key  $S_A$  kennt. Voraussetzung hier ist, dass die Systemzeiten der miteinander kommunizierenden Instanzen relativ synchron laufen, was in administrativ gepflegten Netzen über Timeserver ohne weiteres möglich ist.

### 3.2.2. Ticket Granting Server Reply

Das KDC entschlüsselt das TGT, um mit dem Session Key  $S_A$  den Authenticator  $Auth_{A \rightarrow TGS}$  zu entschlüsseln. Gelingt dieses, so ist die korrekte Kenntnis von  $S_A$  seitens des Clients gesichert. Weicht der gewonnene Timestamp zu weit von einem gesetzten Limit ab, so kommt es zum Abbruch des Authentifikationsablaufs.

Für das dem Request entnommene Ziel sucht das KDC den entsprechenden Master Key, in diesem Fall den Master Key von Bob  $K_B$ . Wiederum wird ein Ticket generiert, was in diesem Fall jedoch nun den Session Key  $S_{A \leftrightarrow B}$  und das Ziel Bob  $B$  enthält. Ansonsten ist es dem zuvor bekannten TGT im Inhalt funktional identisch.

Die Reply-Nachricht ist nun mit dem Session Key  $S_A$  verschlüsselt und enthält den Session Key  $S_{A \leftrightarrow B}$  und das gewünschte Ticket zu Bob  $Ticket_{A \rightarrow B}$ . Dem Client ist über  $L^3$  bekannt, wie lange das Ticket verwendet werden kann. In dem Lifetime-Datenfeld werden 5 Minuten Schritte verwendet, was eine Maximalgültigkeit von etwa 21 Stunden pro Ticket ermöglicht.

Alice hat nun alle Credentials, um sich gegenüber Bob zu authentifizieren.

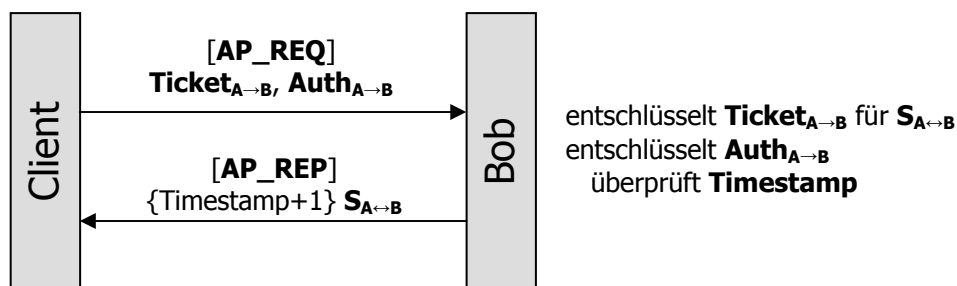


Abbildung 7: Remotezugriff in Version 4, AP\_REQ, AP\_REP (basierend auf [KPS02] S. 311)

### 3.2.3. Application Request

Mittels des Application Requests (**AP\_REQ**) kann Alice nun mit Bob direkt in Kommunikation treten. Der Client sendet dazu ihr Ticket zu Bob  $Ticket_{A \rightarrow B}$  und einen neu generierten Authenticator  $Auth_{A \rightarrow B}$  unverschlüsselt zu Bob. Das Ticket und der Authenticator sind natürlich zum einen mit dem Master Key von Bob, und zu anderen mit dem Session Key  $S_{A \leftrightarrow B}$  verschlüsselt.

### 3.2.4. Application Reply

Bob, in Kenntnis seines Master Keys  $K_B$  und somit ebenfalls am KDC authentifiziert, kann das Ticket  $Ticket_{A \rightarrow B}$  entschlüsseln – weiß damit die Authentizität des KDCs – und ermittelt aus dessen Informationen den anfragenden Benutzer  $A$ . Jetzt verifiziert Bob, ob dieser Benutzer auch der Besitzer des Tickets ist. Mit dem aus dem Ticket gewonnenen Session Key  $S_{A \leftrightarrow B}$  entschlüsselt Bob den Authenticator, welcher ja immer direkt von dem sendenden Client generiert wird, und kann somit nachweisen, dass der Client den identischen Session Key des Tickets verwendet. Mit dem Timestamp kann Bob nur zeitnahe Anfragen zulassen, etwa 5 Minuten, sodass Reply-Angriffe verhindert werden.

In der optionalen Application Reply Nachricht (**AP\_REP**) kann Bob nun gegenüber Alice seine Authentizität nachweisen, wenn Alice dieses wünscht. Dazu sendet er den Timestamp von Alice inkrementiert um 1 an Alice, verschlüsselt mit dem Session Key  $S_{A \leftrightarrow B}$ , zurück, was Alice zweifelsfrei erkennen lässt, dass Bob korrekt authentifiziert ist.

Nach Abschluss des Authentifikationsablaufes kann für die weitere Kommunikation zwischen Alice und Bob der gemeinsame Session Key  $S_{A \leftrightarrow B}$  für die Verschlüsselung verwendet werden. Darüber hinaus unterstützt Kerberos integritätssichernde Verfahren mittels Checksummen. In Abwägung von Performance und Sicherheit kann jedoch auch komplett auf derartige Mechanismen verzichtet werden.

### 3.3. Interrealm-Authentifikation - Direkt

Zwangsläufig wird es zu der Situation kommen, dass Principals zweier Realms miteinander in Kommunikation treten wollen. Da prinzipiell Principals nur ihrer eigenen Realm trauen, bedarf es erweiterter Funktionalitäten, die in Kerberos unter der Bezeichnung Interrealm-Authentifikation bereitgestellt werden. Im Folgenden wird die direkte Interrealm-Authentifikation – im obigen Beispielstil – eingeführt, wie sie in Version 4 verwendet wird. Abschnitt 4.5 widmet sich der in Version 5 neu hinzugekommenen indirekten Interrealm-Authentifikation.

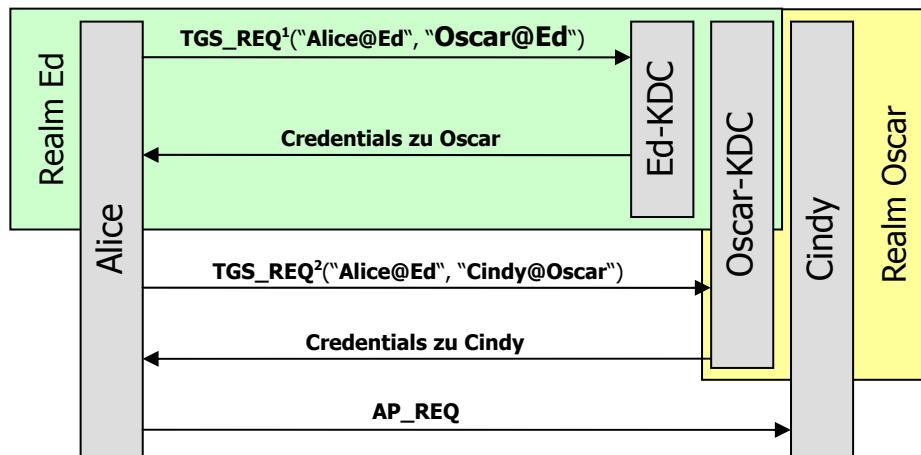


Abbildung 8: Interrealm-Authentifikation - Direkt (basierend auf [KPS02] S. 317)

Alice, die Principal der Realm Ed ist, möchte eine authentizitätsgesicherte Kommunikation mit Cindy etablieren, welche Principal der Realm Oscar ist, was Alice bekannt ist.

Nachdem sich Alice bei ihrem KDC Authentication Service authentifiziert hat und somit weitere Tickets anfordern kann, sendet sie ihrem Ticket Granting Server ein TGS Request (**TGS\_REQ<sup>1</sup>**) um ein Ticket zu dem Ticket Granting Server vom Cindy zu erhalten, also Oscar.

Ed, Alice's KDC prüft nun, ob Oscar als Principal in seiner Datenbank eingetragen ist. Ist dieses der Fall, dann erstellt Ed für Alice ein TGT für Oscar, welches mit einem speziellen Interrealm Key verschlüsselt ist, welcher beiden KDCs bekannt ist. Da Alice nun ein TGT für Oscar besitzt, sendet sie wiederum ein TGS Request (**TGS\_REQ<sup>2</sup>**), mit dem Ziel Cindy, an Oscar's TGS. Oscar entnimmt der Request-Nachricht den Hinweis, dass Alice einer anderen Realm angehört und überprüft seine Zugehörigkeit zu dessen Principalsbasis, indem er den entsprechenden Interrealm Key aufzulösen versucht, mit dem er anschließend das TGT entschlüsseln wird. Nach erfolgter Entschlüsselung erstellt Oscar für Alice einen Session Key Alice↔Cindy und ein Ticket zu Cindy. Alice kann nun mit Cindy mittels des auch im Ticket an Cindy enthaltenen gemeinsamen Session Keys eine gesicherte Kommunikationsbeziehung (**AP\_REQ**) etablieren, ganz so als seien sie in einer gemeinsamen Realm.

Die eindeutige Schwäche der direkten Interrealm-Authentifikation ist, dass mit zunehmender Vernetzung verschiedener Realms, ein Komplexitätsmaß erreicht wird, welches nur noch schwer handhabbar ist.

Unter der Annahme, dass in einem Interrealmverbund n Realms vorhanden wären, so müssten von den KDCs jeweils n-1 Interrealm Keys und KDC-Principals verwaltet werden, was bei genügend großem n beachtliche – administratorisch nicht mehr überschaubare – Maße erreicht. Jedoch ist anzumerken, dass die in Abschnitt 4.5 aufgezeigte indirekte Interrealmauthentifikation einen entschiedenen Vorteil der direkten Methodik eliminiert, nämlich die direkten, vorgegeben und somit gegen Sniffing und unseriösen Realmverwaltern recht sicheren Verbindungswege.

# 4. Kerberos Protokoll Version 5

Das Protokoll von Kerberos Version 5 unterscheidet sich nur geringfügig von seiner Vorgängerversion. Die wenigen Protokollergänzungen ergeben sich aus neu gewonnenen Funktionalitäten, wie den neuen Tickettypen Forwardable, Proxieable, Renewable und Postdated, sowie der Beseitigung von Überflüssigkeiten, etwa der Doppelverschlüsselungen bei Tickets in Kerberos Version 4.

Herausragende Neuerung gegenüber Version 4 ist die Verwendung der Abstract Syntax Notation 1, kurz ASN.1, als neue Datenrepräsentationsprache für die im Kerberos Protokoll übertragenden Nachrichten. Abschnitt 4.4. gibt hierzu kurze Einblicke.

Zu den bereits in den vorherigen Abschnitten genannten Neuerungen der Version 5 zählt die stark modularisierte Konzeption, welche unter anderem im Hinblick auf Verschlüsselung- und Integritätsalgorithmen, sowie dem verwendeten Protokollstapel administrative Freiheiten gewährt. Version 5 verwendet als Standardverschlüsselungsverfahren den DES Cipher Block Chainging Mode (CBC). Erwähnung fand auch, dass in Version 5 der Instanzbezeichner des Principals durch eine Mehrteilung des Name-Strings emuliert werden kann, und die nun erweiterte Benennungsstruktur nach X.500 die begrenzten 40 Zeichen Strings der Version 4 ablösen. Hierdurch wird erst eine effektive, wie auch effiziente, Nutzung der indirekten Interrealm-Authentifikation ermöglicht.

Einige weitere Neuerungen in Version 5:

- Die Passwort-zu-Master Key Hashfunktion berücksichtigt nun auch die jeweilige Realm, damit bei gleichen Benutzern in mehreren Realms mit gleichem Passwort nicht einmaliges Knacken gleich Mehrfachzugriff bedeutet.
- Die sogenannte Preauthentication soll verhindern, dass beliebige Personen ein AS\_REQ unter Angabe eines falschen Benutzernamens versenden können und somit ein mit dem Master Key des Benutzers verschlüsseltes Paket erhalten. Bei der Preauthentication wird in der AS\_REQ-Nachricht zusätzlich ein mit dem Master Key des Benutzers verschlüsselter lokaler Timestamp an den Authentication Server gesendet. Dieses erfordert natürlich die vorherige Eingabe des Passwortes. Dass diese zusätzlich gesendeten Informationen wiederum gesniffert werden können, steht dabei außer Frage.

Der Vollständigkeit halber sei hier auch der illustrierte – am Beispiel geleitete – Protokollablauf der Version 5 dargestellt, wobei folgende Abkürzungen ergänzend Verwendung finden:

T <sub>A</sub>	Timestamp, Zeitpunkt der Authentifikation, A = Auth-Time
T <sub>S</sub>	Timestamp, Zeitpunkt, ab wann das Ticket gültig ist, S = Start-Time
T <sub>E</sub>	Timestamp, Zeitpunkt der Ungültigkeit, E = End-Time
T <sub>R</sub>	Timestamp, Zeitpunkt, bis zu dem das Ticket erneuert werden kann, R = Renew-Till

Geringfügige Unterschiede finden sich beim Authentication Server Reply (**AS\_REP**) bzw. Ticket Granting Server Reply (**TGS\_REP**) und der hierbei involvierten Tickets. Es sollte jedoch Erwähnung finden, dass der hier abgebildete Nachrichtenverkehr im Zuge der ASN.1 Nutzung nur die per Standard eingerichteten Formate repräsentiert und auch einige spezielle Datenfelder nicht abgebildet werden.

Die Felder  $T_A$  und  $T_E$  weisen die gleiche Funktionalität wie die Felder Erstellungszeitstempel  $T^E$  und Lifetime  $L$  in Version 4 auf. Die optionalen Felder  $T_S$  und  $T_R$  ermöglichen des Weiteren die Verwendung von Postdated- und Rewable-Tickets, die in Abschnitt 4.3 genauer erläutert werden.

Das bereits verschlüsselte TGT bzw. Remote-Ticket wird im jeweiligen Reply nicht noch einmal verschlüsselt.

### 4.1. Login

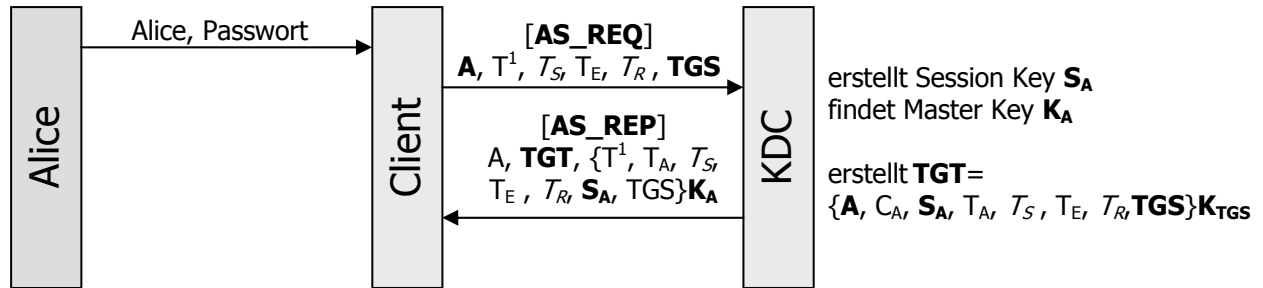


Abbildung 9: Login in Version 5, AS\_REQ, AS\_REPLY (basierend auf [KPS02] S. 311)

### 4.2. Remotezugriff

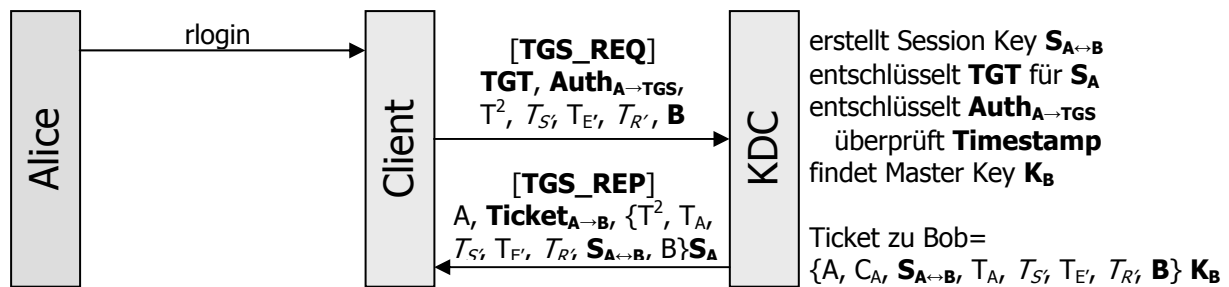


Abbildung 10: Remotezugriff in Version 5, TGS\_REQ, TGS\_REPLY (basierend auf [KPS02] S. 313)

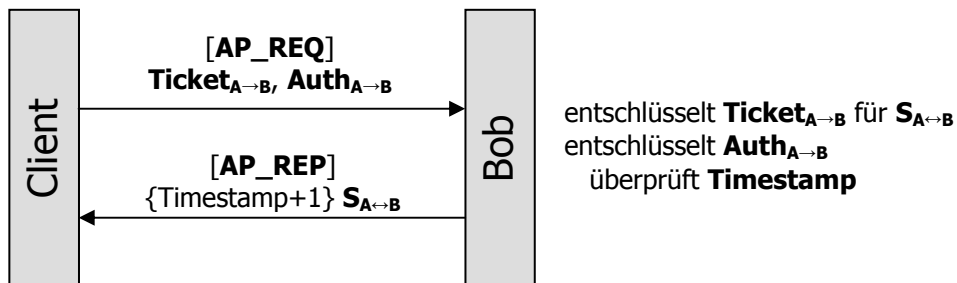


Abbildung 11: Remotezugriff in Version 5, AP\_REQ, AP\_REPLY (basierend auf [KPS02] S. 313)

## 4.3. Tickettypen

### 4.3.2. Gültigkeitszeitraum: Renewable & Postdated

In Kerberos Version 4 betrug die maximale Gültigkeitsdauer eines Tickets etwa 21 Stunden. Dieses ergab sich aus der auf 1 Oktett begrenzten Feldgröße der in 5 Minuten Schritten angelegten Lifetime-Variablen.

Diese Beschränkung wird in Version 5 durch das Auffrischen der Gültigkeit eines Tickets aufgehoben. Der einfachere Weg, einfach einen wesentlich größeren Speicherbereich für das Lifetimefeld eines Tickets zu verwenden, birgt die Gefahr des Missbrauches.

Abhängig von der jeweiligen Konfiguration eines Kerberosystems sind Tickets generell bis zu einem bestimmten Endzeitpunkt gültig, bzw. bis zu einem bestimmten Zeitpunkt als gültig gesetzt ( $T_E$ ). Sollte nun der Bedarf nach einer längeren Gültigkeit bestehen, so muss beim Request ein von dem KDC mit dem *Renewable*-Flag gesetztes Ticket angefordert werden, das bis zu einem zu bestimmenden Zeitpunkt ( $T_R$ ) erneuerbar sein soll. Bevor nun der Endzeitpunkt überschritten wird, muss das Ticket erneuert werden, indem das Ticket zum KDC zurückgesendet wird. Ist der Endzeitpunkt bereits überschritten, so wird eine Erneuerung von dem KDC abgelehnt. Das KDC wird im Erfolgsfall den Endzeitpunkt neu setzen, jedoch niemals über den maximalen Erneuerungszeitpunkt ( $T_R$ ) hinaus.

Die Timestamps  $T_S$  (Start-Time) und  $T_E$  (End-Time) ermöglichen die Ausstellung von zukünftig gültigen Tickets. Nachdem ein Postdated Ticket vom KDC angefordert worden ist, enthält es solange ein *Invalid* Flag, bis es in seinem Gültigkeitszeitraum zum KDC zwecks der Eliminierung dieses Flags zurückgesendet wird. Ist dagegen die Nutzung des Tickets nicht mehr vorgesehen, so kann dem KDC dieses signalisiert werden, sodass das *Invalid* Flag eine Validation des Tickets verhindert.

Mittels des weiteren Flags *Postdated* kann der Anwendungen signalisiert werden, dass es sich um ein ursprüngliches Postdated Ticket handelt. *May-Postdated* in TGTs liefert dem KDC die Information, dass auch Tickets in Postdated-Format ausgestellt werden dürfen.

### 4.3.1. Delegation von Rechten: Forwardable & Proxiable

Mittels der TGT-Flags *Forwardable* und *Proxiable* kann ein Ticketinhaber seine Tickets von einem anderen Client aus nutzen, wie auch weitere anfordern. Beide Flags ermöglichen die Verwendung von mehreren Host-Netzwerkadressen, sodass durchaus auch mehrere verschiedene Hosts verwendet werden können. Ein typischer Fall für ein *Forwardable* TGT ist etwa eine Situation, bei der über ein Remotelogin die komplette Funktionalität einer Umgebung genutzt werden soll – ganz so als sein man direkt vor Ort.

Wenn nun also Alice von Bob aus ein Ticket für Cindy anfordern möchte, so übermittelt Alice Bob ein *Forwardable* TGT, das von Bob aus durch ein neues TGT, nun mit Bob's Netzadresse, vom KDC – auf Alices Wunsch hin – ersetzt werden kann. Alice kann dabei angeben, ob das neue TGT wiederum ein gesetztes *Forwardable*-Flag haben soll. Ist dieses der Fall, so kann wiederum Bob mittels dieses TGTs für einen weiteren Principal ein TGT anfordern.

Das TGT-Flag *Proxiable* ermöglicht die Anforderung von Tickets, die eine andere Netzadresse als das TGT aufweisen sollen. *Proxiable* ermöglicht dagegen nicht mehr die Ausstellung von neuen TGTs. Somit kann Alice *Proxiable*-Tickets an Bob delegieren, jedoch die Verwendung von TGTs bei Bob ausschließen.

Es bleibt somit anzumerken, dass *Forwardable*-Tickets die komplette Identität (TGT) eines Benutzers auf einen anderen Client transferieren, wohingegen bei *Proxiable*-Tickets nur ausgewählte Tickets übergeben werden.

## 4.4. ASN.1

Die Abstract Syntax Notation One – kurz ASN.1 – ist eine als ISO-Norm X.680 eingetragene Standardnotation für die Definition von abstrakten Datentypen und ihren diesbezüglichen Werten.

ASN.1 wurde entwickelt, um Programmierer von der leidigen Little-Endian/Big-Endian Problematik zu befreien und ihnen somit ein ordnungsunabhängiges Format zur Nachrichtenspezifikation in die Hand zu geben.

Mittels ASN.1 können komplexe Datenobjekte konstruiert werden, die aus einfacheren Typen zusammengesetzt werden. ASN.1 selber ist, wie erwähnt, eine maschinenunabhängige Sprache. Zur weiteren Umwandlung der Sprachkonstrukte in Bitsequenzen müssen entsprechende Codierungsregeln verwendet werden. Zu diesen Regelsätzen zählt zum einen das Basic Encoding Rules (BER) Verfahren, sowie das Distinguished Encoding Rules (DER) Verfahren, wobei BER mehrere Möglichkeiten des Codierens beinhaltet und DER eine dieser Möglichkeiten – als Untermenge zu BER – repräsentiert.

Kerberos Version 5 verwendet ASN.1/DER zum Codieren und Decodieren der unterschiedlichen Protokollnachrichten.

Dabei ist die Flexibilität von ASN.1 nützlich, um Protokolländerungen oder -erweiterungen relativ einfach vorzunehmen. Optionale Felder und variable Feldgrößen lassen sich so schnell einbinden.

Nachteilig ist jedoch die erhöhte Auslastung des Systems durch das ständige Codieren und Decodieren des Nachrichtenverkehrs. Ebenso verursacht ASN.1 einen im Vergleich zum Dateninhalt relativ hohen Overhead; dieses sowohl bei der Datenhaltung in der Datenbank, als auch beim Nachrichtentransfer.

Beispiel: Spezifikation der Hostadresse in Kerberos V5 mittels ASN.1.

```
Hostaddress ::= SEQUENZ {  
    addr-type[0]    INTEGER,  
    address[1]     OCTET STRING }
```

## 4.5. Interrealm-Authentifikation - Indirekt

Neben der direkten Interrealm-Authentifikation, wie sie in Abschnitt 3.3 eingeführt wurde, kann in Kerberos Version 5 auch eine indirekte Interrealm-Authentifikations-Methodik eingesetzt werden, die die begrenzte Interrealm-Skalierbarkeit aus Version 4 behebt, wobei diese per Standard jedoch deaktiviert ist.

In Version 5 ist es somit möglich, dass Authentifikationssicherheit transitiv, über mehrere aneinander anschließende Zwischenrealms, erzielt wird. Die Eintragung des Remote-KDCs als Principal in die Realm des fragenden Principals ist nicht mehr erforderlich, solange eine Serie von Realms an einem Ende den Fragenden und am anderen Ende den Angefragten eindeutig identifizieren kann und zwei benachbarte Realms dieser Serie jeweils einen Interrealm Key teilen.

Einfacher ausgedrückt heißt dieses, wenn Alice ihre Identität gegenüber Cindy nachweisen möchte, so genügt es, dass Alice dieses gegenüber Bob leisten kann, wenn Bob wiederum seine Identität gegenüber Cindy nachweisen kann. Alice erhält somit ihr Ticket zu Cindy, indem sie zuerst ein TGT zu Bob's KDC erhält und mit diesem dann um ein TGT zum KDC von Cindy bittet. Alice erhält somit immer solange ein TGT zum nächsten in der Serie befindlichen Realm-KDC, bis sie schließlich bis zu Cindy's Stamm-KDC durchgebrochen ist, wo sie ihr Ticket zu Cindy erhält.

Die Reihenfolge der bei einer Interrealm-Authentifikation beteiligten Realms wird durch eine hierarchische Anordnung der Realms, nach DNS bzw. X.500 Norm, zueinander bestimmt. Die Vertrauensverhältnisse der Realms bzw. derer KDCs werden dabei durch die Hierarchie beeinflusst, sodass jede Realm mit ihrer Wurzel und ihren Kindern jeweils einen KDC Interrealm Key teilt. Des Weiteren sind jedoch auch aus der Hierarchie losgelöste Interrealm Key Verzweigungen möglich, die kürzere Authentifizierungswege erlauben bzw. als Rücksicherung für ausgefallene Pfade dienen können. Aus Sicherheitsgründen sollten diese sogenannten „Cross-Links“ vorbestimmbare Wege verkürzen, als dass sie völlig neue kreieren.

Damit dem servicegewährenden Principal eine Vertraulichkeitsüberprüfung der Authentifikationswege ermöglicht wird, findet eine Protokollierung der involvierten Realms, mit Hilfe eines *Transited* Feldes, in jedem Ticket statt. Dabei hängt der jeweils verarbeitende TGS einer Zwischenrealm die aus dem TGT vom Vorgänger-TGS stammenden Identifikationsinformationen an die bestehende Realmliste im *Transited* Feld an. Die Eintragung des Vorgängers soll das Vortäuschen einer falschen Identität verhindern. Auch das später ausgestellte Service-Ticket enthält dieses *Transited* Feld. Der Service entscheidet in Abhängigkeit seiner hierarchischen Einordnung, der Länge des Realm-Pfades oder bestimmter beteiligter Realm-KDCs über eine Zulassung oder Abweisung der Authentifikation.

Die indirekten Interrealm-Authentifikation birgt die Gefahr, dass mit zunehmender Pfadlänge die Möglichkeit von böartigen Angriffen, aber auch die Nichtverfügbarkeit von hierarchisch benötigten Realms, steigt. Angreifern bietet sich nun das Sniffing an verschiedener – vielleicht mit weniger Bedacht gesicherter – Stelle an. Darüber hinaus könnten ganze KDCs unseriösen Zwecken dienen.

Da KDCs in der Regel nur Interrealm Keys mit den jeweils nach oben und unten direkt anschließenden KDCs teilen, könnte der Ausfall einer dieser KDCs im schlechtesten Fall die Authentifikation mit dem ganzen angeschlossenen Baum blockieren.

# 5. Allgemeine Systemschwächen

Zu den bereits im Verlauf des Textes genannten Systemschwächen zählen:

- Die beiden verschiedenen Versionen sind nicht vollständig kompatibel zueinander. Version 5 ist zwar abwärtskompatibel zur häufiger verwendeten Version 4, diese ist jedoch nicht aufwärtskompatibel.
- Bei replizierten KDCs können widersprüchliche Änderungen seitens der Slave-Kopien nicht vollends ausgeschlossen werden. Beim Ausfall der Master-Kopie ist nur noch ein Readonly-Betrieb des gesamten Kerberosnetzes möglich. Des Weiteren besteht während des Ziehens der Slave-Kopien die Gefahr, dass die gesamten Master Keys der in dem betreffenden KDC archivierten Principals geschnitten werden könnten, da die Datenbank nicht als Gesamtpaket gezogen wird. Zwar sind die Keys mit dem Master Key des KDCs verschlüsselt, jedoch könnten geeignete Offline-Entschlüsselungsverfahren zur Dechiffrierung eingesetzt werden.
- In Version 4 kann Jeder verschlüsselte Credentials eines beliebigen Benutzers anfragen, da beim AS\_REQ nur eine Benutzerkennung versendet werden muss. Die in Version 5 mögliche Preauthentication verlängert die clienteneigene Verwendung des Passwortes.
- Die maximal mögliche Ticketgültigkeit in Version 4 ist auf etwa 21 Stunden begrenzt. Die mittels *Renewable* und *Postdated* gewonnene Flexibilität ermöglicht zwar länger laufende Jobs, jedoch liefert eine als zu lang gewählte Gültigkeitsdauer wertvolle Zeit für Angreifer, ist sie wiederum zu kurz gewählt, so ist ein häufiger Neu-Login notwendig.
- Die direkte Interrealm-Authentifikation wird mit zunehmender Zahl an Interrealm Keys unadministrierbar, wiederum die indirekte Interrealm-Authentifikation mit zunehmender Pfadlänge anfälliger für Angriffe und Ausfallsituationen.
- Die Einführung einer maschinenunabhängigen Datenrepräsentation mittels ASN.1 beseitigt jedwede Little-Endian/Big-Endian Problematiken der Version 4, hat jedoch den Nachteil einer geringeren Performance, durch ständiges Codieren und Decodieren der ausgetauschten Nachrichten.

Darüber hinaus gilt es als generelle Systemschwächen von Kerberos zu nennen:

- Kerberos gewährt keinen Schutz vor Trojanern oder anderen möglich Schnüfflern, dazu zählen natürlich auch menschliche. Die Identifikation erfolgt bei Kerberos vollständig über das geheime Passwort des Benutzers. Je schlechter dieses Passwort gewählt wird, desto schlechter – trotz bester Verschlüsselung – ist der Schutz. Die Kombination des Passwortes mit weiteren Identifikationsmerkmalen, wie etwa Smart-Cards oder Biometrische Daten, ist anzuraten.
- Der geheime Schlüssel des angefragten Services ist beim Benutzer immer in Form des mit ihm verschlüsselten Tickets vorhanden. Das gleiche gilt für den Master Key des KDCs, mit dem von ihm verschlüsselten TGT. Auch hier könnte eine Offline-Entschlüsselung angewendet werden.
- Das Kerberos Protokoll eignet sich mit seinen derzeitigen Funktionalitäten nicht für zwei aktiv miteinander kommunizierende Instanzen, etwa zwei menschlichen Benutzern. Die Kommunikation erfolgt immer nur zwischen einem aktiv agierenden Client (Benutzer) und einem passiv reaktiven Server.
- Die Verwendung eines zentralisierten Passwortservers für eine Vielzahl von Diensten setzt das uneingeschränkte Vertrauen der Benutzer in die Administratoren dieses Servers voraus.

Trotz der oben genannten Schwachstellen ist Kerberos eine weitestgehend akzeptierte, recht sichere, gegenseitige, delegierbare und interoperabilitäre Authentifikationsmethode, die ihren Dienst in einer stetig wachsenden Anzahl von Rechnern tagtäglich leistet.

# A. Abbildungsverzeichnis

ABBILDUNG 1: KERBEROS LOGO .....	1
ABBILDUNG 2: KERBEROS SYSTEMARCHITEKTUR. REALM, PRINCIPAL & KDC .....	7
ABBILDUNG 3: THEORETISCHES KDC MIT KEYVERGABE AN BEIDE PARTEIEN.....	8
ABBILDUNG 4: REPLIZIERTE KDCs .....	10
ABBILDUNG 5: LOGIN IN VERSION 4, AS_REQ, AS_REP.....	12
ABBILDUNG 6: REMOTEZUGRIFF IN VERSION 4, TGS_REQ, TGS_REP .....	13
ABBILDUNG 7: REMOTEZUGRIFF IN VERSION 4, AP_REQ, AP_REP .....	14
ABBILDUNG 8: INTERREALM-AUTHENTIFIKATION - DIREKT .....	15
ABBILDUNG 9: LOGIN IN VERSION 5, AS_REQ, AS_REP.....	17
ABBILDUNG 10: REMOTEZUGRIFF IN VERSION 5, TGS_REQ, TGS_REP .....	17
ABBILDUNG 11: REMOTEZUGRIFF IN VERSION 5, AP_REQ, AP_REP .....	17

# B. Literaturhinweise

[KPS02] C. Kaufmann, R. Perlman, M. Speciner; *Network Security – Private Communication in a public World*; 2<sup>nd</sup> Edition; Prentice Hall. 2002

[SNS88] J. G. Steiner, C. Neuman, J. I. Schiller; *Kerberos: An Authentication Service for Open Network Systems*; Project Athena; Massachusetts Institute of Technology Cambridge. 1988

[Ker03] Kerberos Version 5 Documentation – *Kerberos V5 Unix User's Guide, Kerberos V5 Administrator's Guide, Kerberos V5 Installation Guide*, Massachusetts Institute of Technology Cambridge. 2003, 2000, 2003

[Mic99] Microsoft; *Windows 2000 Kerberos Authentication – White Paper*; Microsoft Corporation. 1999

[KoN93] J. Kohl, C. Neuman; *RFC 1510 – The Kerberos Network Authentication Service (V5)*; Digital Equipment Corporation. 1993

[Eis99] M. Eisele, *Kerberos V4, V5*; Hauptseminar: Sicherheitsarchitekturen in verteilten Systemen; Universität Ulm. 1999

[USG00] United States Government – Navy, *Frequently Asked Questions about Kerberos*; URL <http://www.cmf.nrl.navy.mil/CCS/people/kenh/kerberos-faq.html>. 2000

[Rea98] M. Raeppe; *Sicherheitskonzepte für das Internet*; Dpunkt Verlag. 1998

[Boe02] W. Böhmer; *VPN – Virtual Private Networks*; Hanser. 2002