Fachhochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Fachbereich Informatik
Department of Computer Science

Abschlussarbeit

im Bachelor Studiengang

Konzeption einer Kerberos-basierten Single Sign-On Lösung für ein ausgewähltes Szenario im Hochschulbereich

von
Daniel Stojceski

Erstbetreuer: Prof. Dr. Martin Leischner
Zweitbetreuer: Prof. Dr.-Ing. Wolfgang Borutzky

Eingereicht am: 31.03.2006

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ACL	Access Control List
ADM	Administration Server
AFS	Andrew File System
API	Application Programming Interface
AS	Authentifizierungs-Server
ASP	Authentifizierungs-Service Provider
BOS	Basic OverSeer Server
CIFS	Common Internet File System
DAP	Directory Access Protocol
DES	Data Encryption Standard
DIT	Directory Information Tree
DNS	Domain Name System
FC	Fedora Core
FTP	File Transfer Protocol
GID	Group Identification Number
GSS-API	Generic Security Service Application Program Interface
GUI	Graphical User Interface
HP	Hewlett-Packard
HTTP	Hypertext Transfer Protocol
ID-M	Identifizierungsmerkmal
ID-FF	Identity Federation Framework
ID-SIS	Identity Services Interface Specifications
ID-WSF	Identity Web Services Framework
IE	Internet Explorer
IT	Informationstechnologie
KAS	Kerberos Authentication Server
KDC	Key Distribution Center
KfW	Kerberos for Windows
KVNO	Key Version Number
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
LSA	Local Security Authority
MD5	Message Digest Algorithm 5
MIT	Massachusetts Institute of Technology
NFS	Network File System

NIS	Network Information Service
NSS	Name Service Switch
NTP	Network Time Protocol
OASIS	Organization for the Advancement of Structured Information Standards
OSI	Open Systems Interconnection
PAM	Pluggable Authentication Modules
PDC	Primary Domain Controller
PIN	Personal Identification Number
pts	Protection Server
RPM	RPM Package Manager
SAML	Security Assertion Markup Language
SASL	Simple Authentication and Security Layer
SCP	Secure Copy
SFTP	SSH File Transfer Protocol
SMB	Server Message Block
SOAP	Simple Object Access Protocol
SP	Service Provider
SPML	Service Provisioning Markup Language
SPNEGO	Simple and Protected GSS-API Negotiation
SSL	Secure Sockets Layer
SSH	Secure Shell
SSO	Single Sign-On
SSPI	Security Support Provider Interface
TCP	Transmission Control Protocol
TGS	Ticket Granting Server
TGT	Ticket Granting Tickets
TLS	Transport Layer Security
TTP	Trusted Third Party
TUC	Technische Universität Chemnitz
UF	Universität Freiburg
UID	User Identification Number
UK	Universität zu Köln
UP	Universität Paderborn
USB	Universitäts- und Stadtbibliothek
VLDB	Volume Location Database
vos	Volume Server
WAN	Wide Area Network

1 Einleitung

1.1 Problemstellung

Historisch bedingt zeichnen sich aktuelle Informationstechnologie (IT)-Systemlandschaften in Organisationen durch Heterogenität aus. Sie entsteht sukzessive, indem sich Organisationen an neue Kommunikationsbedürfnisse und Anforderungen anpassen. Zur wachsenden Heterogenität tragen Betriebssysteme, Anwendungen, Netzzugangssysteme, Protokolle und andere Komponenten bei, die innerhalb einer Netzinfrastruktur nebeneinander existieren und miteinander kommunizieren müssen. Heterogenität bringt das Problem mit sich, dass Daten auf verschiedene Teile eines Systems in verschiedenen Formaten vorliegen. Diese Dezentralität – die mit Verbreitung des Client-Server-Modells Einzug in Computernetzen gehalten hat – und die Vielfalt der Datenformate sorgen oft dafür, dass es an Interoperabilität zwischen den kommunizierenden Komponenten fehlt.

Der Mangel an Interoperabilität hat direkte Auswirkungen auf die Authentifizierung in einer Client-Server-Umgebung, in der einem Benutzer viele dezentrale Dienste zur Verfügung gestellt werden können. Der Benutzer muss aufgrund uneinheitlicher Authentifizierungskonzepte seine Identität gegenüber jedem Dienst beweisen, den er benutzen möchte, wohingegen ein Server seine Identität gegenüber dem Benutzer meistens nicht beweisen muss.

Außerdem ist es wegen fehlender Interoperabilität der Komponenten meist nicht möglich, die verschiedenen Mechanismen zusammenzuführen. Insbesondere müssen oft aufgrund uneinheitlicher Formate der Identifizierungsmerkmale eines Benutzers (z.B. Benutzername/Passwort) Middleware-Komponenten in die Systemarchitektur integriert werden, die zwischen Anwendungen vermitteln und verschiedene Protokolle unterstützen müssen.

Diese Situation ist auch in Hochschulen vorzufinden (vgl. [Sten_05]). Oft bringen oben genannte Komponenten eigene Datenbanken und Nutzerverwaltungen mit und müssen getrennt verwaltet werden. Sie enthalten Daten über Benutzer – einschließlich ihrer Zugriffsberechtigungen – und IT-Ressourcen der Hochschule, die verteilt in der Netzinfrastruktur abgelegt sind und mehrfach vorkommen. Diese Datenredundanz kann zu inkonsistenten Daten führen und erhöht den Administrationsaufwand. Auch bei Einsatz eines zentralen Verzeichnisdienstes wie LDAP, das das Problem der Datenredundanz

lösen kann, bleibt das Problem der mehrfachen Authentifizierung gegenüber verschiedenen Anwendungen.

1.2 Zielsetzung

Zielvorgabe dieser Arbeit ist die Machbarkeitsprüfung einer Kerberos-basierten Single Sign-On-Lösung anhand eines Migrationskonzepts für den Fachbereich Informatik der Fachhochschule Bonn-Rhein-Sieg. Neben Kerberos als Hauptkomponente, soll vor allem das auf Kerberos aufbauende Andrew File System (AFS) als Möglichkeit der Datenablage in das Konzept miteinbezogen werden.

Kerberos sorgt in einer heterogenen Umgebung für ein einheitliches Identitätsformat und gegenseitige Authentifizierung zwischen Client und Server. Daher ist Kerberos ein möglicher Lösungsansatz zur oben beschriebenen Problematik, der auch ein Single Sign-On (SSO) ermöglicht. SSO gestattet einem Benutzer das einmalige, zentrale Anmelden in einem Netz, um nach erfolgreicher Anmeldung netzweit alle Dienste, für die er Zugriffsrechte besitzt, nutzen zu können. Ein erneutes Authentifizieren oder Autorisieren gegenüber einem Dienst ist für eine Sitzung nicht mehr erforderlich.

Die derzeitige Benutzerverwaltung und Authentifizierung der Benutzer des Fachbereichs erfolgt zentral über einen LDAP Verzeichnisdienst. Dabei existiert ein physikalischer LDAP-Verzeichnisdienst, der aufgrund der genutzten Betriebssysteme Linux und Windows und der in diesem Zusammenhang vorhandenen Passwortproblematik in zwei logische Verzeichnisdienste aufgeteilt werden musste. Die meisten Dienste des Fachbereichs nutzen zur Authentifizierung des Benutzers ebenfalls die Benutzerinformationen aus dem LDAP-Verzeichnisdienst, wobei sich ein Benutzer für jeden genutzten Dienst separat gegenüber jedem Dienst authentifizieren muss.

Aufbauend auf einer Ist-Analyse der Authentifizierungs-Landschaft des Fachbereichs soll ein Migrationskonzept für eine Single Sign-On-Lösung mit Kerberos V5 und AFS entwickelt und prototypisch für einige ausgewählte Dienste im Laborversuch umgesetzt werden. Im Fokus dieser Arbeit liegen insbesondere die Authentifizierungs-Infrastruktur des Fachbereichs und Dienste, die das Kerberos-Protokoll unterstützen. Das Ziel ist, dass ein einmaliges Anmelden an einem Rechner ohne weitere Passworteingaben zur Nutzung weiterer Dienste berechtigt. Weiterhin sollen weiterführende Fragen wie die Möglichkeit der Integration der Dienste der FH-Verwaltung in das Migrationskonzept geklärt werden.

1.3 Aufbau dieser Arbeit

Kapitel 2 führt den Begriff Single Sign-On ein. Nach einer Definition des Begriffs SSO, widmet es sich vor allem der Einordnungsmöglichkeiten verschiedener SSO-Architekturen. Als Beispiel eines SSO-Systems wird Kerberos vorgestellt. Das Kapitel endet mit Kurzvorstellungen von auf Kerberos aufbauenden Lösungsansätzen anderer Hochschulen.

Kapitel 3 beschreibt die Authentifizierungs-Infrastruktur des Fachbereichs Informatik, die als Grundlage für das Migrationskonzept in dieser Arbeit dient. Im Zentrum des Kapitels stehen Dienste der Linux- und Windows-Umgebungen und ihre Beziehungen zueinander.

Aufbauend auf den Ergebnissen aus Kapitel 3 wird in Kapitel 4 das konkrete SSO-Szenario konzipiert. Nach einem Vorschlag für einen Migrationsablauf werden die Architektur und die Komponenten des Neusystems sowie die Interaktionen des Systems vorgestellt. Anschließend wird die erarbeitete SSO-Architektur in eine der in Kapitel 2 vorgestellten SSO-Klassen kategorisiert.

Eine prototypische Umsetzung der in Kapitel 4 vorgestellten Architektur erfolgt am Beispiel einiger Anwendungen in Kapitel 5. Es beschreibt die Testumgebung und stellt die für die Implementierung durchgeführten Schritte sowie die Konfiguration der einzelnen Anwendungen auf dem Server und den Clients vor.

Das vorletzte Kapitel 6 umfasst eine Evaluierung des Szenarios. Es resümiert die Hauptunterschiede der Alt- und Neuarchitektur und erläutert die aus der Neuarchitektur resultierenden Vor- und Nachteile. Ein abschließendes Unterkapitel geht auf die Ausbaufähigkeit der Architektur ein.

Den Abschluss der Arbeit bildet das Kapitel 7, das die Ergebnisse dieser Arbeit zusammenfasst, ein Fazit zieht und einen Ausblick auf eine mögliche weitere Entwicklung gibt.

2 Single Sign-On und dessen Ansätze im Hochschulbereich

In diesem Kapitel werden grundlegende Aspekte des Single Sign-On (SSO) vorgestellt, die für das Verständnis der Arbeit notwendig sind. Darunter fällt eine Definition des Begriffs SSO und verschiedene Architekturansätze von SSO. Als Beispiele für SSO-Systeme werden Kerberos V5 [Neum_05] und das Andrew File System (AFS) sowie dessen Einsatz in anderen Hochschulen vorgestellt.

2.1 Authentifizierung von Benutzern

Benutzer vertrauen in einer geschlossenen oder offenen IT-infrastruktur einer Menge von Vertrauensinstanzen - auch Trust Center genannt. Geschlossene Infrastrukturen, wie z.B. die Infrastruktur einer Fachhochschule werden wegen der logischen Abgeschlossenheit auch Domänen genannt (vgl. [Hors_99]). Vertrauensinstanzen sind Komponenten einer Domäne, die bestimmte Sicherheitsdienste erbringen, wie z.B. das Authentifizieren von Benutzern. Im Sinne dieser Arbeit wird eine Vertrauensinstanz durchgehend als Service Provider (SP) bezeichnet und ein SP, der Benutzer authentifiziert als Authentifizierungs-SP (ASP), der auch eine Trusted Third Party (TTP) - eine spezielle Form eines Trust Centers, der zwei Parteien vertrauen - sein kann.

SP können einen Benutzer authentifizieren oder bieten ihm einen bestimmten Dienst oder Inhalt. Beispiele für SP sind z.B. Web Services, das File Transfer Protocol (FTP), Telnet, usw. SP befinden sich innerhalb einer Domäne, aus Kerberossicht innerhalb eines Kerberos Realms (siehe Glossar) oder aus der Sicht des AFS innerhalb einer AFS-Zelle. In dieser Arbeit soll zunächst durchgehend der Begriff Domäne gebraucht werden und an gegebener Stelle durch die Begriffe Kerberos Realm oder AFS-Zelle abgelöst werden. Die Begriffe stehen jedoch stets für das Äquivalent der logischen Abgeschlossenheit einer Infrastruktur.

Damit sich ein Benutzer bei einem SP anmelden kann, muss er in irgendeiner Form seine Identifizierungsmerkmale (ID-M, siehe Glossar) mit dem SP teilen. Die Identifizierungsmerkmale, die üblicherweise in einer Datenbank oder einem Verzeichnisdienst wie dem Lightweight Directory Access Protocol (LDAP) aufbewahrt werden, sind mit Zugriffsrechten für den Benutzer für das jeweilige System verknüpft. Die Benutzerregistrierung und Berechtigungsvergabe werden oft über eine Benutzermanagement-Komponente durchgeführt und definiert. Abb. 1 zeigt ein für heute typisches Authentifi-

zierungs-Szenario für einen Benutzer. Ein typischer Benutzer verfügt über ein bis N verschiedene Identifizierungsmerkmale für einen bis N SP, die in eine bis M Domänen liegen können. Dabei kann es genauso viele Datenbanken, Verzeichnisdienste und Benutzermanagement-Komponenten geben wie es SP gibt, also N.

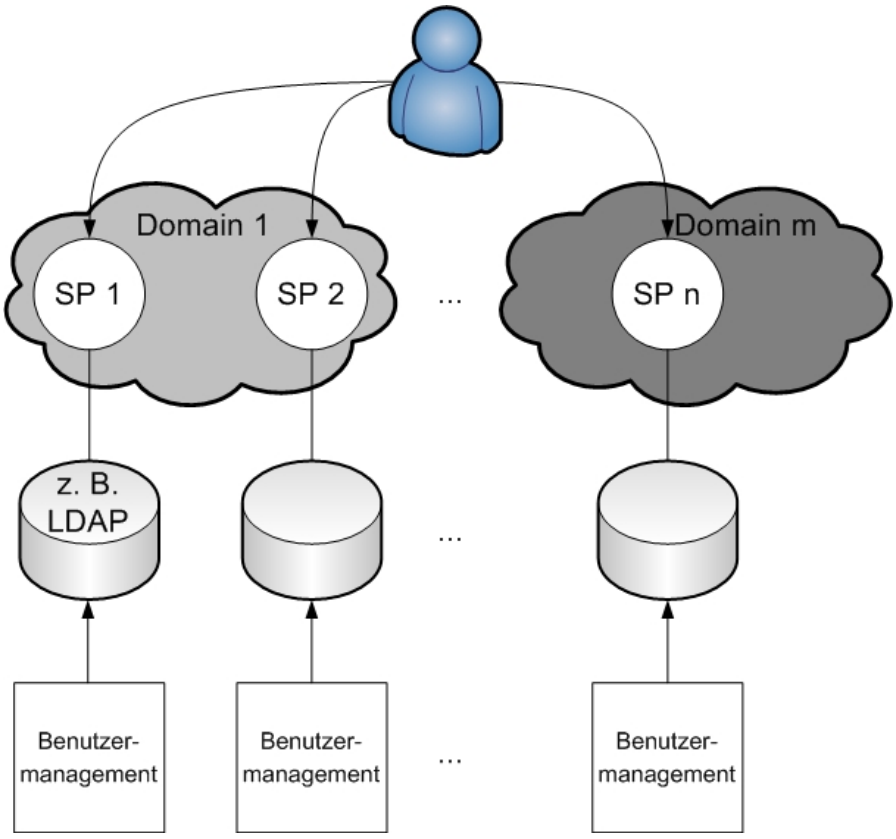

Abb. 1: Typisches Authentifizierungs-Szenario für einen Benutzer

Es gibt vier verschiedene Arten der Benutzerauthentifizierung gegenüber einem ASP (vgl. [Fisc_01]):

1. Der Benutzer weiß etwas (z.B. ein Passwort)
2. Der Benutzer besitzt etwas (z.B. ein Token auf einer SmartCard)
3. Der Benutzer besitzt eine physikalische/besondere Eigenschaft (z.B. sein Fingerabdruck oder seine Art zu unterschreiben) – auch als Biometrische Verfahren bezeichnet.
4. Eine Kombination aus den hier aufgezählten Arten – auch als Hybridverfahren bezeichnet.

Passwörter kann sich ein Benutzer merken oder aber auf ein Medium, wie z.B. einen Notizzettel oder einem Passwortmanager aufbewahren. Die Verwendung eines Notiz-

zettels ist wahrscheinlicher, weil der heutige Benutzer mit einer Fülle von Passwörtern, Benutzernamen oder aber bei numerischen Eingabesystemen mit Personal Identification Numbers (PIN) für viele verschiedene SP ausgestattet ist, die er sich nicht alle merken kann. Passwörter können ausgespäht, systematisch erraten (Brute-Force-Verfahren) oder vom Benutzer aus Versehen oder Unachtsamkeit offen gelegt werden und stellen daher die schwächste Form der Authentifizierung dar. Die Sicherheit eines Systems hängt stark von der Einhaltung vorgegebener Passwortrichtlinien sowie ihrem Management ab und ist gefährdeter je mehr Passwörter für einen Benutzer existieren.

Um die Anzahl der Passwörter auf ein Minimum zu reduzieren und die Sicherheit zu erhöhen wird verstärkt auf ein Konzept gesetzt, das als Single Sign-On (SSO) bezeichnet wird. Im nächsten Unterkapitel wird auf das Konzept hinter SSO näher eingegangen.

2.2 Single Sign-On (SSO)

2.2.1 Definition

Wie die nachfolgenden Unterkapitel zeigen werden, werden verschiedene Ansätze für das Implementieren eines SSO-Szenarios verfolgt, je nach darunter liegender Infrastruktur und benötigten Eigenschaften. Das Hauptmerkmal und die Hauptanforderung an ein SSO-Szenario bleiben immer gleich. Die folgende Definition soll dieses Merkmal in einer allgemeinen und für diese Arbeit gültigen Form festhalten.

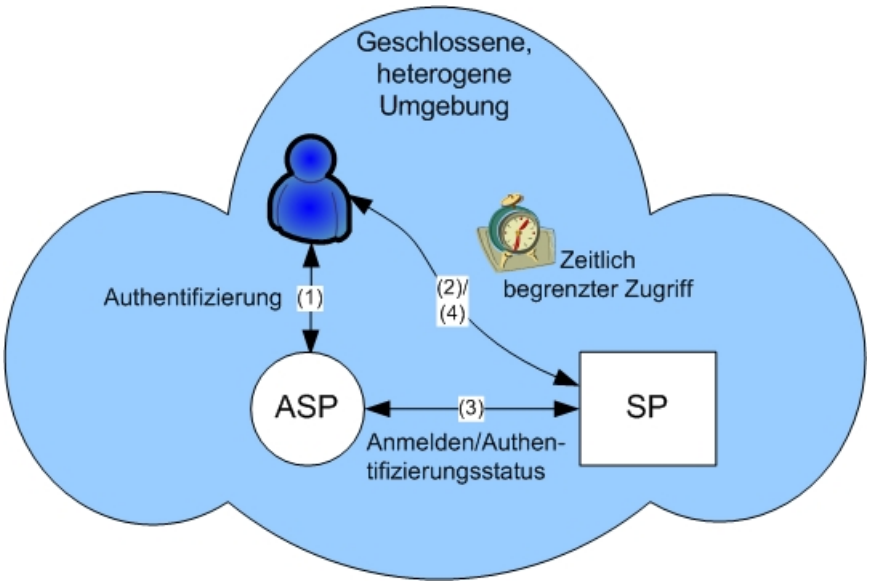

Abb. 2: Allgemeines SSO-System

Definition: Single Sign-On ist ein Konzept, das es einem Benutzer nach einer einmaligen Authentifizierung gegenüber einem ASP ermöglicht, auf geschützte Ressourcen für die der Benutzer Zugriffsrechte besitzt in einer wohl definierten (geschlossenen) heterogenen Umgebung zuzugreifen, ohne dass für einen wohl definierten Zeitraum weitere manuelle Authentifizierung vom Benutzer gegenüber einer Ressource verlangt wird.

Abb. 2 verdeutlicht die SSO-Definition bildlich. In einem allgemeinen SSO-Szenario authentifiziert sich der Benutzer gegenüber einer ASP-Komponente (1) und fordert Zugriff auf einen SP (2). Die ASP-Komponente kann einen Benutzer bei anderen SP anmelden (3) oder die SP über den Authentifizierungsstatus des Benutzers informieren (3). Schließlich stellt der SP den gewünschten Dienst bereit (4). Wie die ASP-Komponente eingeordnet werden kann, soll im nächsten Kapitel erläutert werden.

2.2.2 Einordnung von SSO-Systemen

Für SSO-Systeme existieren verschiedene Ansätze zur Einordnung und Klassifizierung. Während SSO-Systeme vor wenigen Jahren noch als desktop- und webbasierte Systeme (vgl. [Blum_02]) oder zentralisierte und föderierte (engl.: federated) SSO-Systeme (vgl. [Roes_02]) klassifiziert wurden, wird in [Pash_03] ein abstrakterer Ansatz verfolgt, der weiter unten genauer erläutert wird.

Mit desktop- und webbasierten Systemen sind Systeme gemeint, die entweder den Zugriff auf interne Applikationen und Datenbanken im internen Netz vereinfachen und fokussieren (desktopbasiert) oder den Fernzugriff auf ein geschlossenes Netz ermöglichen (webbasiert).

Als zentralisierte SSO-Architekturen werden SSO-Systeme bezeichnet, die eine zentrale Authentifizierungsinstanz und eine zentrale Benutzerdatenbank vorhalten, z.B. Authentifizierung über Kerberos. Föderierte SSO-Systeme – auch alliierte Domänen (vgl. [Hors_99]) oder Federated Identity genannt – behandeln die Identität eines Benutzers als Allgemeingut, die nicht auf ein System oder einer Systemlandschaft beschränkt ist [Buch_03]. Die jeweiligen Systeme verbinden sich zu alliierten Systemen, indem sie eine Vertrauensbeziehung miteinander eingehen und dadurch sicher miteinander kommunizieren können. Benutzerinformationen können mittels eines Übertragungsprotokolls, wie beispielsweise die vom OASIS-Konsortium [OASI_06A] entwickelte Security Assertion Markup Language (SAML) [OASI_06B], zwischen Systemen aus-

getauscht werden, so dass benutzertransparent SSO möglich wird. Bekannte Vertreter dieses Prinzips sind die von Sun initiierte Liberty Alliance [Libe_06] und .NET Passport von Microsoft [Pass_06], die sich hauptsächlich durch die Art der Datenhaltung unterscheiden.

Diese Arbeit hält sich an die Taxonomie von [Pash_03] (vgl. auch [Fumy_05], S. 271), weil die dort definierten Schemata eine Unterstützung bzgl. des Designs eines SSO-Systems bieten und allgemein auf viele Systeme anwendbar sind. Die Unterteilung von SSO-Systemen in desktop- und webbasierte Systeme wird nicht näher in Betracht gezogen, da inzwischen aufgrund der in den letzten beiden Jahren raschen Entwicklung auf dem Identitätsmanagement-Segment Identitätsmanagement-Software von einer Vielzahl an Anbietern auf dem Markt angeboten wird, die desktop- und webbasierte Aspekte vereinen – wie z.B. Hewlett Packards OpenView Select Access 6.0 [Hewl_05] oder IBMs Tivoli [IBM_05]. Dadurch kann die Klassifizierung von SSO-Systemen in desktop- und webbasierte Systeme als veraltet betrachtet werden.

Die in [Pash_03] erarbeiteten SSO-Schemata liefern zwei Hauptklassen von SSO-Architekturen, die aufgrund des Standortes der SSO-Komponenten je Hauptklasse zwei allgemeine Unterkategorien aufweisen. Der Standort der SSO-Komponenten kann entweder lokal auf der Benutzerplattform oder als externer Dienst – als so genannter SSO Proxy – vorhanden sein. Die Unterklassen erhalten in der Bezeichnung dementsprechend das Suffix Lokal oder Proxy-basiert. Die zwei Hauptklassen sind Pseudo-SSO und True SSO:

1) Pseudo-SSO (siehe Abb. 3)
   Eine Komponente, die als Pseudo-SSO Komponente bezeichnet wird, verwaltet ein bis N verschiedene Identifizierungsmerkmale (Pseudonyme) eines Benutzers, die der Benutzer sonst zum Authentifizieren gegenüber verschiedenen SP benutzt. Die Pseudo-SSO Komponente hat die Aufgabe automatisch den von einem SP verlangten Authentifizierungsmechanismus und das geeignete Identifizierungsmerkmal des Benutzers benutzertransparent bereitzustellen. In diesem Szenario können mehrere Identifizierungsmerkmale eines Benutzers einem SP zugeordnet sein, so dass von einer N:1 Beziehung gesprochen werden kann (vgl. [Pash_03]).

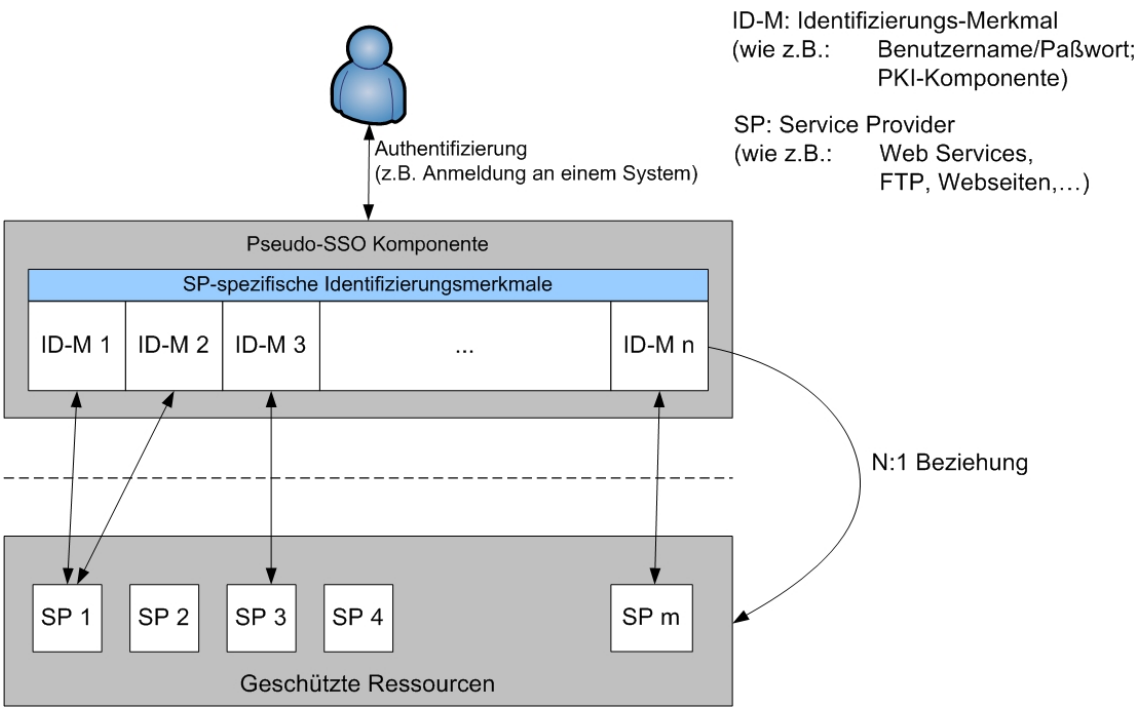

Abb. 3: Lokales Pseudo-SSO (in Anl. an [Pash_03])

2) True SSO (siehe Abb. 4)

Auszeichnend für ein True SSO-System ist eine ASP-Komponente, die mit Service Provider in Verbindung steht und Benutzer authentifiziert. SP werden vom ASP über den Authentifizierungsstatus und die SSO-Identität eines Benutzers informiert.

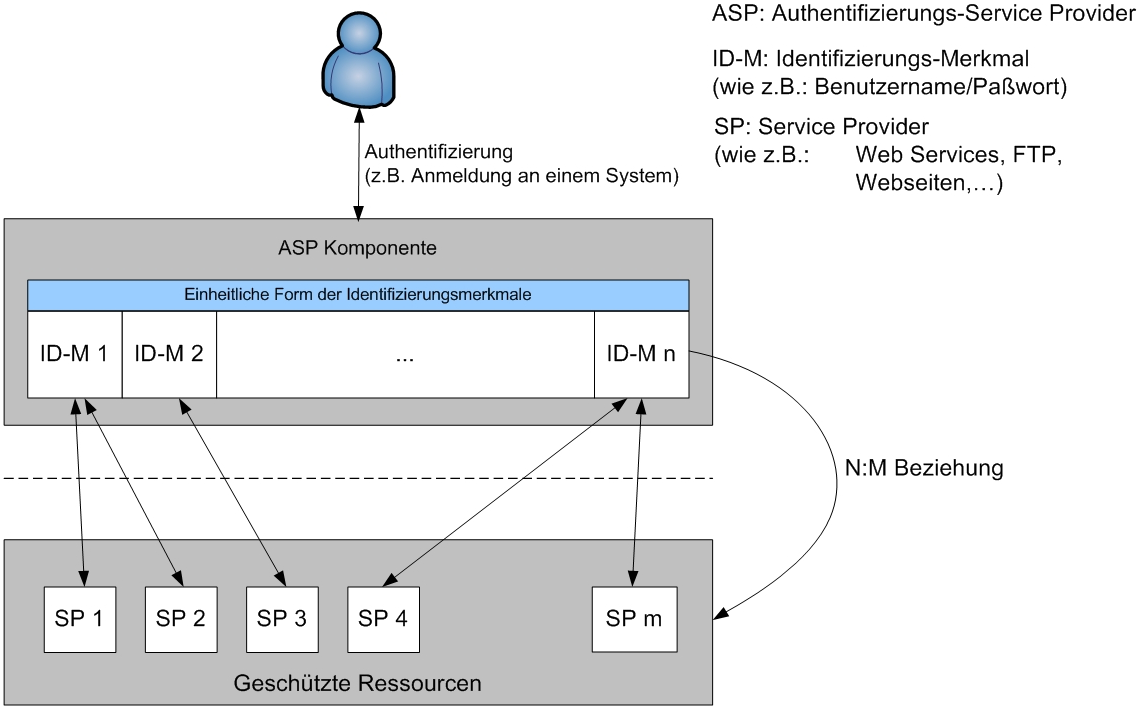

Abb. 4: Lokales True SSO (in Anl. an [Pash_03])

Während durch die Pseudo-SSO-Komponente mehrere Anmeldevorgänge stattfinden können, ist beim True SSO für ein SSO-Szenario ein Anmeldevorgang durch den Benutzer ausreichend. In der Regel ist bei dieser Architekturvariante eine einheitliche Form der ID-M gegeben. Daraus ergibt sich eine weitere Besonderheit von True SSO: Ein Benutzer kann aus einer Menge von ID-M wählen oder mit einem ID-M auf mehrere SP zugreifen. Eine N:M Beziehung zwischen ID-M und SP ist die Folge.

2.3 SSO-Umsetzung mit Kerberos und AFS

2.3.1 Kerberos

Kerberos ist ein am Massachusetts Institute of Technology (MIT) von Steve Miller und Clifford Neumann entwickeltes TTP Authentifizierungs- und Autorisierungssystem für Client-Server-Umgebungen, das auf dem Prinzip des geteilten Geheimnisses (engl.: shared secret) basiert. Das Kerberos-Protokoll ist eine verfeinerte symmetrische Variante des Needham-Schroeder-Protokolls [Need_78], modifiziert durch den Zusatz von Zeitstempeln [Mill_88] um replay-Attacken vorzubeugen wie sie in [Voyd_83], ab S. 165 beschrieben sind.

Es wurde ursprünglich für die interne Benutzung für das MIT Projekt Athena entworfen und ab 1988 durch das MIT verbreitet. Das Projekt Athena wird in [Balk_85] beschrieben. In [Kohl_93] wurde das Kerberos-Modell erstmals als RFC spezifiziert und im Juli 2005 durch [Neum_05] abgelöst.

Diese Arbeit beschäftigt sich mit dem SSO-Konzept von Kerberos und erläutert nicht tiefergehende Protokolleinzelheiten. Für vollständige und umfassende Informationen sei an dieser Stelle besonders auf [Need_78], [Mill_88], [Stei_88] und [Neum_05] verwiesen.

Die TTP ist das so genannte Key Distribution Center (KDC), das Benutzer authentifiziert und verwaltet. Die logische Abgeschlossenheit einer Kerberos Infrastruktur wird durch den Kerberos-Bereich definiert, dem so genannten Kerberos Realm (ab hier wird der Kerberos-spezifische Begriff Realm verwendet, vgl. Glossar). Mitglieder einer Realm sind Benutzer und SP, die als Kerberos Principals (ab hier wird der Kerberos-spezifische Begriff Principal verwendet, vgl. Glossar) bezeichnet werden. In jedem Realm ist mindestens ein KDC vorhanden. Der KDC teilt mit einem Principal einen geheimen Schlüssel, d.h. alle Schlüssel liegen an zwei Stellen vor, beim Principal und

dem KDC. Handelt es sich bei dem Principal um einen Benutzer, leitet das KDC den Schlüssel üblicherweise aus dem Benutzer-Passwort ab und bewahrt ihn in verschlüsselter Form in einer Datenbank auf.

Primäre Ziele von Kerberos sind [Mill_88]:

- Authentifizierung (engl.: authentication) von Benutzern und Service Provider. Ziel ist die Unterstützung für Einweg- und gegenseitige Authentifizierung von Principals. Nur ein authentifizierter Benutzer kann auf SP in einer Realm zugreifen.
- Autorisierung (engl.: authorization). SP die Kerberos als Authentifizierungsinstanz nutzen, sollen eigene Autorisierungsmodelle implementieren und sich dabei auf die Authentifizierung durch das KDC verlassen können.

Einige wichtige Aspekte der Designziele:

- Klartextpasswörter sollen auf einem Client nur so kurz wie möglich gespeichert und so schnell wie möglich zerstört, nicht über das Netz übertragen und nicht auf den Servern aufbewahrt werden. Dadurch soll das Abfangen von Passwörtern in einem Netz erschwert werden.
- Die Authentifizierung, Verschlüsselung und Benutzung von verschiedenen SP soll benutzertransparent vollzogen werden. Der Benutzer bekommt nur das für ihn übliche Anmelden am System mit.
- Die Authentifizierungsgültigkeit besitzt eine begrenzte Lebenszeit, soll aber innerhalb einer Sitzung immer wieder gebraucht werden können. Aus dieser Eigenschaft ergibt sich die SSO-Fähigkeit von Kerberos.

Kerberos gehört zu den ersten entwickelten sicheren Protokollen mit SSO-Eigenschaften und erlaubt den Principals sich gegenseitig zu authentifizieren, indem es so genannte Granting Tickets (vgl. Glossar) an einen Principal verteilt. Die gesamte Kommunikation zwischen Principals basiert auf den Austausch von Tickets, die den Zugriff auf SP in einer Realm erlauben. Dadurch entfällt für einen Benutzer die wiederholte Authentifizierung gegenüber SP durch Eingabe eines Benutzernamen und eines Passwort, wenn er auf diese zugreifen möchte.

Ein Ticket ist ein elektronischer Legitimationsnachweis (engl.: credential, vgl. Glossar),

der zur Identitätsverifizierung genutzt wird und der einen so genannten Session Key enthält. Derselbe Session Key liegt immer beim Client und Server vor. Er kann für die Verschlüsselung des Nachrichtenaustauschs zwischen Principals verwendet werden. Nach einer kurzen Vorstellung der Kerberos-Komponenten wird im Folgenden beschrieben, wie Kerberos SSO ermöglicht und in welche SSO-Klasse Kerberos fällt.

Kerberos Komponenten

Die zentrale Komponente von Kerberos ist das KDC. Es ist die zentrale Vergabestelle von Schlüsseln und verwaltet Kerberos-Anmeldungen. Das KDC besteht im Wesentlichen aus folgenden Komponenten (vgl. Abb. 5):

- Authentifizierungs-Server (AS). Hat die Aufgabe Principal Anfragen entgegenzunehmen und ihnen ein mit ihrem Schlüssel verschlüsseltes Ticket Granting Ticket (TGT) auszustellen, der ein Session Key beinhaltet. Das TGT kann somit nur von dem Principal entschlüsselt werden, der in Besitz des entsprechenden Schlüssels ist.
- Ticket Granting Server (TGS). Nimmt Principal Anfragen mit gültigem TGT entgegen und stellt SP-spezifische Tickets aus.
- Principal Datenbank. Verschlüsselte Informationen über alle Principals in einer Realm. Zu jedem Principal sind das vor allem Information wie sein eindeutiger Name und das zu ihm gehörende Geheimnis.
- Administration Server (ADM). Stellt ein Netzinterface mit Lese- und Schreibzugriff für die Principal-Datenbank bereit.

Alle Komponenten, bis auf den Administration Server, haben nur lesenden Zugriff auf die Datenbank. Möchte ein Principal auf SP in einer Realm zugreifen, verwendet er das TGT für die Anfrage weiterer SP-spezifischer Tickets beim TGS. Mit diesen Tickets kann sich der Principal gegenüber einem SP ausweisen und auf sie zugreifen. Aus dem Session Key konstruiert der Client einen so genannten Authenticator, der einen Zeitstempel beinhaltet (vgl. [Stei_88, S. 5]) und replay-Attacken vorbeugen soll. Er wird für die Kommunikation zwischen Principals verwendet. Jede Kerberos-Infrastruktur beinhaltet neben den oben beschriebenen Komponenten noch die folgenden:

- Kerberisierte Clients und Server. Der Begriff „kerberisiert“ hat sich in der Kerberos-Fachwelt durchgesetzt und deutet auf Client und Server hin, die um Kerberos-Funktionalitäten erweitert sind.

Kerberos Kommunikationsablauf

Abb. 5 zeigt den wesentlichen Kommunikationsablauf zwischen einem kerberisiertem Client und Server und dem KDC:

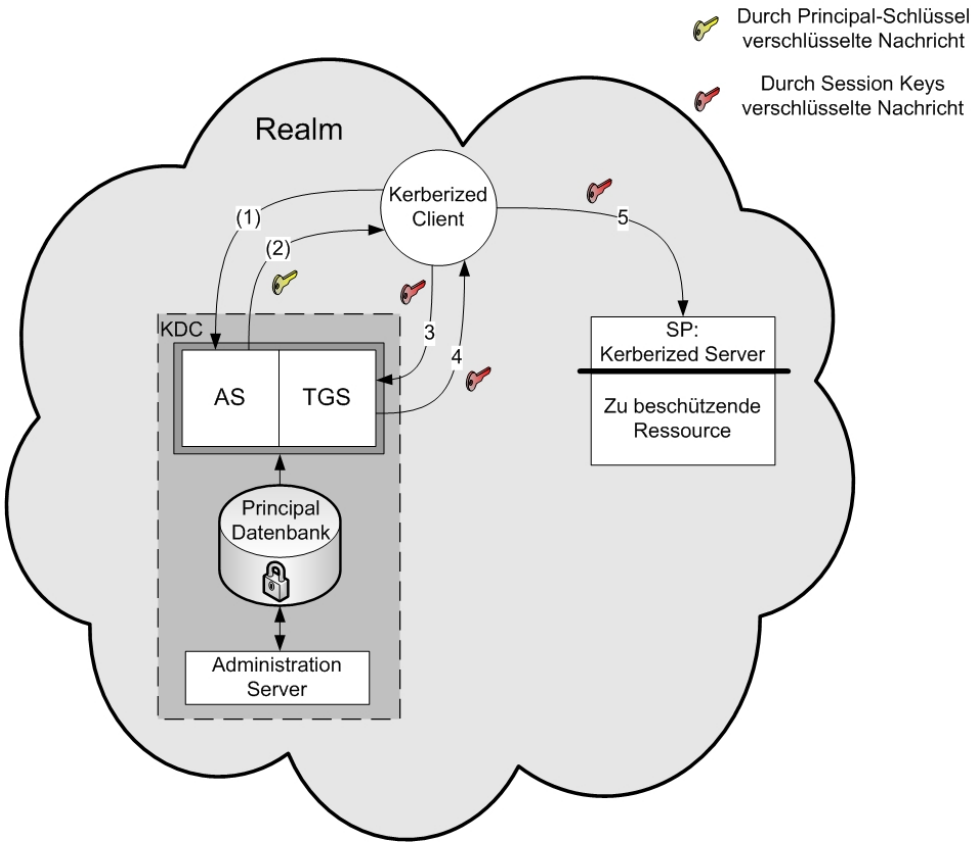

Abb. 5: Kerberos-Kommunikationsablauf und -Komponenten

(1) AS Request. Der Client sendet eine unverschlüsselte TGT-Anfrage zum KDC, die unter anderem den Principal Namen des Clients beinhaltet.

(2) AS Reply. Der KDC sendet dem Client eine Antwort, die mit dem Schlüssel des Clients verschlüsselt ist und unter anderem das TGT und einen TGS-Session Key beinhaltet.

(3) TGS Request. Der Client möchte auf einen Dienst zugreifen. Dafür sendet er eine mit dem TGS-Session Key verschlüsselte Nachricht zum KDC. Die Nachricht beinhaltet unter anderem das TGT, den Authenticator und den gewünschten Service Principal Namen.

(4) TGS Reply. Der KDC überprüft das TGT und den Authenticator und sendet dem Client eine mit dem Session Key aus Schritt (2) verschlüsselte Antwort. Sie beinhaltet vor allem ein Service Ticket und ein neuen Session Key, der die Kommunikation zwischen Client und Service Principal verschlüsseln soll.

(5) Service Request. Der Client kann schließlich das Service Ticket mit Hilfe des

Session Keys aus Schritt (4) verschlüsseln und Anfragen an einen SP richten.

Cross-Realms

Mehrere Realms können miteinander in Beziehung gesetzt werden, so dass ein Benutzer ohne erneutes Anmelden kerberisierte SP anderer Realms nutzen kann (vgl. Kapitel 4.3.2, Abb. 21). Diese Art Beziehung wird als Cross-Realm Trust bezeichnet. Der Cross-Realm Trust wird dadurch erreicht, dass der Principal-Schlüssel des TGS mit einer anderen Realm geteilt wird.

SSO-Einordnung von Kerberos

Kerberos lässt sich in der SSO-Hauptklasse True SSO der Unterkategorie Proxy-based zuordnen. SSO wird dadurch erreicht, dass der Benutzer sich so lange nicht an einem System anmelden muss wie sein TGT gültig ist. Mit einem ID-M, dem TGT, kann auf M verschiedene SP zugegriffen werden, so dass hier eine 1:M Beziehung besteht (vgl. [Pash_03] und siehe Kapitel 4.6, Abb. 29). Kerberos wird in dieser Hinsicht auch als Ticket-basierte Architektur angesehen (vgl. [Cler_02], S. 46).

2.3.2 Andrew File System (AFS)

Das Andrew File System (AFS) ist ein seit 1983 durch die Carnegie Mellon Universität und der IBM Corporation entwickeltes verteiltes Dateisystem, das auf dem Client-Server Prinzip basiert und auf Kerberos als Authentifizierungsinstanz für den Zugriff auf Daten aufbaut. Da die Beschreibung der Konzepte, die hinter AFS stecken den Rahmen dieser Arbeit sprengen würden, kann hier nicht detailliert auf AFS eingegangen werden. Dafür sei hier vor allem auf [Morr_86] und die umfangreiche Dokumentation auf [OAFS_06] verwiesen.

Primäreinsatzgebiet von AFS ist die effiziente Bereitstellung von Dateien in einem Local Area Network (LAN) oder Wide Area Network (WAN). Im Jahr 2000 hat IBM den Code als Open Source freigegeben. Freie AFS Client- und Server-Implementierungen sind nun in Form von OpenAFS [OAFS_06] für verschiedene Betriebssysteme (darunter: UNIX, Linux, MacOS, Windows, …) in der zurzeit stabilen Version 1.4.0 auf [OAFS_06] verfügbar. Das AFS zeichnet sich durch folgende Eigenschaften aus:

- Hierarchischer Namensraum. Die Wurzel des Dateibaums ist `/afs`, die den globalen Einstiegspunkt in den Dateibaum bereitstellt (siehe Abb. 6) und sich bei einem Client als Subbaum des lokalen Dateisystems präsentiert. Dabei

sind die Daten auf den AFS Servern in ein AFS-spezifisches Format abgelegt und nur durch einen AFS Client oder einen Client mit AFS-Unterstützung erreichbar. Jeder AFS Client hat standortunabhängig immer dieselbe Sicht auf den AFS-Namensraum.

- Verschlüsselter Client-Server Datenaustausch. Der gesamte Datenaustausch zwischen Client und AFS-Server ist verschlüsselt.
- Erweiterung um Zugriffskontrolllisten (engl.: Access Control List (ACL)), die eine fein abgestufte Zugriffskontrolle auf Verzeichnisebene erlauben (siehe Tabelle 1).
- Caching von Daten auf dem Client. Aus dem AFS abgerufene Daten werden lokal in einem Cache des Clients aufbewahrt und vom so genannten Cache Manager verwaltet. Eine Datenaktualisierung auf den Servern findet größtenteils nach Beendigung einer Sitzung statt. Dadurch wird die Netzlast verringert und lokal die Performance gesteigert. Die Datenkonsistenz auf den Servern wird durch einen speziellen Mechanismus – dem so genannten callback Mechanismus – gewährleistet.

Im Gegensatz zu Netzwerkdateisystemen wie das Network File System (NFS) unter UNIX oder Common Internet File System/Server Message Block (CIFS/SMB) unter Microsoft Windows, baut AFS auf Kerberos auf, so dass es die Eigenschaften von Kerberos erbt. Der wesentliche Unterschied besteht in der Gültigkeit und Handhabung der Tickets. AFS-Tickets können nur für AFS-interne Server verwendet werden. Um auf durch das AFS geschützte Daten zuzugreifen, muss ein Benutzer im Besitz eines AFS-Tickets sein. Der Benutzer erhält das Ticket beim einmaligen Authentifizieren gegenüber dem AFS und kann es für eine bestimmte Zeit transparent für verschiedene AFS-Server verwenden, um verschlüsselt auf Daten im AFS-Namensraum zuzugreifen. AFS besteht aus einer Vielzahl von AFS-Servern, die in Kapitel 4.3.3 vorgestellt werden. Das Gegenstück zu Kerberos Cross-Realms wird bei AFS als Cross-cell bezeichnet, wobei einem authentifizierten Benutzer der Zugriff auf Daten einer fremden Zelle (siehe nächster Abschnitt) gewährleistet wird.

Die größte administrative Einheit von AFS ist die so genannte Zelle (siehe Abb. 6). Eine Zelle ist eine Sammlung von Servern, die ein einziges zusammenhängendes Dateisystem präsentiert. Typischerweise besteht eine Zelle aus einer Menge von Hosts, die denselben AFS Domänennamen teilen. Der Name der Zelle wird in der Regel durch den Domänennamen der Organisation, in der AFS zum Einsatz kommt, bestimmt.

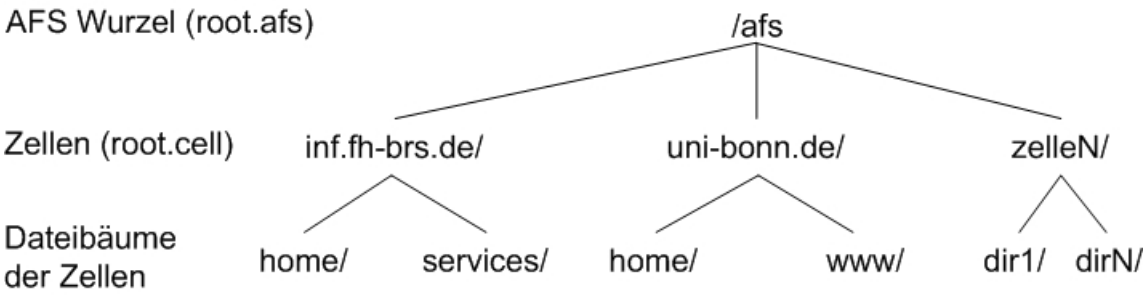

Abb. 6: Beispiel für einen AFS Dateibaum

Die kleinste Einheit ist das so genannte AFS Volume. Volumes bilden die Grundlage für die Kontingentierung des AFS-Speichers. Ein Volume ist eine Datenstruktur – eine Art Container –, die aus einer Sammlung zusammenhängender Dateien und Verzeichnisse gebildet wird (z.B. das Homeverzeichnis eines Benutzers) und durch eine festgelegte Quota begrenzt wird. Volumes werden in den AFS Dateibaum durch einen so genannten Mountpoint montiert und können im laufenden Betrieb, auch während der Benutzer gerade auf dieses Volume zugreift, durch AFS oder den Administrator benutzertransparent auf einen anderen AFS-Server verschoben werden. Dadurch kann eine Lastverteilung auf den AFS-Servern erreicht werden.

Zugriffsrecht	**Bedeutung**
administer (a)	Zugriffsrechte auf ein Verzeichnis ändern
lookup (l)	Dateien im Verzeichnis auflisten
insert (i)	Dateien in einem Verzeichnis hinzufügen
delete (d)	Dateien in einem Verzeichnis Löschen
read (r)	Inhalte von Dateien in einem Verzeichnis lesen
write (w)	Dateien in einem Verzeichnis schreiben
lock (l)	Dateien in einem Verzeichnis sperren

Tabelle 1: Elemente der AFS Zugriffskontrollliste

Für das Authentifizieren gegenüber dem AFS ist in der Regel der mit OpenAFS gelieferte AFS Kerberos Authentication Server (KAS) zuständig, der allerdings auf der veralteten Kerberos Version 4 basiert. Zur Benutzerauthentifizierung besteht die Möglichkeit den AFS KAS durch einen anderen KDC zu ersetzen. Diese Strategie wird in dieser Arbeit verfolgt (siehe Kapitel 4.3.3). Der Vorteil ist zum einen, dass Schwachstellen des Kerberos-Protokolls Version 4 gemieden werden, indem Kerberos V5 eingesetzt wird. Zum anderen wird die Pflege einer zusätzlichen Principal-Datenbank verhindert. Die Unterschiede zwischen einem AFS-KAS und einem Kerberos V5-KDC sind in [Schu_98] detailliert beschrieben.

2.4 SSO Lösungsansätze mit Kerberos und AFS im Hochschulbereich

Viele Hochschulen haben dezentrale IT-Infrastrukturen zu verwalten, die das Resultat der in Kapitel 1.1 beschriebenen Problematik sind. Aus Angst vor künftigen Anforderungen an Hochschulen (z.B. Internationalisierung) und der Besorgnis, dass die IT-Infrastrukturen nicht zukunftsfähig sind, versuchen aktuelle Aktivitäten der Hochschulen zentrale Dienste in die IT-Struktur zu etablieren.

Einen ersten Lösungsansatz sehen die Hochschulen in der Integration eines Verzeichnisdienstes, um Anwendungen Benutzerinformationen zentral zur Verfügung zu stellen. Dies bestätigt eine Umfrage – an der auch die Fachhochschule Bonn-Rhein-Sieg beteiligt war – des ZKI Arbeitskreises [ZKI_06], deren Ergebnisse teilweise [Stenz_05] vorstellt. Beim ZKI Arbeitskreis steht der hochschulübergreifende Erfahrungsaustausch informationstechnischer Themenbereiche zwischen Hochschulen im Vordergrund, darunter befindet sich auch der Themenbereich Single Sign-On.

Auf der Grundlage eines zentralen Verzeichnisdienstes verfolgen Hochschulen verschiedene Lösungsstrategien, um dem Ziel eines SSO näher zu kommen. In diesem Kapitel sollen Single Sign-On-Ansätze einiger Hochschulen kurz vorgestellt werden, die dem Lösungsansatz in dieser Arbeit ähnlich sind. Dieser zeichnet sich durch Kerberos-Authentifizierung und dem AFS zur Datenhaltung aus.

2.4.1 Technische Universität Chemnitz

Die Technische Universität Chemnitz (TUC) kann als Vorreiter in der Implementierung von AFS-Strukturen im Hochschulbereich betrachtet werden. Die TUC betreibt seit 1993 die AFS-Zelle `tu-chemnitz.de` und weitere interne Zellen [Chem_06], die mit Hilfe von OpenAFS-Clients erreichbar ist. Das nötige AFS-Ticket wird beim Anmelden an einem Rechner beschafft. AFS wird unter anderem für die Ablage folgender Daten eingesetzt:

- Homeverzeichnisse. Diese können mit Hilfe des OpenAFS-Clients oder unter Windows auch über das CIFS/SMB-Protokoll erreicht werden.
- Projekt-Verzeichnisse. Separate AFS-Volumes für die Ablage größerer Datenmengen oder Projektarbeiten zu speziellen Themen.
- WWW-Verzeichnisse. Die Dokumente des WWW-Servers `www.tu-chemnitz.de` liegen ebenfalls im AFS.

2.4.2 Universität Paderborn

Aufgrund dezentraler Strukturen und daraus resultierender Datenredundanzen (wie z.B. dutzender Benutzerverwaltungen) entschied sich die Universität Paderborn (UP) für eine Zentralisierung ihrer IT-Basisdienstleistungen. Dafür wurde die in Abb. 7 gezeigte IT-Struktur konzipiert, mit dem Ziel auch SSO zu verwirklichen.

Dem Jahresbericht 2004 [Pade_04] des Zentrums für Informations- und Medientechnologien der UP ist zu entnehmen, dass für die zentrale Benutzerverwaltung ein OpenLDAP Verzeichnisdienst als Backend eingesetzt wird. Seit September 2004 werden Kerberos zur Authentifizierung sowie AFS als Dateisystem – das NFS (UNIX) und CIFS/SMB (Windows) abgelöst hat – auf allen Windows- und UNIX-Rechnern des Zentrum für Informations- und Medientechnologien genutzt.

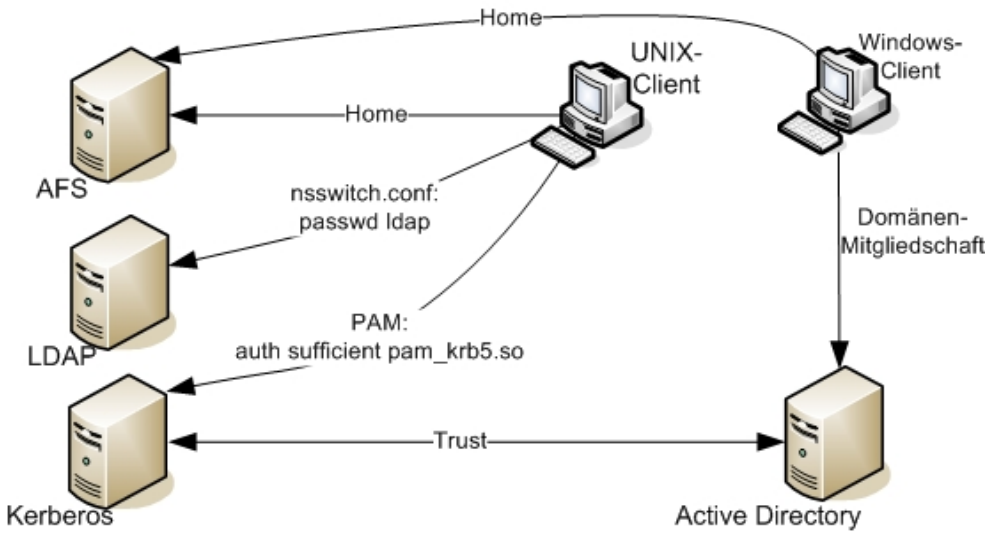

Abb. 7: Single Sign-On an der Universität Paderborn (in Anl. an [Oden_05])

Die Realm `UNI-PADERBORN.DE` erlaubt SSO und wird für die Authentifizierung von Benutzer der UNIX- und Windows-Rechner verwendet. Zur Absicherung der Verfügbarkeit der Realm kommen vier KDCs zum Einsatz. Die Verfügbarkeit der AFS-Zelle `uni-paderborn.de` wird von drei OpenAFS Fileserver gewährleistet. Weitere Charakteristika der AFS-Zelle sind:

- Ablage der Homeverzeichnisse. Im AFS liegen alle Homeverzeichnisse, persönliche Webseiten und Gruppenbereiche sowie deren Webseiten.
- Der Zugriff auf Homeverzeichnisse ist durch einen OpenAFS-Client möglich oder mit dem Secure Shell (SSH)-Protokoll durch die Anwendungen Secure Copy

(SCP) und SSH File Transfer Protocol (SFTP).

- Prominente Kunden sind z. B. die Bibliothek und die naturwissenschaftliche Fakultät.

Die Erfahrungen der UP haben gezeigt, dass der Aufbau einer solchen Architektu mühsam ist, jedoch hohe Flexibilität und Skalierbarkeit mit sich bringt und sich in heterogenen Umgebungen bewährt hat.

2.4.3 Universität Freiburg

In [Welt_04] wird der zentrale Webserver der Universität Freiburg detailliert vorgestellt. Eine besondere Bedeutung bei dem Projekt spielten die Aspekte hohe Performance, Verfügbarkeit und Ausfallsicherheit. Deshalb hat sich die Universität Freiburg (UF) bzgl. des Webservers für AFS als Ablagesystem für WWW-Daten entschieden, das universitätsweit seit 1992 erfolgreich im Einsatz ist. Es kommen virtuelle Webserver zum Einsatz, wobei für jeden dieser Server ein eigenes Volume im AFS-Verzeichnisbaum mit den Seiten und Programmen der Website angelegt wurde. Abb. 8 veranschaulicht schematisch die Gesamtkonfiguration.

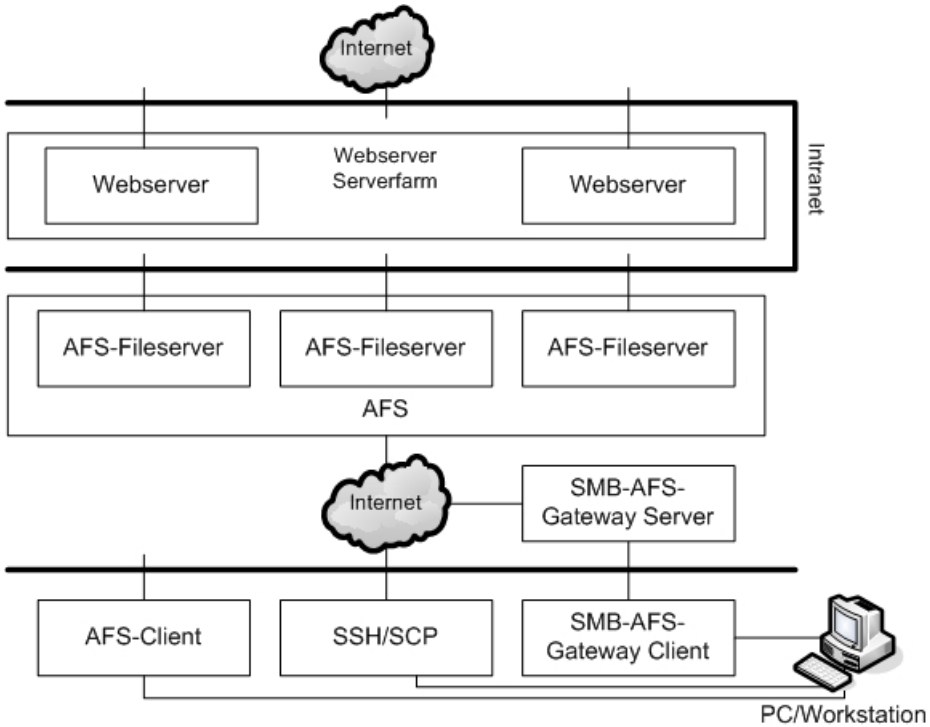

Abb. 8: Der zentrale Webserver der Universität Freiburg (in Anl. an [Welt_04])

Der Zugang zum AFS-Namensraum wird den Benutzern der UF durch folgende An-

wendungen ermöglicht:

- OpenAFS Client.
- SSH und SCP.
- CIFS/SMB-AFS-Gateway. Eigenentwicklung der UF, die Zugang zum AFS und eine grafische AFS-Rechte- und Gruppenverwaltung ermöglicht.

Die Erfahrungen der UF haben gezeigt, dass der zentrale Webserver im Zusammenhang mit dem AFS breite Akzeptanz gefunden hat. Die UF schlussfolgert, dass die schnelle Implementierung dieser Architektur auf vorhandenes AFS-Know-how zurückzuführen ist.

2.4.4 Universität zu Köln

Die Universität zu Köln (UK) unterhält mehrere AFS-Zellen. So benutzt das Regionale Rechenzentrum RRZK AFS zur Bereitstellung von Software, WWW-Dokumenten und Homeverzeichnissen. Dabei kommt für den Zugriff auf diese Dateien der OpenAFS-Client zum Einsatz.

Die Universitäts- und Stadtbibliothek (USB) zu Köln hat ein SSO-Konzept erarbeitet (vgl. [Köln_04], ab S. 110), das der Infrastruktur in Abb. 9 zugrunde liegt und den Mitarbeitern ein SSO ermöglicht. Die Mitarbeiter authentifizieren sich gegenüber einen Kerberos KDC, während die Benutzerinformationen aus einem OpenLDAP Verzeichnisdienst ausgelesen werden. Weitere Charakteristika sind:

- Authentifizierungs- und Benutzerverwaltung über Kerberos/LDAP/Samba.
- File-Server über OpenAFS und Samba.
- Citrix-Server für Windows-Anwendungen.

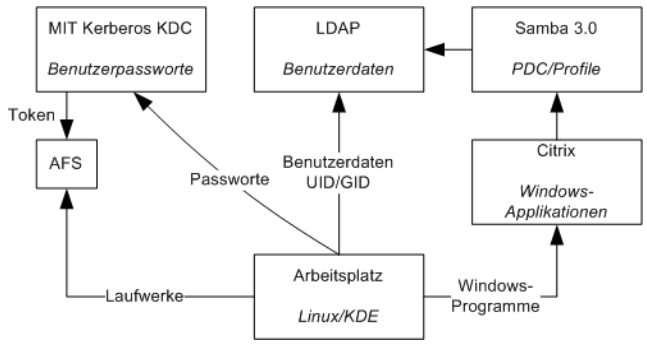

Abb. 9: Infrastruktur der USB zu Köln (in Anl. an [Köln_04], S. 120)

3 Analyse der Authentifizierungs-Infrastruktur des Fachbereichs Informatik

In diesem Kapitel erfolgt eine Analyse der Authentifizierungs-Infrastruktur des Fachbereichs Informatik. Die Analyse ist notwendig, da die aktuelle Authentifizierungs-Infrastruktur nicht ersetzt, sondern um Komponenten erweitert werden soll, die ein True SSO ermöglichen.

3.1 Vorüberlegungen

Ziel einer Altsystemanalyse ist die Beschreibung des Ist-Zustandes eines Systems. Mit ihrer Hilfe wird ein Verständnis für das Altsystem vermittelt und die Grundlage für die Weiterentwicklung oder die Migration von Systemteilen gelegt. In der Analyse werden Funktionalität, Ziele und Grobarchitektur des Altsystems beschrieben sowie die Interaktionen des Systems zu seiner Umgebung identifiziert. Dafür ist es notwendig das relevante System und die funktionalen Eigenschaften einzugrenzen sowie die zentralen Benutzer und Komponenten innerhalb der Infrastruktur zu erfassen.

Diese Arbeit ist eine Vorstudie zur Machbarkeit eines SSO Authentifizierungs-Szenarios. Aufbauend auf dieser Arbeit, müsste eine weitere Arbeit mit einem Feinkonzept, einer realen Testumgebung und anschließender Migration folgen. Eine Strukturanalyse der Fachbereichsdomäne `inf.fh-bonn-rhein-sieg.de` des Fachbereichs soll verdeutlichen, welche Komponenten der IT-Infrastruktur von besonderem Interesse für ein SSO-Szenario sind. Die vorgestellten Komponenten sollen in der neuen Authentifizierungs-Infrastruktur weiterverwendet werden können.

Abb. 10 zeigt einen Netztopologieplan des Fachbereichs Informatik. Im Fachbereich werden Linux als auch Windows Client-Rechner in den Poolräumen, Laboren und Mitarbeiterbüros angeboten. Die wesentlichen Benutzer sind:

- Studenten (derzeit ca. 2000).
- Mitarbeiter (Professoren, Sekretariat, wissenschaftliche Mitarbeiter, Administratoren).

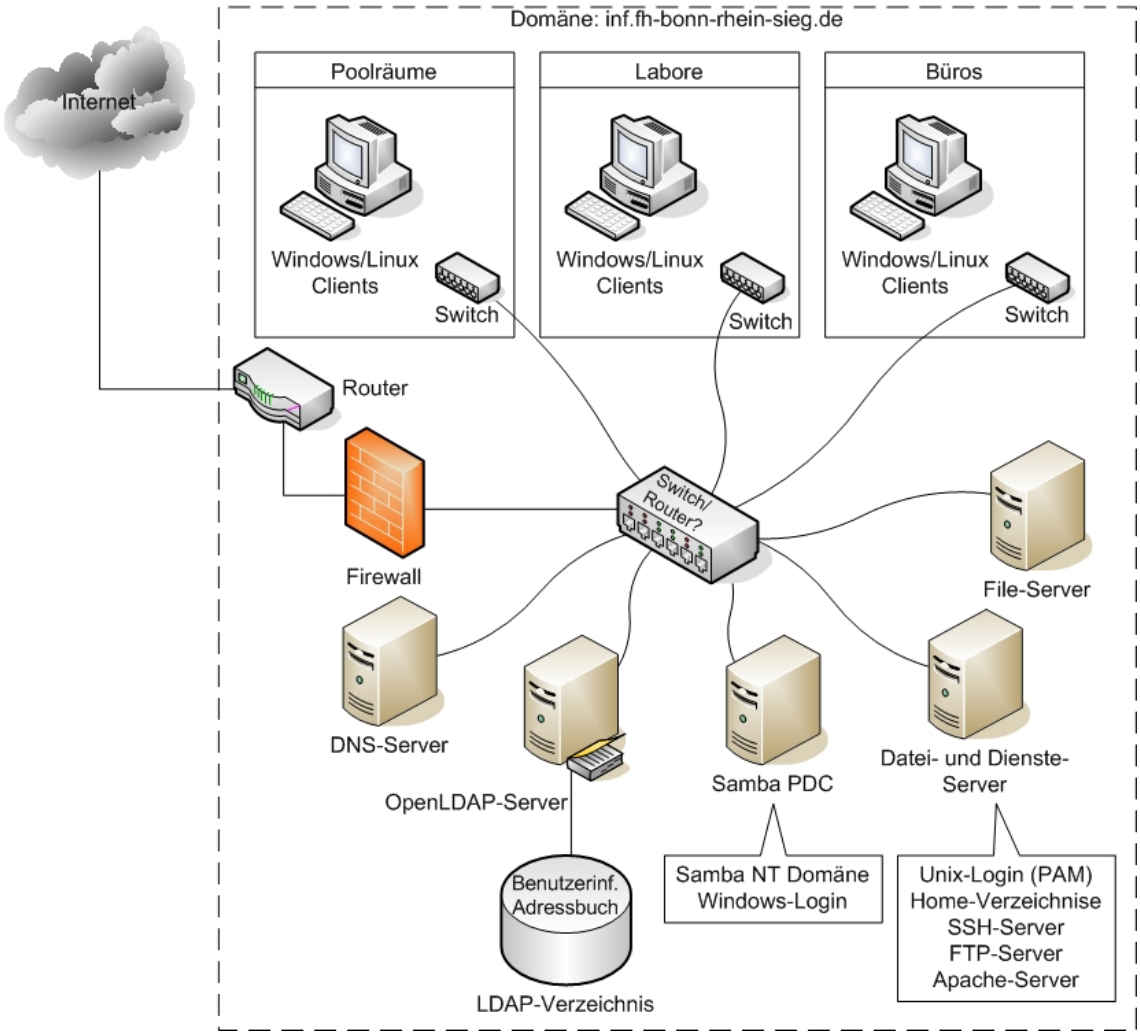

Abb. 10: Netztopologieplan des Fachbereichs

3.2 Client- und Serverlandschaft

Die Serverlandschaft basiert auf Linux-Rechner. Diese realisieren entsprechend der Sicherheitspolitik die Sicherheitsarchitektur des Fachbereichs. Die nutzbaren Dienste sind entsprechend der Sicherheitspolitik konfiguriert. Für diese Arbeit werden Komponenten der Authentifizierungs-Infrastruktur – eine Untermenge der Sicherheitsarchitektur – betrachtet, die von besonderem Interesse für ein Single Sign-On-Szenario sind.

Die wesentlichen Server sind:

- OpenLDAP-Server. Ein Open Source Verzeichnisdienst, in dem die interne Organisationsstruktur des Fachbereiches abgebildet ist und Benutzer, ihre Benutzerinformationen und ein zentrales Adressbuch abgelegt sind. Er wird vor allem zur Abfrage von Benutzer- und Passwortinformationen genutzt (siehe Kapitel

3.3).

- Samba-Server. Eine freie Software-Suite, die das Microsoft CIFS/SMB-Protokoll unterstützt. Dieses Protokoll wird auch als LanManager- oder NetBIOS-Protokoll bezeichnet. Samba ist in der Lage, Funktionen eines Windows-Servers zu übernehmen.
- Datei- und Dienst-Server. Stellen Benutzern Homeverzeichnisse sowie folgende Dienste zur Verfügung:
    - Anmelde-Dienst für Linux-Benutzer über Pluggable Authentication Modules (PAM) (siehe Kapitel 3.4).
    - OpenSSH. Server für Secure Shell (SSH), SSH File Transfer Protocol (SFTP) und Secure Copy (SCP) für verschlüsselten Ende-zu-Ende Datentransfer zwischen zwei Rechnern. Er soll vor allem die unsicheren Anwendungen Telnet und FTP ersetzen.
    - File Transfer Protocol (FTP) für die Dateiübertragung.
    - Apache Web-Server für den Zugriff auf (interne) Webinhalte des Fachbereiches.
    - File-Server. Stellt diverse geteilt genutzte Programme zur Verfügung.

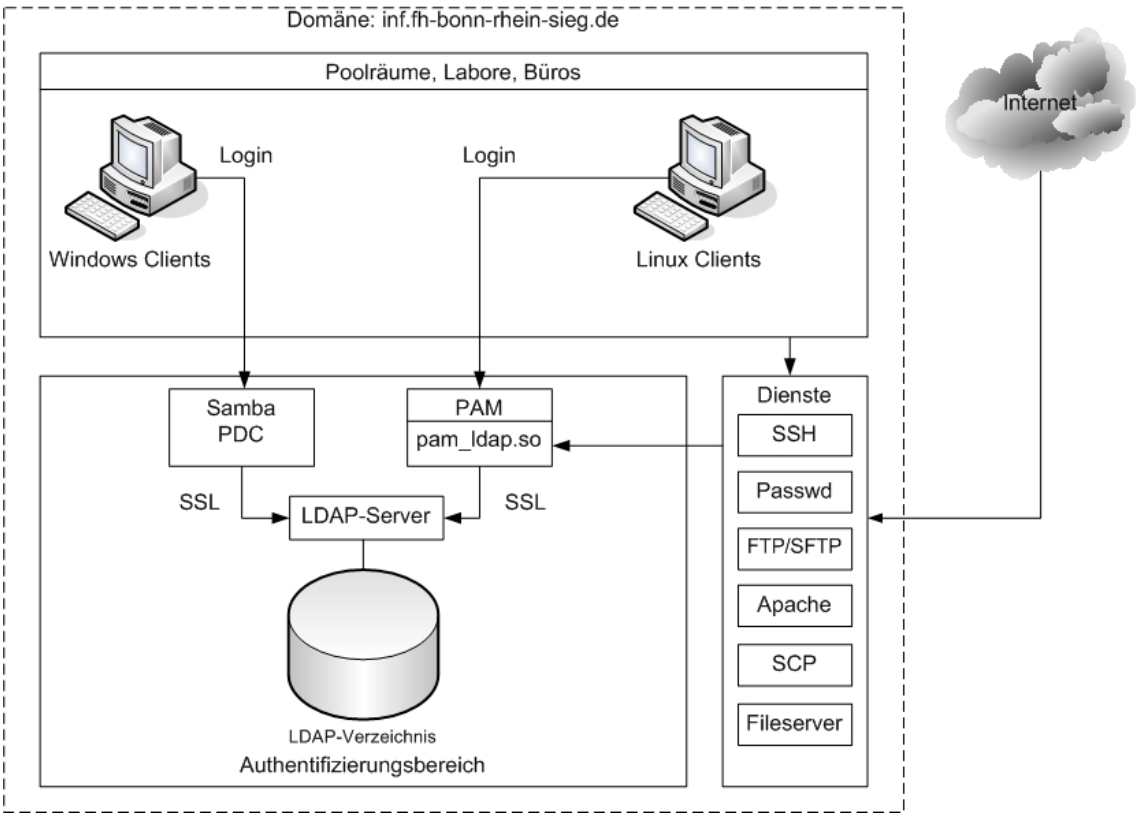

Abb. 11: Authentifizierungs-Architektur des Fachbereichs

In Abb. 11 ist das Zusammenspiel dieser Dienste mit Bezug auf die Authentifizierungs-Architektur des Fachbereichs abgebildet. Sie zeigt die wesentlichen Beziehungen der Dienste zueinander. Abb. 11 verdeutlicht gleichzeitig, welche Dienste als Clients auf die im LDAP-Verzeichnis abgelegten Informationen über die Pluggable Authentication Modules (PAM) zugreifen. Die Clientlandschaft wird von Windows XP- und Linux-Rechner bestimmt. In den nachfolgenden Unterkapiteln wird näher auf die Kommunikationsabläufe eingegangen.

3.3 LDAP

Die zentrale Komponente der Benutzerverwaltung und Authentifizierungs-Infrastruktur des Fachbereichs ist der frei verfügbare OpenLDAP-Verzeichnisdienst [LDAP_06], der auf dem Lightweight Directory Access Protocol (LDAP)-Protokoll-Version 3 aufsetzt [Wahl_97A]. In [Hodg_02] sind alle RFCs zusammengefasst, die das LDAP-Protokoll umfassen. LDAP implementiert eine vereinfachte Version des Open Systems Interconnection (OSI) X.500-Zugriffsprotokolls Directory Access Protocol (DAP).

Ein anwendungsunabhängiger Verzeichnisdienst wie OpenLDAP legt Informationen in einer baumartigen Hierarchiestruktur (siehe Abb. 12), ähnlich dem Domain Name Service (DNS), ab und erlaubt Benutzer- und Konfigurationsinformationen in heterogenen Netzen einheitlich und zentral zu verwalten.

LDAP ist ein auf dem Client-Server-Modell basierendes Zugriffsprotokoll, das auf das Transmission Control Protocol (TCP) aufsetzt (siehe Abb. 13). Es besteht aus einem oder mehreren Servern, die die Informationen bereitstellen, und Clients, die die Informationen benutzertransparent abfragen können.

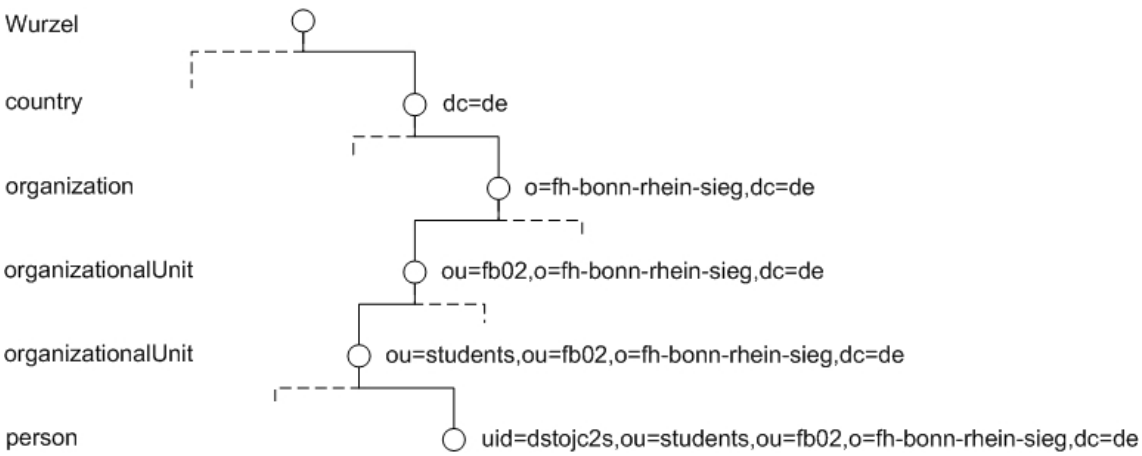

Abb. 12: Beispiel für einen LDAP-Verzeichnisbaum

Der Verzeichnisbaum (engl.: Directory Information Tree (DIT)) besteht aus Einträgen (engl.: entries). Ein Eintrag ist eine Sammlung von Attributen (engl.: attributes) und hat einen global eindeutigen Namen, den so genannten Distinguished Name (DN) [Wahl_97B]. Mit dem DN kann gezielt auf einen Eintrag zugegriffen werden. Jedes Attribut besitzt einen Typ und kann einen oder mehrere Werte annehmen. Jeder Eintrag gehört zu einer Objektklasse (engl.: object class), in der der repräsentierte Typ des Eintrags definiert wird. Information über verwendete Attributtypen und Objektklassen werden in so genannte Schemas verwaltet.

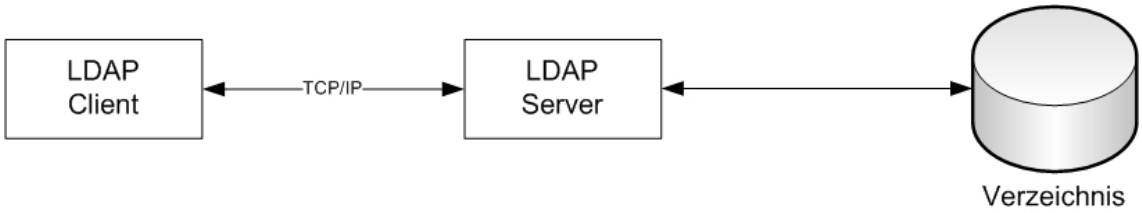

Abb. 13: LDAP Client-Server Modell

Beispiele für Schemas sind:

- shadowAccount: Ablage von Informationen zu UNIX-Benutzerkonten wie Passwörter.
- sambaSAMAccount: Ablage von Informationen zu Benutzerkonten für Windows-Umgebungen wie Passwort-Hash, Login Script Pfad, Homeverzeichnis und dazugehöriger Laufwerkbuchstabe unter Windows sowie das Roaming Profile.

Benutzerverwaltung im Fachbereich über OpenLDAP

Durch den Einsatz des OpenLDAP-Servers sind im Fachbereich folgende Kernfunktionalitäten realisiert:

- Abbildung der internen Organisationsstruktur des Fachbereiches als Baumstruktur bis zur Ebene der einzelnen Benutzer (siehe Abb. 12).
- Zentrale Benutzerverwaltung (Studenten, Professoren, Mitarbeiter).
- Anmeldung und Authentifizierung (Passwortmanagement) von
    - Windows-Benutzern
    - Linux-Benutzer
- Schnittstelle für die Dienste (z.B. SSH, FTP, Mail), die Benutzer authentifizieren müssen oder Benutzerinformationen benötigen.

Der Anmeldedienst auf den einzelnen Arbeitsplatz-Rechner benötigt Benutzerinformationen wie Authentifizierungsdaten, Homeverzeichnis, User Identication Number (UID), Group Identication Number (GID), Login-Shell und die Information, wer sich am jeweiligen Rechner anmelden darf.

Traditionell verwendet ein UNIX Anmeldedienst dafür die Einträge aus den Dateien `/etc/passwd`, `/etc/shadow` oder `/etc/group`. Z.B. beschreibt jede Zeile in `/etc/passwd` einen Benutzer durch eine Menge von Feldern, die durch Doppelpunkte voneinander getrennt werden:

```
dstojc2s:x:1631:102:Daniel Stojceski:/home/dstojc2s:/bin/bash
```

Die Bedeutung der Felder ist der Reihe nach:

- Benutzername
- Passwort („x“ bedeutet: Passwort liegt verschlüsselt in `/etc/shadow`)
- UID
- GID
- Name des Benutzers
- Homeverzeichnis
- Shell, die zum Benutzer gehört

Möchte sich ein Benutzer am System anmelden, trägt er seinen Benutzernamen und sein Passwort in Klartext in ein entsprechendes Feld. Der Dienst, der die Authentifizierung vornimmt, ermittelt daraufhin einen Hashwert aus dem Benutzerpasswort gemäß einem Verschlüsselungsalgorithmus (typischerweise Data Encryption Standard (DES) oder Message Digest Algorithm 5 (MD5)) und vergleicht das Ergebnis z.B. mit dem abgelegtem Hashwert in der `/etc/shadow`. Nur bei Übereinstimmung der Ergebnisse wird dem Benutzer der Zugang zum System gewährt.

Die Authentifizierung unter Windows-Systemen erfolgt ähnlich, jedoch besitzen die Authentifizierungsdaten eine andere Struktur. UNIX und Windows ermitteln zu einem Benutzerpasswort verschiedene Hashwerte. Dadurch muss in einer heterogenen Umgebung, in der Linux- und Windows-Rechner zum Einsatz kommen, alle Passwort-Hashes separat abgelegt werden.

Im Fachbereich ersetzt ein OpenLDAP-Verzeichnisdienst die Informationen von

`/etc/passwd`, `etc/shadow` und `/etc/group` und von Windows benötigten Benutzerinformationen. Dazu werden die oben genannten Felder mit Hilfe entsprechender Schemas in den Verzeichnisbaum für den jeweiligen Benutzer abgelegt. Dies sind im Wesentlichen folgende Informationen:

- Benutzerinformationen wie Name, Vorname, Mailadressen, Mitgliedschaft in Verteilerlisten.
- Getrennt abgelegte Linux- und Windowspasswörter.
- Pfad zu Homeverzeichnisses und Windows-Profildaten (Roaming Profile).
- UID und GID: Die UID-Kennung dient der Benutzer-Identifikation unter UNIX und verschafft ihm vordefinierte Rechte. Die GID dient der Zuweisung eines Benutzers mit einer UID zu einer Gruppe, nachdem er sich authentifiziert hat. Der Gruppe, z.B. Student, sind bestimmte erweiterte Rechte zugeordnet. Dadurch kann eine Autorisierung und eine Erweiterung der Benutzer-Rechte verwirklicht werden. Durch die UID und GID ist im Fachbereich die Rechte- und Zugriffsverwaltung realisiert.
- Quota. Ein Benutzer erhält eine vorgegebene Größe auf der Festplatte des Servers, die er nicht überschreiten kann.
- Login-Shell. Jedem Benutzer ist eine Default-Shell zugewiesen.

Einen typischen LDAP-Eintrag veranschaulicht Abb. 14.

```
dn: uid=dstojc2s,ou=students,dc=fb02,dc=fh-bonn-rhein-sieg,dc=de
objectClass: shadowAccount
objectClass: sambaAccount
givenName: Daniel
sn: Stojceski
cn: Daniel Stojceski
uidNumber: 1631
gidNumber: 102
class: b01w
entryDate: 11596
mail: daniel.stojceski@smail.inf.fh-bonn-rhein-sieg.de
loginShell: /bin/sh
homeDirectory: /home/dstojc2s
quota: default
shadowFlag: 1
uid: dstojc2s
rid: 2110
firstName: Daniel
lastName: Stojceski
displayName: Daniel Stojceski
mailAlias: bachelor01
mailAlias: daniel.stojceski
mailAlias: daniel.stojceski@smail.inf.fh-bonn-rhein-sieg.de
userPassword: {crypt}$1$uQSw.ohy$XuiRSCq0kp...
```

Abb. 14: Typischer LDAP-Eintrag für Benutzerinformationen

3.4 Linux und Windows Authentifizierungs-Landschaft

UNIX- und Windows-basierte Rechner verwenden verschiedene Modelle für die Ablage von Authentifizierungs- und Benutzerinformationen und für im Netz freigegebene Ressourcen. Die implementierten Technologien sind zueinander inkompatibel. Im Folgenden werden die benutzten Technologien mit Bezug auf die Authentifizierungs-Landschaft des Fachbereichs näher betrachtet und aufgezeigt, wie eine Zusammenarbeit der beiden Systeme für eine solche heterogene Umgebung erreicht wurde.

Linux

Damit unter Unix-artigen Betriebssystemen nicht jede Anwendung einen eigenen Authentifizierungsmechanismus implementieren muss, wurde das Pluggable Authentication Modules (PAM)-Framework [Sama_95] entwickelt. PAM hat einen modularen Aufbau, so dass unterschiedliche Authentifizierungsmechanismen eingebunden und von unterschiedlichen Anwendungen verwendet werden können. PAM ist auf allen aktuellen Linux-Distributionen aktiviert.

Abb. 15 veranschaulicht das Zusammenspiel von Anwendungen (hier: FTP, login und SSH), dem PAM Application Programming Interface (API) und den darunter liegenden Authentifizierungsmechanismen. Die Anwendungen machen von der PAM Authentifizierungs-Schnittstelle Gebrauch. Wenn eine Anwendung einen Benutzer authentifizieren möchte, ruft es die PAM API auf, die gemäß der PAM-Konfigurationsdatei `pam.conf` ein entsprechendes Authentifizierungsmodul lädt und stellvertretend für die Anwendung den Benutzer authentifiziert.

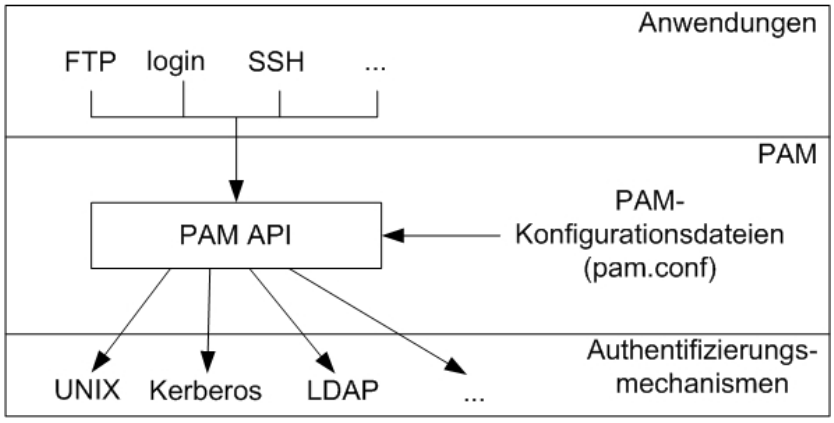

Abb. 15: PAM Architektur (in Anl. an [Sama_95])

Für jeden Authentifizierungsmechanismus existiert ein separates Modul. Die Anfrage einer Anwendung wird von PAM zum Authentifizierungsmodul weitergeleitet (hier: U-

NIX-Passwort, Kerberos oder LDAP), damit die geforderte Prozedur durchgeführt werden kann. PAM gibt das Ergebnis des Authentifizierungsmoduls an die Anwendung zurück.

Im Fachbereich wird für die Authentifizierung von Linux-Benutzer ein PAM LDAP-Modul (`pam_ldap.so`) verwendet, das stellvertretend für alle Authentifizierungsdienste die Authentifizierung durchführt. Zur Absicherung des Kommunikationskanals wird ein durch den Secure Sockets Layer (SSL) verschlüsselter Kanal benutzt (siehe Abb. 16).

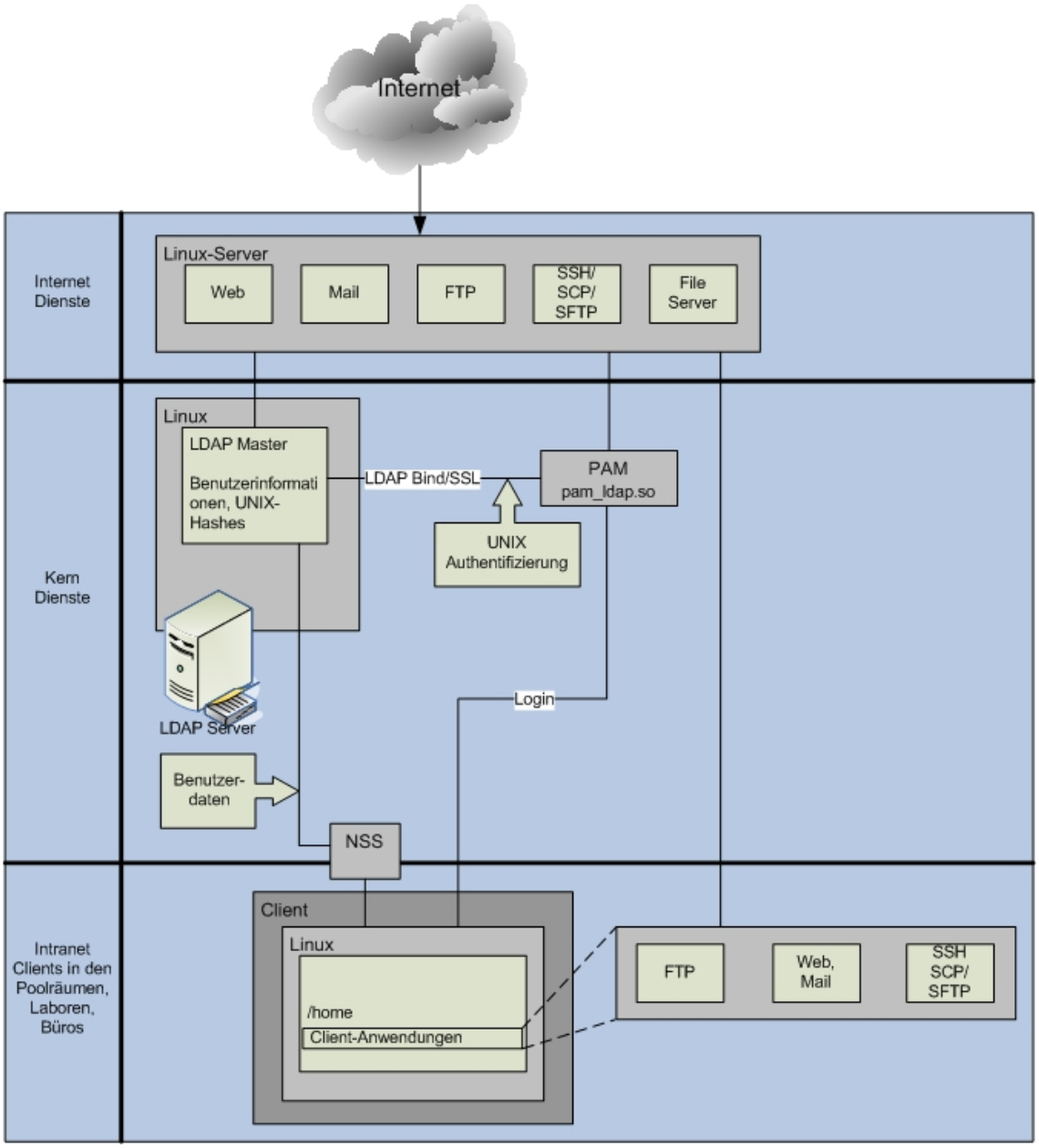

Abb. 16: Authentifizierung für Linux-Benutzer durch PAM und LDAP

Der Name Service Switch (NSS) (siehe Abb. 16) ist ein UNIX-Dienst, der Anwendungen anweist, Benutzerinformationen aus dem LDAP-Verzeichnisdienst auszulesen, falls diese nicht in den lokalen Benutzer-Konfigurationsdateien `/etc/passwd`, `/etc/shadow` oder `/etc/group` vorhanden sind.

Windows

Die Authentifizierung in Windows Domain-Umgebungen erfolgt in der Regel durch einen Windows NT Domain Controller oder ein Active Directory. Gemeinsam genutzte Dateien und Ressourcen können mit dem Microsoft CIFS/SMB-Protokoll erreicht werden.

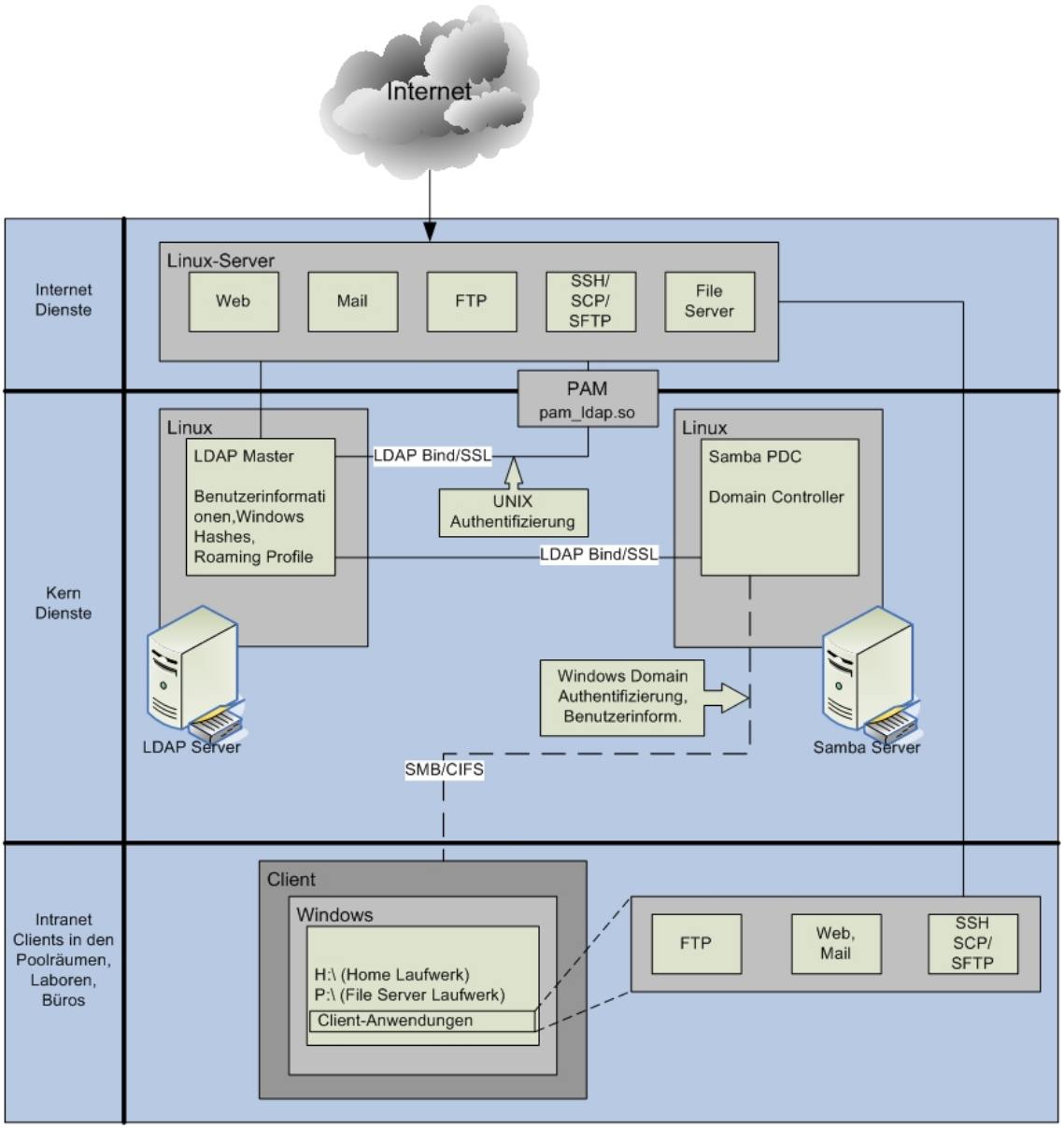

Abb. 17: Authentifizierung von Windows-Benutzern durch Samba und LDAP

Im Fachbereich emuliert Samba in der Version 3 einen Windows Primary Domain Controller (PDC), der der zentrale Anmeldedienst für eine CIFS/SMB-Arbeitsgruppe ist. Dafür zieht Samba eine Windows NT-Domäne auf, die von den Windows-Clients für den Zugriff auf Datei- und Druckdienste genutzt wird. Außerdem übernimmt Samba die Anmeldung von Windows-Benutzern in der Fachbereichsdomäne, indem es Benutzerinformationen und –Passwörter vom OpenLDAP-Server abfragt.

Der Windows-Anmeldedienst ermittelt während der Anmeldung aus dem Benutzerpasswort ein Hashwert (so genannte LanMan- und NT-Hashes) und sendet es zum Samba-Server. Samba führt daraufhin eine direkte mit SSL verschlüsselte LDAP Bind-Operation am LDAP-Server durch um Benutzerinformationen abzufragen und den Benutzer zu authentifizieren. Möchte ein Windows-Benutzer die auf UNIX basierenden Dienste nutzen, erfolgt die Authentifizierung wie oben beschrieben über das PAM LDAP-Modul.

Möchte ein Benutzer die Internet Dienste aus Abb. 16 und Abb. 17 von ausserhalb der Fachbereichsdomäne `inf.fh-brs.de` verwenden, muss bevor das LDAP-Modul die Authentifizierung übernehmen kann, ein Client das Passwort zum Server senden, der dann lokal PAM zum Authentifizieren aufruft. Im Falle von FTP wandert somit das Passwort ungeschützt in Klartext über das Netz.

4 Konzeption des Zielszenarios

Nachdem nun die wesentliche Authentifizierungsstruktur des Fachbereichs vorgestellt wurde, sollen auf ihrer Basis die Schritte aufgezeigt werden, die für die Realisierung der neuen SSO-Authentifizierungsstruktur durchgeführt werden müssen. Somit ist der folgende Schritt und damit Inhalt dieses Kapitels die konkrete Konzeption der Ziel-Authentifizierungsstruktur. Dazu ist die in Kapitel 3 vorgestellte Authentifizierungsstruktur so zu modifizieren und zu erweitern, dass eine Migration der alten Authentifizierungsstruktur hin zur Authentifizierungsstruktur mit True SSO-Eigenschaften ermöglicht wird und implementiert werden kann. Gleichzeitig untersucht dieses Kapitel die Machbarkeit einer Kerberos-basierten True SSO-Lösung. Eine prototypische Implementierung des Konzeptes und Konfigurationshinweise erfolgen in Kapitel 5.

4.1 Anforderungen an das Zielszenario

Aus Sicht des Fachbereiches sollen primär folgende Anwendungen die Grundlage für ein SSO im Fachbereich bilden:

- Linux und Windows Login über Kerberos.
- Linux und Windows Passwortänderung über Kerberos.
- Samba und AFS Filesharing.
- Kerberisierte Dienste wie OpenSSH Version 2 und die Web-Server Apache 1 und 2.

In einem weiteren Schritt, jedoch nicht in dieser Arbeit, wird die Integration der Anwendungen Sendmail und Dovecot angestrebt.

Das Ziel ist das einmalige Anmelden in der Fachbereichsdomäne `inf.fh-bonn-rhein-sieg.de` an einem Rechner, um für eine Sitzung die oben genannten Dienste ohne weitere Anmeldevorgänge verwenden zu können.

4.2 Migrationskonzept

Sinn und Zweck eines Migrationskonzepts ist es Aufgaben und Abläufe zur Überführung relevanter Systemteile eines Altsystems auf die neue Zielumgebung zu beschreiben. Es bildet die Grundlage für die Migration eines bestehenden Systems auf ein

neues System. Dieses Unterkapitel stellt einen Migrationsverlauf sowie ein Migrationskonzept vor, das die Architektur des Altsystems um Proxy-based True SSO-Eigenschaften bereichert.

Für die Migration zu einer Kerberos-basierten SSO-Lösung in Zusammenhang mit dem AFS sollte aufgrund der hohen Komplexität des Themas und erheblicher Änderungen im bestehenden System eine stufenweise Einführung, d.h. eine sanfte Migration angestrebt werden (siehe Abb. 18). Dieses Vorgehen erlaubt Vorgänge möglichst schnell rückgängig zu machen, für den Fall, dass unvorhergesehene Komplikationen auftreten und sorgt für einen sukzessiven Aufbau fehlenden Know-hows. Außerdem erlaubt dieses Vorgehen, dass Anwender sich an die neuen Funktionalitäten gewöhnen können.

Der vorgeschlagene Ablauf zur Umsetzung des Migrationskonzepts umfasst drei Phasen (siehe Abb. 18):

- Servermigrationsphase:
  - Wiederherstellungsplan (Rollbackstrategie) entwickeln.
  - Kerberos Realm planen und einführen.
  - AFS-Zelle planen und einführen.

- Umstellungsphase:
  - Benutzerdaten in das AFS migrieren.
  - LDAP-Server und Verzeichniseinträge modifizieren und erweitern.
  - Samba-Server um AFS-Unterstützung erweitern.
  - LDAP-PAM-Modul durch ein Kerberos/AFS-PAM-Modul ersetzen.

- Erweiterungsphase:
  - Anwendungen kerberisieren
  - Kerberisierte Anwendungen hinzufügen.

In den nachfolgenden Kapiteln werden die einzelnen Phasen näher beschrieben. Es wird gezeigt, welche Änderungen zur Migration durchzuführen sind und an welcher Stelle die migrierten Systemteile in das Neusystem zu integrieren sind, um das Altsystem um True SSO-Eigenschaften zu erweitern.

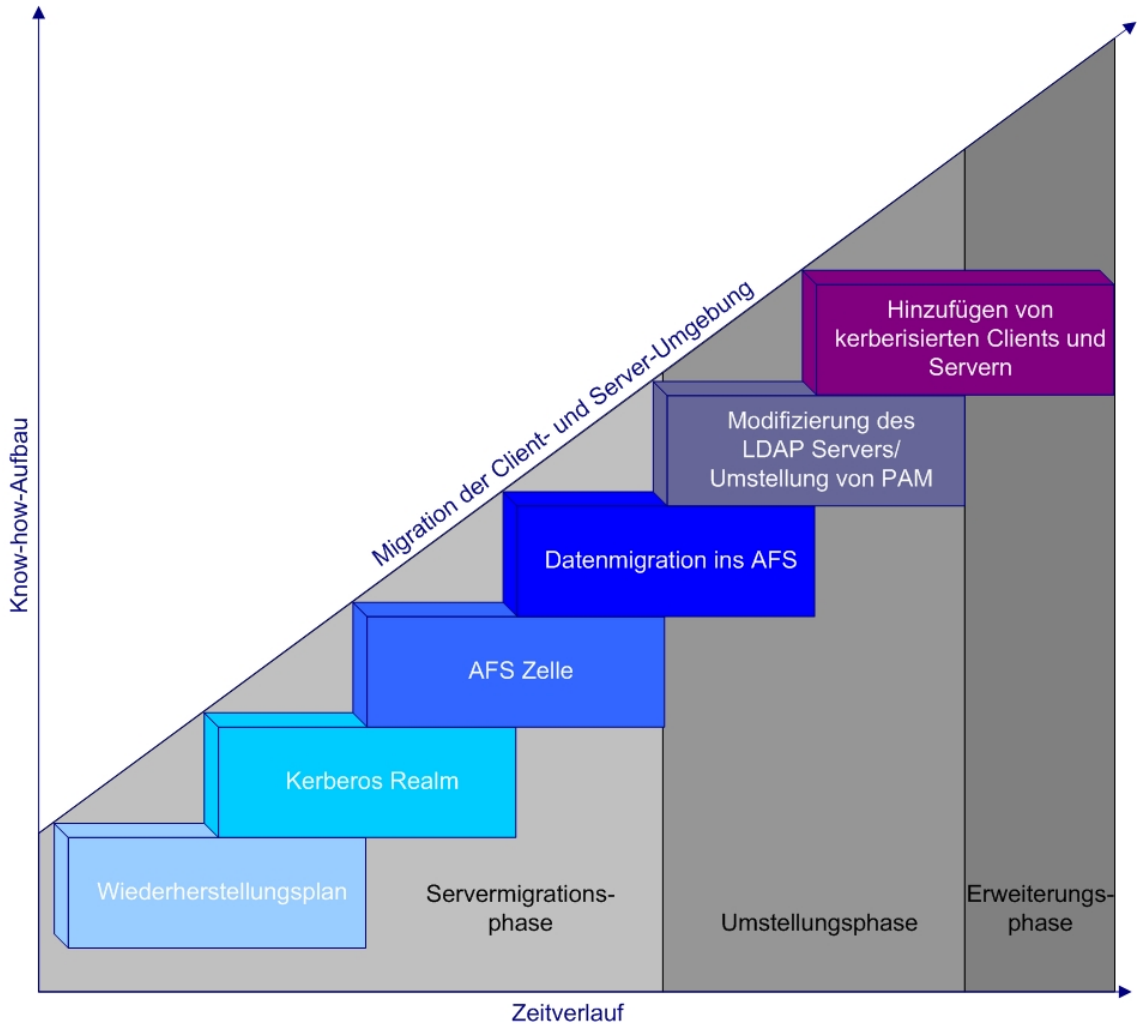

Abb. 18: Vorschlag für einen sanften Migrationsablauf

Die Abbildungen Abb. 19 und Abb. 20 zeigen Architekturvorschläge für die Linux- und Windows-Umgebung für das Neusystem. Es werden die Interaktionen des Systems zu seiner Umgebung und Neukomponenten des Neusystems gezeigt. Dies sind für Linux- und Windows-Umgebungen folgende Neukomponenten:

- Kerberos KDC. Übernimmt die Authentifizierung von Benutzern.
- Kerberisierte Anwendungen. Lösen die bisherigen Dienste ab.
- OpenAFS Server und Client. Übernehmen die Datenablage.
- Austausch des PAM LDAP-Modul durch ein Kerberos/AFS-Modul.
- Generic Security Service Application Program Interface (GSS-API) [Linn_00] und Simple and Protected GSS-API Negotiation (SPNEGO) [Zhu_05]. Werden von Anwendungen verwendet, die Kerberos nicht nativ unterstützen (siehe Kapitel 4.5.1).

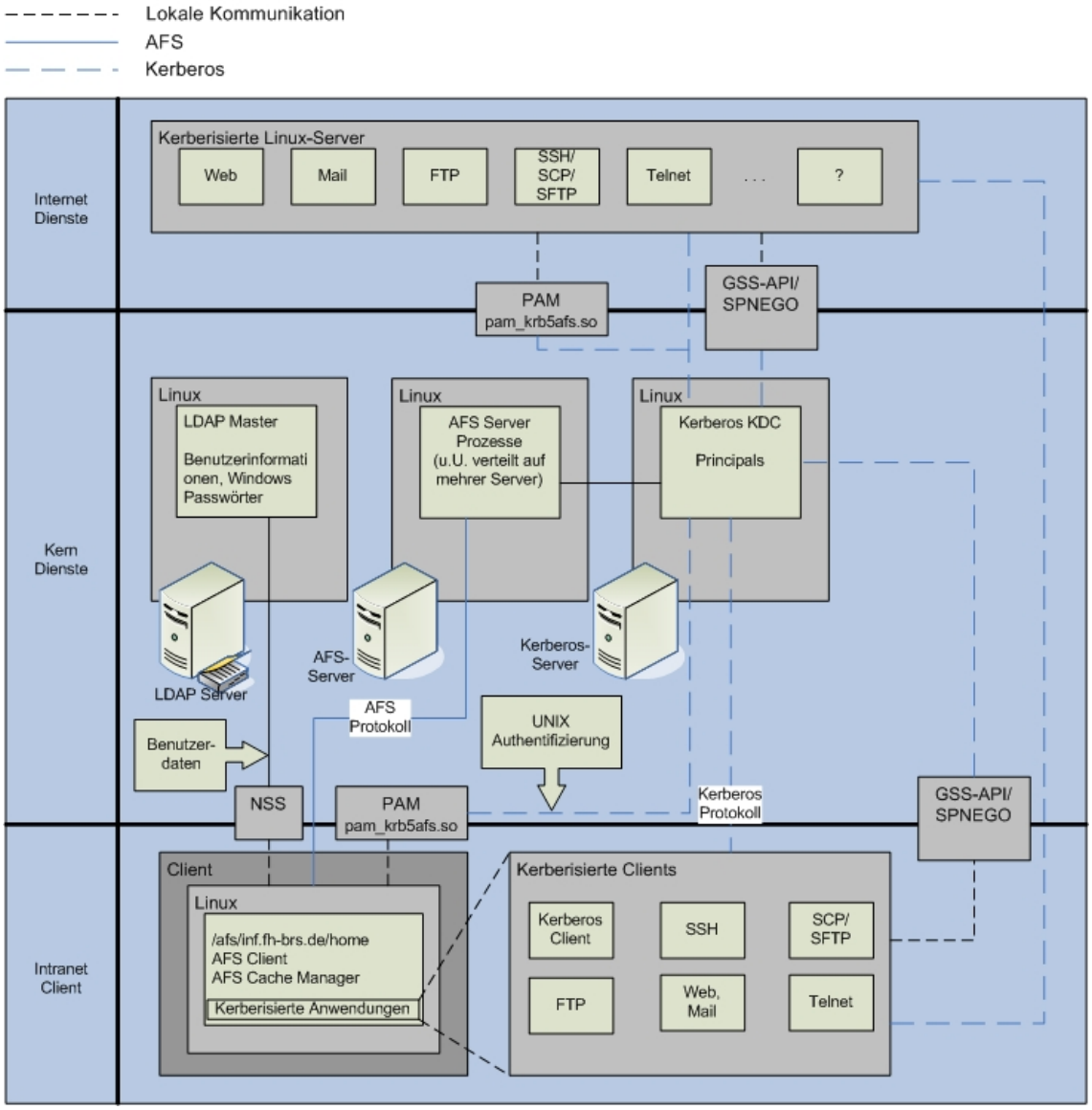

Abb. 19: Vorschlag für die Architektur des Neusystems für die Linux-Umgebung

Die Authentifizierung von Linux-Benutzern übernimmt ein Kerberos/AFS PAM-Modul, das während der Anmeldung Kerberos- und AFS-Tickets beschafft. Die Tickets können für den Zugriff auf kerberisierte Dienste verwendet werden. Die Benutzerinformationen werden von Anwendungen wie im Altsystem über NSS angesteuert.

Windows-Benutzer authentifizieren sich während der Anmeldung gegenüber dem Samba- als auch dem Kerberos-Server, der Tickets für eine Sitzung ausstellt. Neben dem direkten Zugriff über einen AFS-Client, kann auch über den Samba-Server durch das SMB/CIFS Protokoll auf AFS-Daten zugegriffen werden. Dafür müssen Windows- und Kerberos-Passwort übereinstimmen. Die nachfolgenden Kapitel gehen näher auf die einzelnen Migrationsphasen und den Aspekten des Neusystems ein.

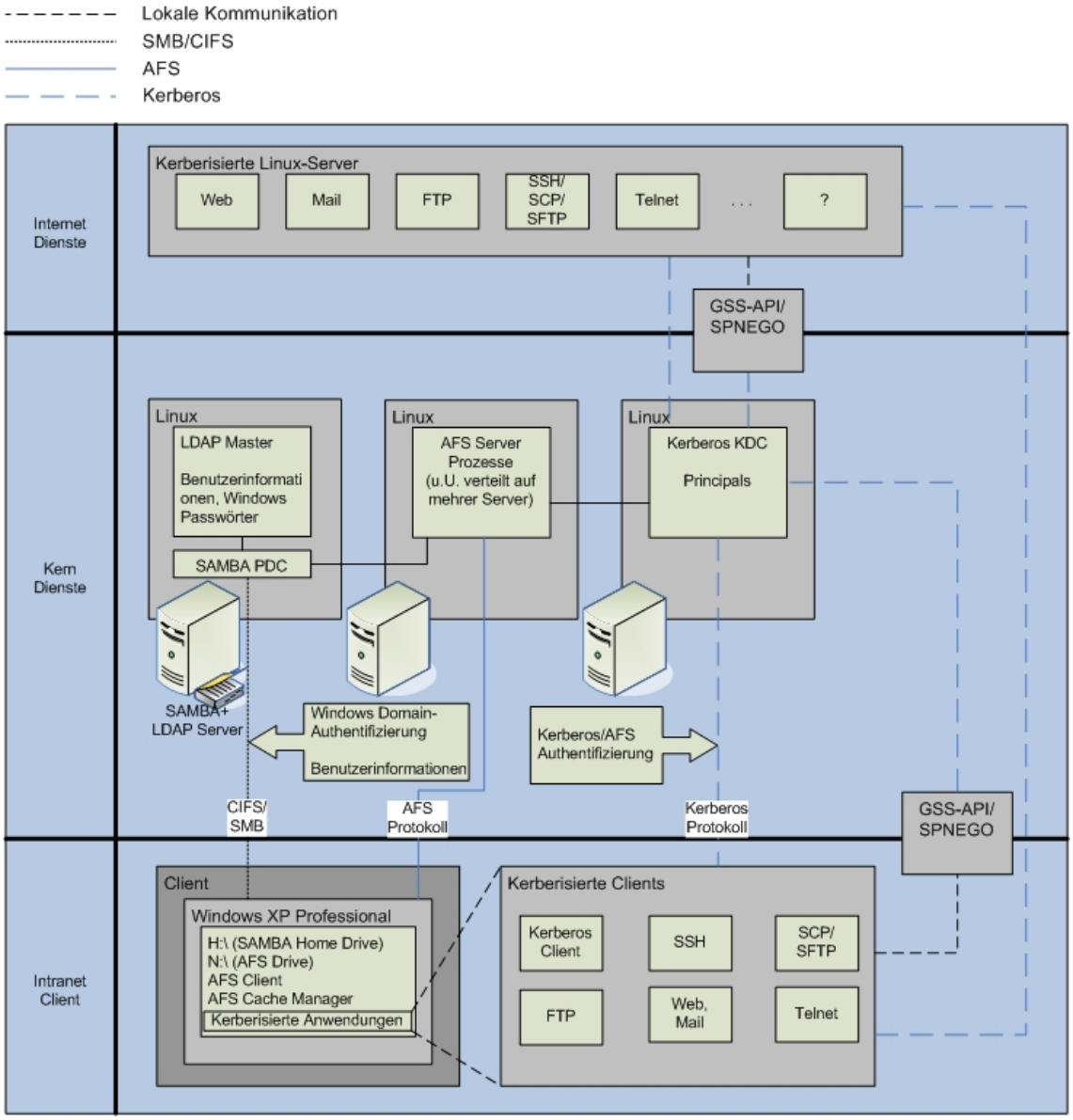

Abb. 20: Vorschlag für die Architektur des Neusystems für die Windows-Umgebung

4.3 Servermigrationsphase

4.3.1 Wiederherstellungsplan

Der Wiederherstellungsplan enthält Informationen und Anweisungen wie das System zu jedem Zeitpunkt des Migrationsprozesses in den Ausgangszustand zurückgesetzt werden kann.

Ein Wiederherstellungsplan soll im Falle eines Scheiterns der Migration den reibungslosen Betrieb des Systems ermöglichen, so dass Benutzer ohne Unterbrechung auf Ihre Daten zugreifen können.

Der Wiederherstellungsplan schließt somit mit ein, vor der Migration für die Sicherung aller Benutzerdaten sowie kritischer Dienste wie dem Domain Name System (DNS) zu sorgen. Der Umstieg in eine Kerberos-basierte Architektur erfordert in der Regel Eingriffe in den DNS-Einträgen, z.B. in Form von DNS SRV-Einträgen, wie sie in [Gulb_00] beschrieben sind (siehe Kapitel 4.3.2, Tabelle 3). SRV-Einträge ermöglichen es Clients, den Hostnamen und die Portnummer eines Servers für ein spezifisches Protokoll oder einer Domain zu lokalisieren.

4.3.2 Planung und Einführung der Kerberos Realm

Nachdem ein Wiederherstellungsplan entwickelt wurde, folgt die Planung und Einführung der Kerberos Realm. Im Wesentlichen erfordert die Implementierung einer Realm folgende Schritte:

- Auswahl einer geeigneten Kerberos-Implementierung.
- Planung der Realms.
- Planung der Principals, die Mitglieder einer Realm sein sollen.
- KDC-Planung.
- Auswahl geeigneter Client- und Server-Software. (siehe Kapitel 4.5.1)

Auswahl einer geeigneten Kerberos-Implementierung

Zurzeit sind zwei weit verbreitete Kerberos-Implementierungen verfügbar:

- MIT Kerberos V5 [MIT_06] (Version 1.4.3, Stand: 01.03.06).
- Heimdal Kerberos 5 [Heim_06] (Version 0.7.2, Stand: 01.03.06).

MIT Kerberos	Heimdal Kerberos
• Implementiert V4 und V5 des Kerberos Protokolls • Unterstützt alle neueren Verschlüsselungstypen (z.B. AES, RC4) • Master Key nicht änderbar • Microsoft kompatibel • Beinhaltet eine GSS-API, die von vielen Anwendungen unterstützt wird	• Implementiert V4 und V5 des Kerberos Protokolls • Unterstützt alle neueren Verschlüsselungstypen (z.B. AES, RC4) • Master Key änderbar • Microsoft kompatibel • Integrierte OpenAFS-Unterstützung • Unterstützt inkrementelles Backup • OpenLDAP als Backend nutzbar

Tabelle 2: Einige Eigenschaften von MIT Kerberos und Heimdal Kerberos

Tabelle 2 fasst die wichtigsten Eigenschaften der Kerberos-Implementierungen von MIT und Heimdal zusammen (Stand: 01.03.06).

Planung der Realms

Als nächstes wird darüber entschieden, ob eine oder mehrere Kerberos Realms eingeführt werden sollen. Bei Einführung mehrerer Realms muss zudem definiert werden in welcher Beziehung die Realms zueinander stehen. Die Definition von Cross-Realm Beziehungen entscheidet darüber, ob ein in einer Realm angemeldeter Benutzer die Ressourcen einer anderer Realm benutzen darf. Zudem erleichtert die Aufteilung die Verwaltung der Kerberos-Landschaft, da die Administration auf die verschiedenen Realms delegiert werden kann.

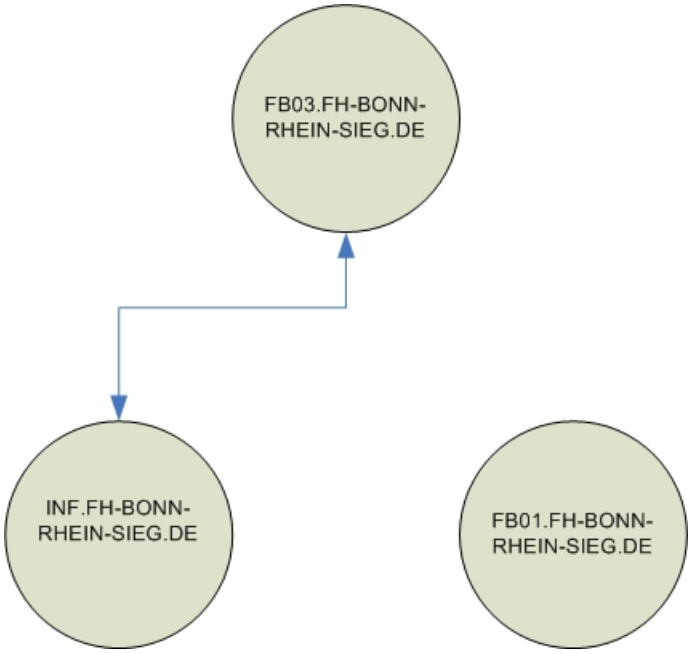

Abb. 21: Kerberos Cross-Realm Zweiwege-Beziehung

Abb. 21 veranschaulicht beispielhaft eine Cross-Realm Zweiwege-Beziehung zwischen den Realms `FB03.FH-BONN-RHEIN-SIEG.DE` und `INF.FH-BONN-RHEIN-SIEG.DE`. Die Realm `FB01.FH-BONN-RHEIN-SIEG.DE` ist von den anderen abgetrennt und für Benutzer anderer Realms nicht erreichbar. Die Realms werden in der Regel nach dem DNS-Domain-Namen, nur in Grossbuchstaben benannt.

Möchte ein Benutzer aus `INF.FH-BONN-RHEIN-SIEG.DE` auf Service Provider aus `FB03.FH-BONN-RHEIN-SIEG.DE` zugreifen, fordert er zunächst in seiner Realm ein TGT-Ticket für `FB03.FH-BONN-RHEIN-SIEG.DE`. Mit diesem TGT kann der Client

direkt mit dem KDC aus FB03.FH-BONN-RHEIN-SIEG.DE kommunizieren und Service Tickets anfordern, die er für den Zugriff auf SP verwenden kann. Auf Cross-Realms wird in diesem Kapitel nicht mehr näher eingegangen, da die hier beschriebene Migration zunächst nur von einer Realm für den Fachbereich Informatik ausgeht.

Besonders wichtig für eine funktionierende Kerberos-Infrastruktur ist eine Zeitsynchronisierung aller Hosts in einer Realm. Die Zeitabweichung zwischen den Hosts darf nicht um mehr als wenige (i.d.R. fünf) Minuten abweichen. Für diesen Zweck kann das Network Time Protocol (Version 3) (NTP) [Mill_92] eingesetzt werden. Eine freie Implementierung von NTP ist z.B. auf [NTP_06] beziehbar.

Für einen ordnungsgemäßen Betrieb der Kerberos-Infrastruktur müssen korrekte DNS-Einträge vorhanden sein. Insbesondere die DNS-Abbildung zur Auflösung von Hostnamen in IP-Adressen und umgekehrt sei hier erwähnt. Für jeden Host in einer Kerberos Realm muss ein Eintrag vorhanden sein, der den Hostnamen in eine IP-Adresse und umgekehrt abbildet. Dabei sollten alle Hostnamen auf ihren Voll qualifizierten Domain Namen (FQDN) gesetzt werden. Ein Host-Principal wird in Kerberos durch die Syntax host/FQDN@REALM, z.B. host/pc01.inf.fh-brs.de@INF.FH-BRS.DE abgebildet. Ein Client, der sich mit dem Hostnamen pc01 ausgibt, kann nicht in der Kerberos-Datenbank gefunden werden, weil kein Datenbankeintrag host/pc01@INF.FH-BRS.DE existiert. Die Folge ist eine Fehlerfunktion.

Kerberos kann DNS zur Lokalisierung von Kerberos-Diensten in einer Domain mittels DNS SRV Einträgen nutzen, wie sie in [Gulb_00] spezifiziert sind (siehe Tabelle 3). In der Tabelle sind zwei Kerberos-KDCs, kdc_master und kdc_slave, aufgeführt, die beide auf den Standard TCP- und UDP-Port 88 lauschen. Das Feld Priority weist Clients an, den Rechner kdc_master zu bevorzugen. Ein weiterer aufgeführter Kerberos-Dienst ist kpasswd, der für Passwortänderungen zuständig ist und auf dem UDP-Port 464 lauscht.

Außerdem kann Kerberos DNS TXT Einträge nutzen, um den Domainnamen des Fachbereichs auf einen gültigen Realm abzubilden (siehe Tabelle 4). Durch diesen Eintrag werden alle Hosts der Domain inf.fh-brs.de angewiesen die Realm INF.FH-BRS.DE zu benutzen.

_Service._Proto.Name	Class	SRV	Prio-rity	Wei-ght	Port	Target
_kerberos._udp.INF.FH-BRS.DE	IN	SRV	1	0	88	kdc_master.inf.fh-brs.de
_kerberos._tcp.INF.FH-BRS.DE	IN	SRV	1	0	88	kdc_master.inf.fh-brs.de
_kerberos._tcp.INF.FH-BRS.DE	IN	SRV	10	0	88	kdc_slave.inf.fh-brs.de
_kpasswd._udp.INF.FH-BRS.DE	IN	SRV	1	0	464	kdc_master.inf.fh-brs.de

Tabelle 3: Beispiel für DNS SRV Einträge für Kerberos-Dienste

_Service	Class	TXT	Realmname
_kerberos.inf.fh-brs.de	IN	TXT	“INF.FH-BRS.DE“

Tabelle 4: Beispiel für einen DNS TXT Eintrag für den Dienst Kerberos

Die Erweiterung der DNS-Einträge führt dazu, dass Kerberos-Clients ohne manuelle Konfiguration das für sie zuständige KDC auffinden können. Alternativ zur DNS Konfiguration, können Kerberos-Clients die Kerberos-Konfigurationsdatei `krb5.conf` verwenden, die in dieser Arbeit verwendet wird. Diese Datei muss dann auf jedem Client installiert sein. Ein Beispiel für eine Kerberos-Konfigurationsdatei `krb5.conf` wird in Anhang A.1 vorgestellt.

Planung der Principals

Nach der Realm-Planung erfolgt die Planung darüber, welche Principals als Mitglieder in eine Realm aufgenommen werden sollen. Dazu gehören Benutzer, Hosts sowie Dienste. Die Benutzer-Principals sollten die Namen der UNIX-Konten tragen. Abb. 22 zeigt schematisch die involvierten Principals in der Realm `INF.FH-BRS.DE` und dazugehörige IP-Adressen für die Domain `inf.fh-brs.de`. Die Benutzer der Realm sind Studenten, Professoren und Mitarbeiter. Die Hosts werden durch die Rechner in den Poolräumen, Laboren und Büros repräsentiert. Beispielhaft für kerberisierte Dienste sind in der Abbildung ein LDAP- und ein WWW-Server aufgeführt.

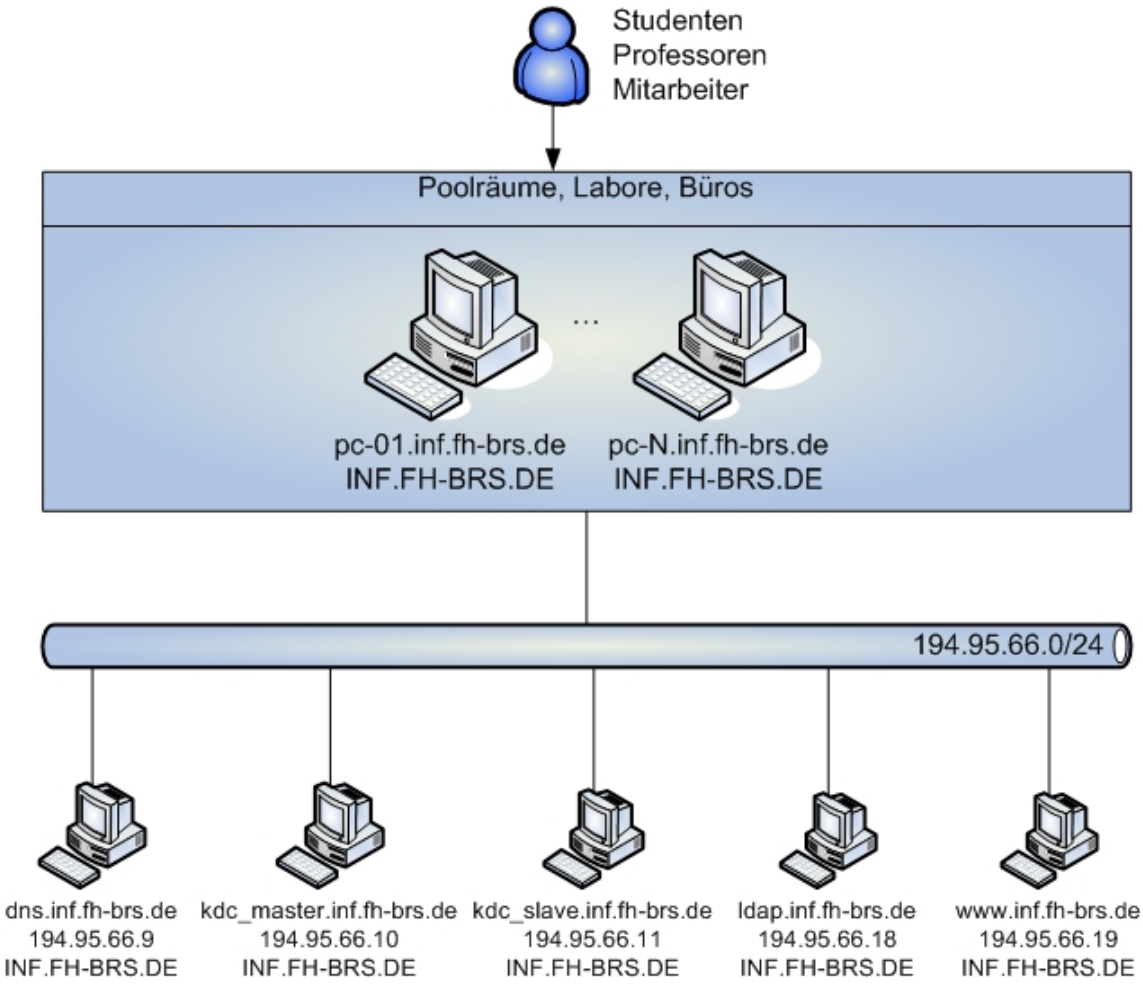

Abb. 22: Beispiel einer Kerberos Realm für den Fachbereich Informatik

KDC-Planung

Nach diesen obligatorischen Überlegungen erfolgt die konkrete Planung der Implementierungsdetails für die Realm. Dies betrifft vor allem das Kerberos KDC, ebenso wie die Anzahl der KDCs, die für die Ausfallsicherheit einer Realm zuständig sein sollen.

Damit ein Benutzer neue Tickets zur Authentifizierung gegenüber Servern anfordern kann, muss der Kerberos-Dienst immer verfügbar, ansprechbar und im Falle einer Fehlfunktion in seinen letzten stabilen Zustand zurückzuführen sein. Deshalb muss eine geeignete Auswahl für das Festplatten-Layout (z.B. Hardware-RAID mit RAID1) des Rechners auf den das KDC läuft und für die Partitionierung der Festplatte getroffen werden. Eine separate Partition für die Logdateien sollte vorgesehen werden, damit sich ein Überlauf der Logdateien nicht auf den Kerberos-Dienst auswirkt. Ebenso sollte eine geeignete Backup-Strategie, die das KDC regelmäßig sichert, gewählt werden.

In Abb. 22 sind bereits zwei physikalisch unabhängige Rechner, die die Rolle eines

KDC übernehmen abgebildet. Dies sind die Rechner `kdc_master` und `kdc_slave`. Der Rechner `kdc_master` ist das Master-KDC der Realm `INF.FH-BRS.DE` und sollte so sicher wie möglich konfiguriert und aufbewahrt werden, da er in seiner Datenbank alle mit einem Principal assoziierten Geheimnisse seiner Realm aufbewahrt. Der Rechner `kdc_slave` ist ein Slave-KDC, dient der Replikation von `kdc_master` und sorgt bei Ausfall von `kdc_master` für einen unterbrechungsfreien Betrieb und für die Verfügbarkeit der Realm.

4.3.3 Planung und Einführung der AFS-Zelle

Nachdem eine funktionierende Kerberos Realm aufgezogen wurde, kann die Planung für eine AFS-Zelle angegangen werden.

Zunächst sollen die wichtigsten AFS-Server vorgestellt werden:

- Basic OverSeer Server (BOS). Überwacht die einzelnen AFS-Serverprozesse.
- File Server (`fs`): eigentlicher Fileserver, der Dateien liefert und Änderungen ins AFS zurück schreibt.
- Kerberos Authentication Server (KAS). Basierend auf dem Kerberos Protokoll 4 verwaltet der KAS die Kerberos-Authentifizierungsdatenbank und sorgt für die gegenseitige Authentifizierung zwischen Benutzer und AFS. Der KAS wird in dieser Arbeit durch den MIT KDC ersetzt.
- Protection Server (`pts`). Während der KAS für die Authentifizierung zuständig ist, ist der Protection Server für die Autorisierung in AFS zuständig und verwaltet die Zugriffsrechte für AFS-Gruppen durch Zugriffskontrolllisten (ACLs).
- Volume Server (`vos`). Verwaltet die AFS-Volumes.
- vldb-server (`vlserver`). Verwaltet die Volume Location Database (VLDB). Der `vlserver` kennt den Standort aller AFS-Dateien und wird von den Clients zur Standortabfrage von Dateien abgefragt. Durch ihn ist im AFS die Standorttransparenz realisiert.

Im Fachbereich sind bedingt durch den Einsatz von AFS in jedem Fall tief greifende konzeptuelle Änderungen im Vergleich zur jetzigen Dateiablage erforderlich. Dies betrifft vor allem ein Umdenken bei Administratoren und Benutzern in der Handhabung von Zugriffsrechten auf Dateien. Dafür sind AFS-eigene Werkzeuge zu benutzen.

Implementierung der AFS-Zelle

Die Implementierung einer AFS-Zelle erfordert insbesondere Planungen zu folgenden Punkten:

Auf der Seite des Servers:

- Auswahl des Betriebssystems. OpenAFS unterstützt alle gängigen Linux-Varianten.
- Anzahl der AFS-Server. Durch die Verwendung von mindestens drei gut abgesicherten AFS-Servern kann eine gewisse Redundanz erzielt werden, so dass der Ausfall eines Servers minimale Konsequenzen auf die Nutzbarkeit des Systems hat.
- Partitionen auf AFS-Server. Jedem AFS File Server muss explizit eine Partition für die AFS Volumes zugeteilt werden, die in das Verzeichnis `/vicep`*`xx`* gemountet wird, wobei *xx* ein oder zwei Kleinbuchstaben sind (z.B. für die erste Partition: `/vicepa`). Die Partition wird mit einem in der AFS Dokumentation aufgelistetem Dateisystem formatiert (z.B. ext3).
- Zellenname. Der Zellenname sollte mit Sorgfalt gewählt werden, da die Erfahrungen zeigen, dass eine Änderung des Zellennamens zu Schwierigkeiten führen kann. Er wird i.d.R. durch den Kerberos Realm einer Organisation bestimmt, wird jedoch in Kleinbuchstaben ausgedrückt. Er wird in der Datei `ThisCell` und in der Datei `CellServDB` eingetragen. In `ThisCell` wird eingetragen, welcher Zelle der Rechner angehört. In `CellServDB` werden die eigene Zelle sowie Zellen weiterer Organisationen aufgelistet. Für den Fachbereich bietet es sich an den Zellennamen `inf.fh-brs.de` zu wählen.
- Registrierung des Zellennamens. Über http://www.central.org/csdb.html kann der Zellenname registriert werden, so dass sich andere Organisationen über die Existenz der Fachbereichs-Zelle informieren können.
- Struktur des AFS-Dateibaums. Hierunter fallen z.B. Planungen über AFS-Pfade für Homeverzeichnisse, Projektverzeichnisse oder Web-spezifische Verzeichnisse.
- Zugriffsrechte für Benutzer. Die Zugriffsrechte eines Benutzers definiert das AFS durch Zugriffskontrolllisten, so genannte Access Control Lists (ACL), in denen Benutzer oder Benutzergruppen und deren Zugriffsrechte definiert werden. Dem Benutzer sollten in seinem Homeverzeichnis alle Rechte zugestanden werden. Zu beachten ist, dass Zugriffsrechte in AFS nur auf Verzeichnisse

gesetzt werden können. Verzeichnisse und Dateien erben die Zugriffsrechte des darüber liegenden Verzeichnisses, die nachträglich geändert werden können.

- Quota für Volumes. Die Quota bezieht sich immer auf ein Volume (z.B. Homeverzeichnis eines Benutzers) und gibt die maximale Größe der Volumes in Kilobyte an.
- Benutzerverwaltung. AFS Gruppen- und Benutzer IDs sollten auf die jetzigen UNIX GIDs und UIDs abgebildet werden, indem bei der Erzeugung der AFS GIDs und UIDs die UNIX GIDs und UIDs vergeben werden.
- Auswahl des KDC. OpenAFS unterstützt ab Version 1.2.8 nativ Kerberos V5, so dass der von OpenAFS auf Kerberos Version 4 aufsetzende KAS gemieden werden sollte Es muss untersucht werden, welcher KDC, MIT oder Heimdal, den Anforderungen des Fachbereichs genügt (vgl. Tabelle 2, S. 37).
- Lebensdauer eines AFS-Tickets. Die Lebensdauer der Tickets entscheidet darüber, wie lange ein Benutzer für eine Sitzung auf AFS-Dateien zugreifen kann. Bei Ablauf der Gültigkeit eines Tickets kann ein Benutzer nicht mehr auf AFS-Dateien zugreifen.
- Backup-Strategie. In [Webb_05] sind AFS-typische Backup-Strategien beschrieben.
- Zusammenarbeit mit fremden Zellen. Es muss überlegt werden, ob und welche fremden AFS-Zellen in den AFS-Dateibaum des Fachbereichs eingehängt werden sollen.

Auf der Seite des Clients:

- Jeder Rechner, der auf das AFS-Filesystem zugreifen will, muss die AFS-Client-Software installiert haben. Die Client-Software besteht neben einigen Kommandos aus dem AFS-Cache-Manager, der das Interface zu den AFS-Servern darstellt und einen lokalen Cache verwaltet. Auf jedem Client sollte ein AFS Cache von mindestens 128 Megabyte vorhanden sein. Idealerweise residiert der Cache in einer eigenen Partition, die i.d.R. unter `/usr/vice/cache` gemountet und z.B. mit ext3 formatiert wird. Er beinhaltet Kopien von Dateien, die aus einem AFS Dateibaum empfangen wurden. Änderungen an den Dateien werden lokal bearbeitet, wodurch die Performance von AFS gesteigert und der Netzverkehr minimalisiert wird.
- Der AFS Dämon `afsd` muss beim Booten eines Rechners starten, damit beim

Anmeldevorgang eines Benutzers ein AFS-Ticket beschafft werden kann.

- PAM-Konfiguration (nur Linux). Die derzeitige PAM-Konfiguration muss von `pam_ldap.so` auf beispielsweise `pam_krb5afs.so` umgestellt werden, so dass das Modul beim Anmeldevorgang eines Benutzers ein Kerberos- und ein AFS-Ticket beschaffen kann.

Einbinden eines MIT- oder Heimdal-KDC zur Authentifizierung von AFS-Benutzern

Damit der AFS KAS durch einen MIT- oder Heimdal KDC ersetzt werden kann und Benutzer sich gegen diese authentifizieren können, sind im Wesentlichen folgende Schritte durchzuführen (vgl. Kapitel 5.3.2):

1) AFS-Principal in der Form `afs/inf.fh-brs.de@INF.FH-BRS.DE` im KDC erzeugen.
2) Extrahieren des AFS-Schlüssels in eine Kerberos 5 Keytab-Datei.
3) Kopieren des AFS-Schlüssel aus der Keytab-Datei in eine AFS-konforme Key-File-Datei. Die KeyFile wird benötigt, damit sich AFS gegenüber sich selbst authentifizieren kann. (siehe Kapitel 5.3.2, S. 67)

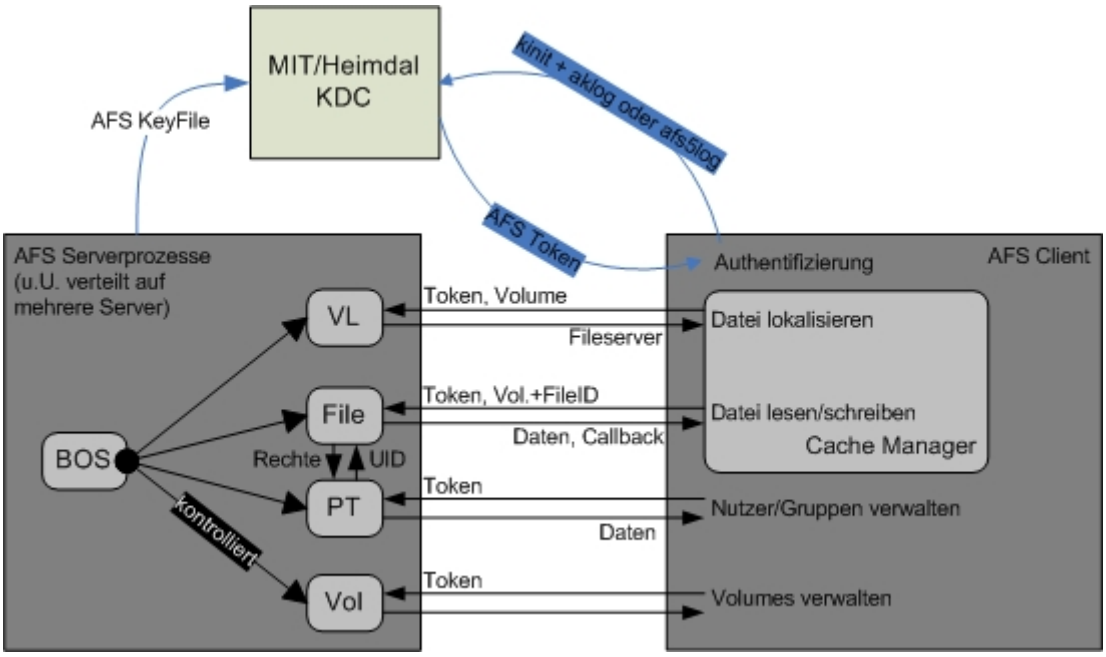

Abb. 23: AFS-Struktur für den Fachbereich (in Anl. an: [Hueb_01])

AFS kann somit in die in Abb. 23 illustrierte Struktur übergeführt werden. Der eigentliche KAS wird durch einen MIT oder Heimdal KDC ersetzt. Ein AFS-Client muss zu diesem Zeitpunkt die Kommandos `kinit`, folgend vom Kommando `aklog` (alternativ

auch `afs5log`) ausführen, um ein AFS-Ticket zu erhalten. AFS selbst authentifiziert sich mit seinem KeyFile gegenüber dem KDC. In Kapitel 4.4.3 wird gezeigt wie ein Linux-Benutzer mit Hilfe von PAM während des Anmeldeprozesses ein TGT und ein AFS-Ticket erhält ohne diese Kommandos auszuführen.

4.4 Umstellungsphase

In der Umstellungsphase erfolgt die eigentliche Umschaltung vom Alt- zum Neusystem. Während dieser Phase ist es notwendig beide Systeme parallel zu betreiben, um den Benutzern einen unterbrechungsfreien Zugriff auf ihre Daten zu ermöglichen.

4.4.1 Datenmigration

Ein wichtiger Schritt in der Umstellungsphase ist die Migration vorhandener Daten ins AFS. Dies sind insbesondere:

- Homeverzeichnisse, inklusive darin enthaltener Daten.
- Software, die über das AFS erreichbar sein soll.
- WWW-Inhalte.
- Gemeinsame genutzte Projektverzeichnisse.

4.4.2 Modifizierung des LDAP-Servers

Die derzeitige Struktur des LDAP-Verzeichnisbaumes muss nicht geändert, höchstens erweitert werden. Ebenso sind die meisten Verzeichnis-Einträge nicht von der Migration berührt. Ausnahmen bilden die folgenden Einträge:

- Erweiterung der Einträge um Kerberos-Principals. Heimdal bietet die Möglichkeit OpenLDAP als Backend für die Ablage der Kerberos-Datenbank zu wählen. Dies hat den Vorteil, dass eine Datenbank weniger gepflegt werden muss, nämlich die des Kerberos KDC. OpenLDAP kann um ein Kerberos-Schema erweitert werden, in dem Kerberos-spezifische Definitionen angegeben sind. Dies sind z.B. Informationen über die Kerberos Realm oder Principals.
- Modifizierung der Einträge für Homeverzeichnisse (UNIX und Windows). Die Einträge müssen dahin geändert werden, dass sie auf die AFS-Pfade verweisen. Beispiel für einen UNIX-Eintrag:
    - `homeDirectory: /home/dstojc2s` wird geändert auf `homeDirectory: /afs/inf.fh-brs.de/home/dstojc2s`

Eine sichere Authentifizierung gegenüber dem Verzeichnisdienst kann nach erfolgreicher Kerberos-Authentifizierung mit SASL/GSS-API erfolgen. Dadurch können z.B. Verzeichniseinträge ohne Angabe eines Passworts durchsucht und falls es die LDAP Zugriffskontrolllisten erlauben, auch modifiziert werden (z.B. wenn ein Benutzer, die ihm zugewiesen Shell ändern möchte). Dafür müssen folgende Schritte durchgeführt werden:

- OpenLDAP mit SASL-Unterstützung kompilieren.
- Ergänzung der zentralen Konfigurationsdatei `slapd.conf` um folgende Einträge:
  - `sasl-host kdc.inf.fh-brs.de # Kerberos KDC`
  - `sasl-realm INF.FH-BRS.DE # Kerberos Realm`
- Ergänzung der `slapd.conf` um reguläre Ausdrücke. Diese bilden die von SASL wiedergegebenen Authentifizierungsstrings in einen gültigen LDAP Distinguished Name ab. Dafür müssen dieser Datei entsprechende Zugriffskontrolllisten hinzugefügt werden, die die Zugriffssteuerung der Clients auf die Verzeichniseinträge definieren.

4.4.3 Linux und Windows Anmeldung

Ein wichtiger Ansatz, um das Altsystem mit True SSO-Eigenschaften auszustatten ist der Anmeldevorgang eines Benutzers an einem Rechner, der ihm ein TGT beschafft, das für die Beschaffung Dienste-spezifischer Tickets benutzt werden kann. Im Folgenden wird beschrieben, wie ein Benutzer während des Anmeldevorgangs ein TGT erhält.

Linux-Anmeldung

Unter Linux bietet es sich an für den Anmeldevorgang PAM oder ein kerberisiertes Anmelde-Werkzeug, wie es z.B. mit der MIT Kerberos-Implementierung (`login.krb5`) geliefert wird, zur Authentifizierung einzusetzen.

PAM bietet keine native Kerberos-Unterstützung, so dass es sinnvoll ist PAM nur zum lokalen Anmelden auf dem Rechner zu nutzen, damit keine Klartextpasswörter über das Netz übertragen werden. Ein Kerberos-PAM-Modul kann von allen Anwendungen zur Authentifizierung eines Benutzers aufgerufen werden, die die PAM-API aufrufen können.

Für den Fachbereich muss ein PAM-Modul eingesetzt werden, das gleichzeitig ein Kerberos und ein AFS-Ticket beschafft (vgl. Kapitel 5.3.5). Das Modul beschafft einem Benutzer ein TGT für die Nutzung kerberisierter Dienste und gleichzeitig ein AFS-Ticket. Mit dem AFS-Ticket wird dem Benutzer ermöglicht nach dem Authentifizieren in seinem AFS-Homeverzeichnis zu landen und auf AFS-Dateien zuzugreifen.

Abb. 24 verdeutlicht den Anmeldevorgang für einen Linux-Benutzer.

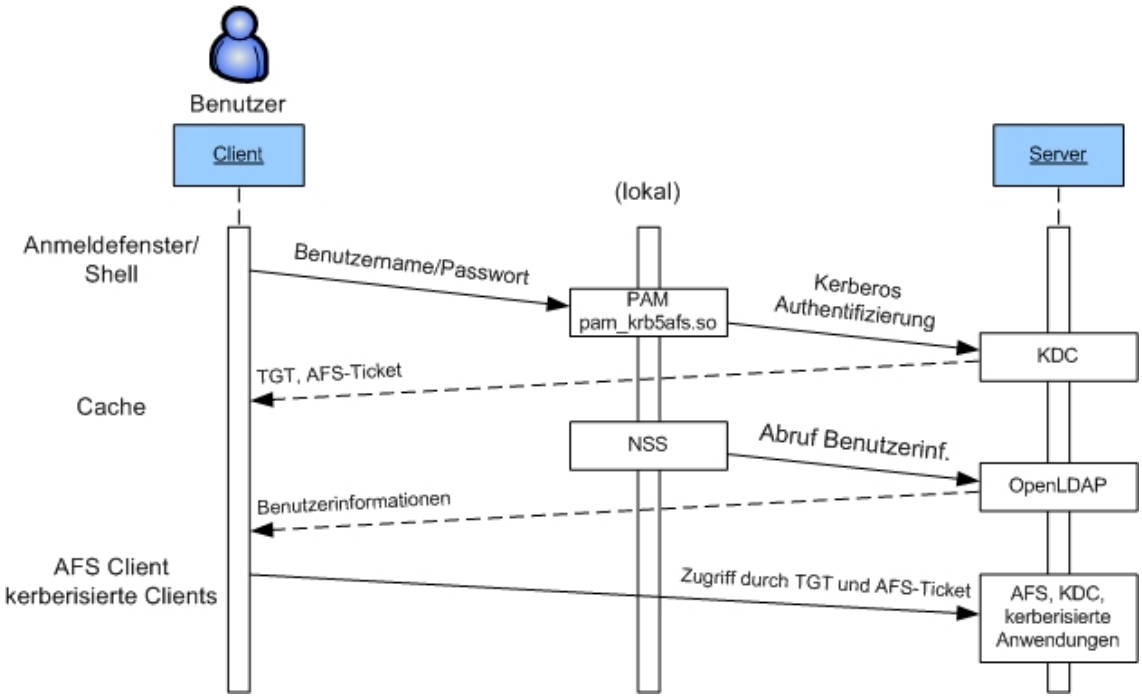

Abb. 24: Anmeldevorgang für Linux-Benutzer

Windows-Anmeldung

Der derzeitige Anmeldevorgang eines Windows-Benutzers erfolgt mit Hilfe eines SAMBA PDC-Servers (vgl. Kapitel 3.4), der die notwendigen Benutzerdaten aus dem LDAP-Server bezieht. Die derzeitige SAMBA-Konfiguration soll beibehalten werden. Diese unterstützt in der derzeitigen Konfiguration keine Anmeldung über ein nicht-Microsoft-KDC (z.B. MIT oder Heimdal). Deshalb muss das Kerberos-TGT sowie das AFS-Ticket über Zusatzsoftware beschafft werden. Dafür können folgende Software-Pakete installiert werden:

- MIT Kerberos for Windows [MIT_06] in der derzeitigen Version 3.0, das auf den Plattformen Windows 2000/XP/2003 lauffähig ist. Das Paket beinhaltet vor allem:

- Kerberos V5 Bibliothek (Version 1.4.3).
- Kerberos V5 GSS-API Bibliothek. Wird von vielen Windows-Anwendungen mit GSS-API-Unterstützung benutzt.
- Network Identity Manager. Graphisches Benutzerinterface zur Verwaltung von Kerberos-Tickets. Durch den modularen Aufbau der Software ist es möglich Plugins einzubinden. Dazu gehört auch ein AFS-Plugin, das es ermöglicht AFS-Tickets zu beschaffen und verwalten. Das Plugin ist auf [Secu_06] verfügbar.
- Kerberos Kommandozeile-Werkzeuge.

- AFS Client-Software für Windows [OAFS_06].

Beide Pakete verfügen über eine Option, die als Integrated Logon bezeichnet wird. Sie sorgt dafür, dass ein Benutzer während des Anmeldevorgangs die entsprechenden Tickets erhält. Voraussetzung ist allerdings, dass das Windows- und das Kerberos-Passwort übereinstimmen.

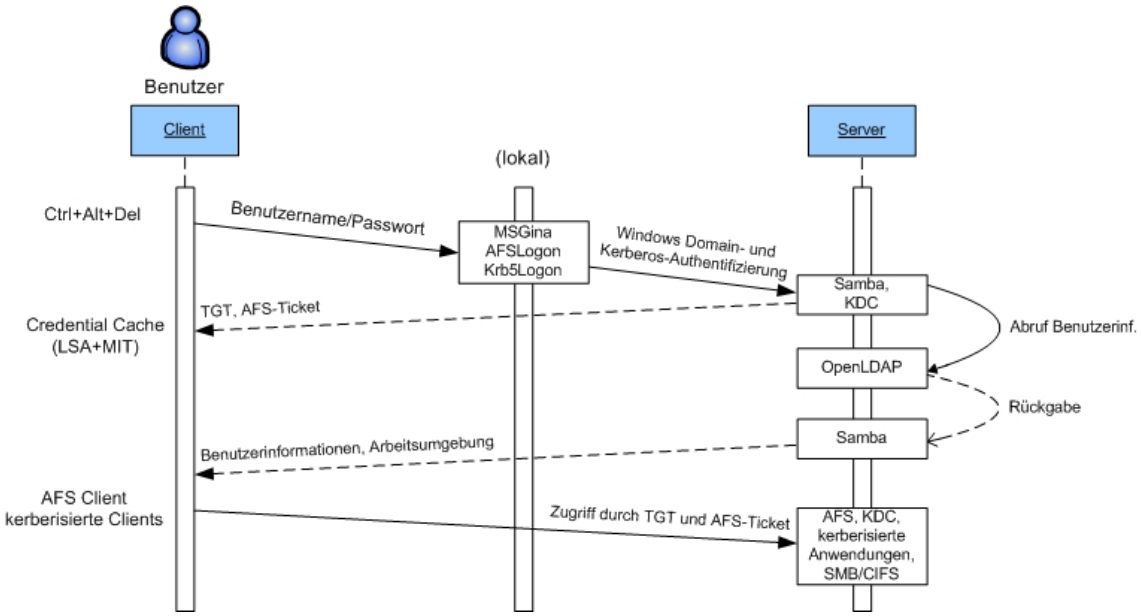

Abb. 25: Anmeldevorgang für Windows-Benutzer

Es gibt auch die Möglichkeit, dass SAMBA-Passwort abzuschaffen und Windows-Benutzer gegen einen nicht-Microsoft KDC zu authentifizieren. Dafür liefert Microsoft das Werkzeug `ksetup.exe`. `ksetup.exe` erlaubt die Konfiguration eines nicht-Microsoft KDC, so dass Benutzer z.B. gegen einen MIT KDC in einer Kerberos Realm authentifiziert werden können. Das so erhaltene TGT wird in den Microsoft Credential

Cache abgelegt und von der Microsoft Local Security Authority (LSA) verwaltet. Der Microsoft Credential Cache kann auch vom Network Identity Manager ausgelesen werden und verwaltet werden. Der Vorteil ist, dass der Anmeldevorgang unter Windows kerberisiert wird und keine Passwörter über das Netz übertragen werden. Eine weitere Folge dieses Vorgehens ist, dass Windows-Benutzer sich nicht an einer Windows-Domäne anmelden und manche Dienste, für die das Microsoft NTLM Passwort benötigt wird, nicht wie bisher genutzt werden können.

4.4.4 Samba

Die eingesetzte Samba Version 3 erlaubt nur bedingt, dass sich Windows-Benutzer an einen MIT oder Heimdal Kerberos 5 KDC authentifizieren können. Ein Benutzer kann sich an einen nicht-Microsoft KDC nur als Mitglied einer Arbeitsgruppe und nicht als Mitglied einer Windows-Domäne anmelden. Die derzeitige Samba-Konfiguration bietet nur das Anmelden als Mitglied einer Windows-Domäne und soll beibehaltet werden. Dieses Problem könnte mit der demnächst verfügbaren Samba 4 Distribution gelöst werden, die einen integrierten Kerberos 5 KDC beinhaltet und die SPNEGO/GSS-API unterstützt (vgl. [Bart_05]). Dazu beitragen soll auch die noch nicht verfügbare MIT Kerberos V5 Version 1.5, die eine bessere Samba und Windows-Unterstützung verspricht (vgl. [Hegd_05]).

Samba unterstützt OpenAFS. Für eine nahtlose Integration von Samba und OpenAFS sind aufgrund der unterschiedlichen Semantik von SMB und AFS Änderungen in Samba notwendig. Samba muss mit AFS-Unterstützung kompiliert und installiert werden. Samba agiert dann zwischen SMB/CIFS und AFS als Proxy.

In der Konfigurationsdatei von Samba kann eine Freigabe auf AFS eingerichtet werden, so dass Windows-Benutzer über das SMB/CIFS Protokoll auf ihre Daten im AFS zugreifen können. So ist es auch möglich die Windows-Arbeitsumgebung (z.B. Roaming-Profiles) aus dem AFS zu laden. Im Testbetrieb ergaben sich jedoch einige Schwierigkeiten bezüglich dem Zurückschreiben einiger Windows-Verzeichnisse des Roaming Profiles ins AFS. Dem Autor war es aus zeitlichen Gründen nicht mehr möglich diese Problematik anzugehen. Eine mögliche Lösung dieses Problems besteht darin, das von Samba angehängte Netzlaufwerk zu meiden und stattdessen ein von AFS angehängtes Laufwerk für den Zugriff auf AFS-Dateien und die Zugriffsrechtesteuerung zu benutzen. AFS wird mit einem lokalen SMB-Server geliefert, der den Zugriff auf AFS über die Freigabe `\\AFS` auf dem Windows-Client erlaubt. Weitere

Hinweise zur Problematik zwischen Samba 3 und AFS sowie Lösungsansätze sind in [Lend_04] und [Dyer_04] beschrieben.

4.5 Erweiterungsphase

4.5.1 Anwendungen kerberisieren

Nach der Realm- und KDC-Planung, erfolgt die Auswahl geeigneter Client- und Server-Software, um die Vorzüge einer Kerberos-basierten Infrastruktur auszuschöpfen. Die im Fachbereich vorhandenen Anwendungen können entweder kerberisiert oder müssen durch kerberisierte Anwendungen ersetzt werden. Im Wesentlichen können Client-Server-Anwendungen, die Kerberos unterstützen, zwei Methoden implementieren, um ein Kerberos-Ticket zu erhalten:

1) Ein Client sendet dem Server das Benutzerpasswort in Klartext. Der Server kontaktiert dann das KDC im Namen des Benutzers, um ein TGT zu erhalten und es dem Benutzer zurückzusenden. Das TGT kann dann für kerberisierte Dienste verwendet werden. Ein typischer Vertreter dieser Methode ist PAM. Eine Kerberos-Infrastruktur, die nur diese Art Client-Server-Anwendungen integriert, erlaubt kein True SSO, da ein Benutzer sein Passwort für jeden Dienst neu eingeben muss.

2) Client und Server sind kerberisiert, d.h. sie unterstützen nativ das Kerberosprotokoll oder den Simple Authentication and Security Layer (SASL) [Myer_97]. Dadurch erst wird True SSO ermöglicht. Eine Kerberos-Infrastruktur, die nur solche Client-Server-Anwendungen integriert, erfordert nach dem Anmelden eines Benutzers an einen Rechner keine weiteren Eingriffe seitens des Benutzers beim Authentifizieren gegenüber anderen Diensten für eine Sitzung.

In der Regel ist für eine Kerberos-Infrastruktur eine Mischung aus den oben genannten Client-Server-Anwendungen erforderlich. Das liegt daran, dass nicht sehr viele Anwendungen das Kerberosprotokoll nativ unterstützen. Es sollte dennoch angestrebt werden, nur Client-Server-Anwendungen mit nativer Kerberos-Unterstützung zu integrieren, damit die von Kerberos gebotene Sicherheit nicht von Client-Server-Anwendungen der ersten Methode untergraben werden kann, indem z.B. Klartextpasswörter über das Netz versendet werden. Für Anwendungen der ersten Methode kann ein Kerberos PAM-Modul verwendet werden, das einen Benutzer stellvertretend für eine Anwendung

authentifiziert.

Für Anwendungen der zweiten Methode gilt, dass der Client und der Server kerberisiert sein müssen. Diese Anwendungen transportieren das Passwort nicht im Klartext oder gar nicht über das Netz. Außerdem ist mit den Sessionkeys ein verschlüsselter Datenaustausch möglich, z.B. für die unsichere Anwendung `ftp`. Für jeden kerberisierten Server muss ein Schlüssel im KDC angelegt sein, wobei dieser in der Regel zusätzlich auf der Festplatte, auf der die Anwendung läuft, in einer speziellen Kerberos Keytab-Datei extrahiert wird. Pro kerberisiertem Server sollte eine Keytab-Datei vorhanden sein und nur von diesem Server lesbar sein. Sie wird vom Server genutzt, um sich beim KDC zu authentifizieren.

Die MIT und Heimdal Kerberos-Distribution liefern Server für FTP und den UNIX-Anmeldedienst `login`, die das Kerberos-Protokoll nativ unterstützen. Die Standard Server-Anwendungen wie `ftpd` und `login` können somit durch kerberisierte Anwendungen ausgetauscht werden. Diese bieten dieselbe Funktionalität wie die Standard Server, sind jedoch um Kerberos-Funktionalitäten erweitert.

Um die MIT-Server nutzen zu können, müssen auch die Clients kerberisiert werden. Dazu liefern MIT und Heimdal z.B. die Clients `rlogin` (remote login), `ftp` und `ksu` (für den Root-Benutzer), die ebenfalls um Kerberos-Funktionalitäten erweitert sind und benutzertransparent die Kerberos-Tickets zur Authentifizierung benutzen und optional den Datenaustausch mit Hilfe der Sessionkeys verschlüsseln.

Viele Anwendungen, die Kerberos nicht nativ unterstützen, unterstützen SASL. SASL beinhaltet unter anderem den Mechanismus Generic Security Service Application Program Interface (GSS-API) (siehe Abb. 26), mit dem verschiedene Authentifizierungsmechanismen gestartet werden (ähnlich PAM) können. Zu diesen Authentifizierungsmechanismen gehört vor allem das Kerberos-Protokoll, das nativ unterstützt wird. GSS-API ist in [Linn_00] spezifiziert. Eine freie Implementierung von SASL ist durch das Cyrus SASL Projekt [Cyru_06] verwirklicht. Microsofts Implementierung zur GSS-API heißt Security Support Provider Interface (SSPI) und ist kompatibel zur GSS-API.

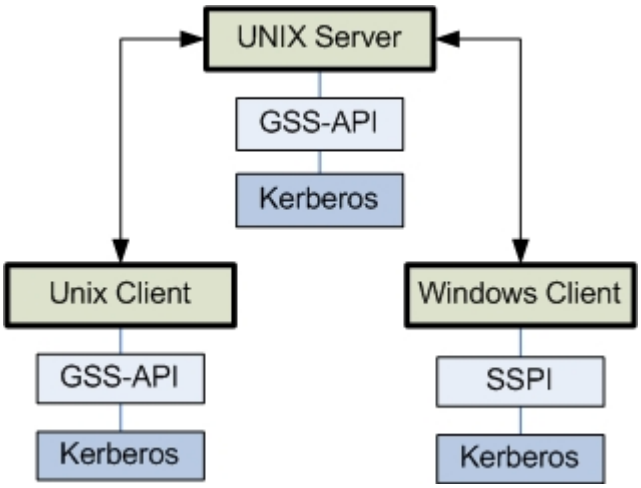

Abb. 26: GSS-API und SSPI Authentifizierungs-Mechanismus

Im Folgenden werden einige Anwendungen vorgestellt, die um Kerberos-Funktionalitäten erweitert sind oder die GSS-API verwenden und für das Zielszenario von Bedeutung sind. Webbasierte Anwendungen werden in Kapitel 4.5.2 vorgestellt.

Linux-Client und -Server mit GSS-API-Unterstützung:

- OpenLDAP. Um den Zugriff auf Informationen des OpenLDAP-Servers zu steuern und Benutzer gegenüber den Server zu authentifizieren, kann OpenLDAP die Cyrus SASL Bibliotheken verwenden. Die in der OpenLDAP-Suite enthaltenen Clients wie `ldapsearch`, `ldapadd` oder `ldapdelete` können dadurch ohne Angabe eines Passworts benutzt werden.
- OpenSSH [OSSH_06]. Der SSH Dämon `sshd` erlaubt das entfernte Anmelden an einem Host durch ein verschlüsselten Ende-zu-Ende Kanal. Dafür enthält die OpenSSH Suite die Clients `ssh`, `sftp` und `scp`, die unsichere Anwendungen wie `telnet` und `ftp` ablösen sollen. OpenSSH erlaubt Kerberos und AFS Ticket Passing, so dass das Anmelden am entfernten Host durch Eingabe eines Passworts entfällt. Während Version 1 Kerberos nativ unterstützt, kann ab Version 2 die GSS-API zur Authentifizierung benutzt werden. Die Implementierungen von OpenSSH bis zur Version 2 können das AFS-Ticket vor der Überprüfung der Identität des Nutzers zum Zielrechner übertragen. Dadurch kann der SSH-Server den Benutzer in sein AFS-Homeverzeichnis leiten.

Da die Server alle auf Linux laufen sollen, werden für das Betriebssystem Windows nur Clients vorgestellt, die überwiegend über die GSS-API auf das Kerberos-Protokoll

zugreifen. Die meisten Clients machen von der MIT GSS-API Gebrauch, so dass MIT Kerberos for Windows auf Windows installiert sein muss, das die MIT GSS-API Bibliothek enthält.

Windows-Clients mit GSS-API Unterstützung:

- PuTTY [Putt_06], das hauptsächlich von Simon Tatham geschrieben wurde, ist eine freie Implementierung von Telnet und SSH für Windows. Es unterstützt daneben auch das SCP- und SFTP- Protokoll. GSS-API-Unterstützung muss durch einen Patch (z.B. [VTM_06] oder [CSS_06]) hinzugefügt werden. Putty setzt eine installierte MIT GSS-API voraus.
- WinSCP [Přik_06]: WinSCP ist ein Open Source SFTP- und SCP-Client mit einer Graphical User Interface (GUI) für Windows, der auf PuTTY aufbaut und dadurch SSH unterstützt. Seine Hauptfunktion besteht darin, Dateien sicher zwischen dem lokalem und einem entfernten Rechner zu kopieren. WinSCP unterstützt die MIT GSS-API.
- KTelnet [Nyst_06]: Telnet-Client mit Kerberos- und GSS-API-Unterstützung.
- FileZilla [File_06]: FTP/SFTP-Client und FTP-Server mit GSS-API-Unterstützung.

4.5.2 Webbasierte Anwendungen

Kerberisierte Webanwendungen sollen an dieser Stelle das SSO-Szenario abrunden. Der Fachbereich bietet geschützte Webinhalte und Mailserver an, die nach einer Authentifizierung mit dem UNIX-Passwort zugänglich sind. Dieses Kapitel stellt Web- und Mailserver vor, die kerberisiert werden können und beschreibt die passwortlose Authentifizierung gegenüber dem Web-Server, so dass ein Übertragen des Passworts über das Netz vermieden wird.

Folgende Authentifizierungsmechanismen können Clients gegenüber Web-Server verwenden:

- Die in [Frank_99] definierten challenge-response Authentifizierungsmechanismen für das HTTP-Protokoll:
    - Basic Authentication Scheme. Benutzername und Passwort werden unverschlüsselt über das Netz zum Web-Server gesendet werden. Diesen

Mechanismus benutzt der Fachbereich, jedoch wird das übertragene Passwort durch einen SSL-Kanal geschützt.

  - Digest Access Authentication Scheme. Es kommen Verschlüsselungstechniken (z.B. MD5) zum Einsatz, um die Klartextübertragung des Passwortes zu vermeiden. Der Nachteil ist, dass der Web-Server auf alle Klartextpasswörter Zugriff haben muss, um die Authentifizierungsantworten verifizieren zu können.

- Den in [Zhu_05] definierten Simple and Protected GSS-API Negotiation (SPNEGO)-Mechanismus. Ursprünglich wurde SPNEGO in [Baiz_98] spezifiziert. SPNEGO erlaubt GSS-API-Teilnehmern festzustellen, ob der jeweils andere einen bestimmten GSS-API-Mechanismus (z.B. Kerberos V5) unterstützt. Es bildet somit eine Schicht zwischen einer Anwendung und der GSS-API (siehe Abb. 27). Der SPNEGO HTTP Authentifizierung-Mechanismus wird nur zur Aushandlung der Authentifizierungsmethode eines Benutzers benutzt. Die HTTP-Header oder die in HTTP enthaltenen Informationen sind nicht geschützt und sollten mit SSL geschützt übertragen werden.
- Die GSS-API (siehe Kapitel 4.5.1).

Da während des Authentifizierungsprozesses keine Klartextpasswörter über das Netz übertragen werden sollen, werden in diesem Unterkapitel nur Anwendungen betrachtet, die den SPNEGO-Mechanismus oder die GSS-API direkt unterstützen. Für die GSS-API Unterstützung werden die MIT oder Heimdal GSS-API Bibliotheken gebraucht.

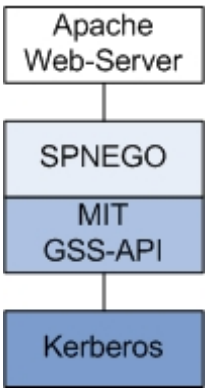

Abb. 27: SPNEGO als Zwischenschicht zwischen Anwendung und GSS-API

Folgende Module können die im Fachbereich eingesetzten Apache Web-Server (Version 1 und 2) [Apac_06] erweitern und dadurch kerberisieren:

- Kerberos Module For Apache Mod_auth_kerb Version 5.0 [Kour_06]. Das Modul unterstützt die Apache Versionen 1 und 2. Zur Authentifizierung können die Basic Authentication Scheme sowie der SPNEGO-Mechanismus eingesetzt werden. Durch Aktivierung von SPNEGO wird für die Authentifizierung eines Benutzers eine auf Tickets basierende Kommunikation erreicht, so dass von Benutzern keine Passworteingabe in einem Browser, der SPNEGO unterstützt, für den Zugriff auf geschützte Ressourcen gefordert wird.
- Apache Kerberos/SPNEGO module [Möll_06]. Das Modul unterstützt die Apache Versionen Apache 1.3 and 2.0 und fügt dem Server SPNEGO-Funktionalität hinzu.

Folgende Browser unterstützen den SPNEGO-Mechanismus oder die GSS-API:

- Microsoft Internet Explorer (IE) ab Version 5.5 [MSIE_06]. Der IE unterstützt das SPNEGO-Protokoll. Konfigurationshinweise sind in [Sura_02] beschrieben.
- Mozilla [Mozi_06]. Auf Mozilla basierende Browser können mit dem Negotiateauth [Nego_06] Plugin um SPNEGO erweitert werden.
    - Firefox [Fire_06] ab Version 1.5 (Linux + Windows). Der auf Mozilla basierende Firefox-Browser unterstützt von Haus aus die GSS-API.
- KDE Konqueror [Konq_06] ab KDE-Version 3.3.1. KDE unterstützt Kerberos HTTP Authentifizierung. Dafür kann mod_auth_kerb mit Konqueror benutzt werden. Der KDE Konqueror unterstützt auch SPNEGO, allerdings nur mit der Heimdal-Distribution.

Folgende Mail-Anwendungen unterstützen ebenfalls die GSS-API:

- Sendmail [Send_06]. Ein Message Transfer Agent, der als Mail Server genutzt wird. Seit Version 8.10 unterstützt Sendmail SMTP AUTH, wie in [Myer_99] definiert, das auf SASL basiert.
- Cyrus IMAP [Cyru_06]. Alternativ zu Sendmail kann auch Cyrus IMAP eingesetzt werden, das auf Cyrus SASL aufsetzt.
- Thunderbird [Thun_06] Email Client ab Version 1.5. Der auf Mozilla basierende Mailclient unterstützt die GSS-API. Dafür muss der Mail-Server SASL/GSS-API anbieten.
- Pine [Pine_06] Mail-Client ab Version 4 (Linux und Windows). Mail-Client mit MIT Kerberos-Unterstützung.

Abb. 28 zeigt ein Beispielszenario für die Integration des SPNEGO-Mechanismus:

(1) Ein Benutzer fordert während des Anmeldevorgangs ein TGT von einem KDC.
(2) Der KDC authentifiziert den Benutzer und sendet ihm ein TGT.
(3) Das TGT wird im Cache des Benutzers abgelegt.
(4) Der Benutzer möchte mit einem Browser auf die geschützte Web-Ressource `start.html` zugreifen. Der Browser sendet einen HTTP GET Request und erhält vom Server zunächst eine Fehlermeldung (401 Access Denied) und eine Negotiate-Antwort, die auf eine Kerberos-Authentifizierung hindeutet.
(5) Der Client fordert ein HTTP-Ticket für den Zugriff auf den Web-Server, verpackt es in ein SPNEGO-Token, sendet erneut einen HTTP GET Request und zusätzlich das SPNEGO-Token.
(6) Der Web-Server packt den Berechtigungsnachweis über die GSS-API aus.
(7) Der Web-Server prüft die Zugriffsberechtigung des Benutzers für die geschützte Ressource, gewährt dem Benutzer dementsprechend den Zugriff und sendet ihm die Ressource `start.html`.

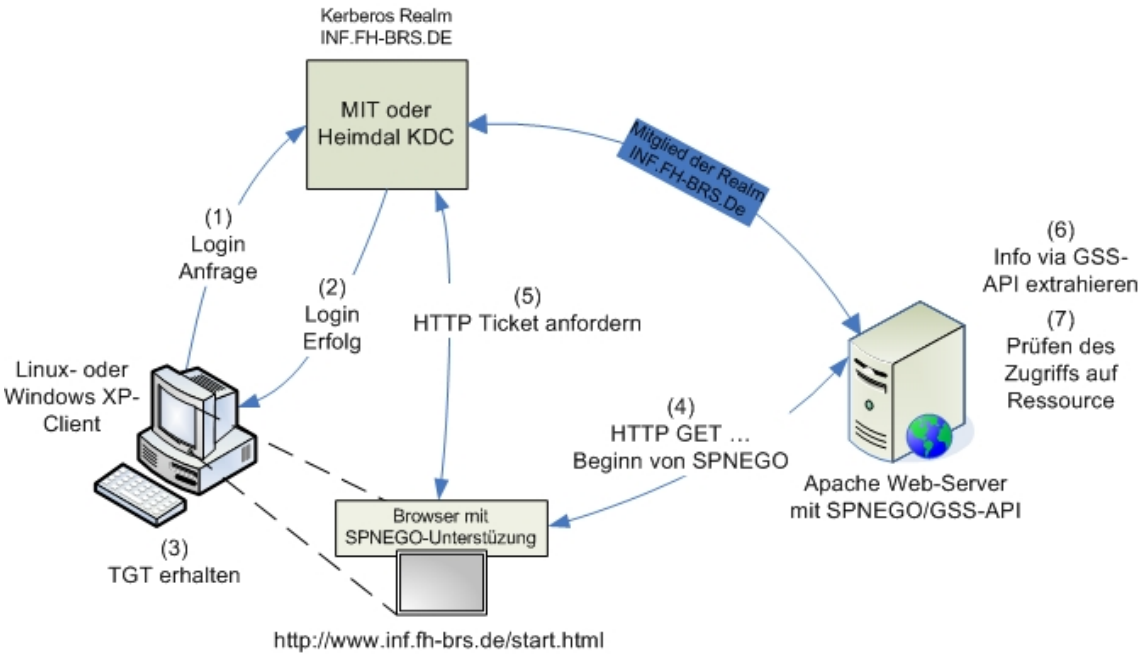

Abb. 28: Szenario für SPNEGO/GSS-API Anwendungen (in Anl. an [Sura_02])

4.5.3 Passwortänderungen

Das vorgestellte Konzept bietet vier Möglichkeiten zum Ändern des Kerberos-Passworts:

- Unter Linux:

  - Lokal über eine Shell oder eine GUI mit dem Werkzeug `krb5`, das ähnlich dem Network Identity Manager auch Tickets verwalten kann.
  - Über eine Shell kann der Benutzer von einem beliebigen Rechners aus per SSH sein Passwort zu ändern.
- Unter Windows:
  - Lokal über die GUI des Network Identity Managers oder über die Windows-Shell.
  - Über eine Shell kann der Benutzer von einem beliebigen Rechners aus per SSH sein Passwort zu ändern.

Über die Shell kommt unter Linux und Windows stets das speziell für Passwortänderungen entwickelte Werkzeug `kpasswd` zum Einsatz, das in den MIT- und Heimdal-Distributionen enthalten ist.

Das Windows NTLM Passwort muss wie bisher geändert werden. Dem Benutzer sollte die Möglichkeit geboten werden die beiden Passwörter zu synchronisieren (z.B. mit Hilfe eines Scripts), so dass der Benutzer für Linux und Windows dasselbe Passwort besitzt.

4.6 Zusammenfassung

Durch einen stufenweisen Migrationsablauf lässt sich im Fachbereich ein Proxy-based True SSO-Szenario aufbauen. Das hier vorgeschlagene Migrationskonzept besteht aus drei Phasen:

- Servermigrationsphase: Die Planung und Einführung von Kerberos und AFS.
- Umstellungsphase: Hier erfolgt ein Umschalten vom Alt- zum Neusystem. Dafür werden Daten ins AFS migriert und Anwendungen zur Nutzung von Kerberos und AFS modifiziert.
- Erweiterungsphase: Das Kerberisieren von Anwendungen rundet in diesem Schritt das Single Sign-On-Szenario ab.

In Bezug auf die in Kapitel 2.2.2 beschriebenen SSO Schemata und Kerberos, ergibt sich für den Fachbereich das in Abb. 29 dargestellte Proxy-based True SSO Szenario. Ein Benutzer authentifiziert sich gegenüber einen Kerberos KDC und erhält ein TGT. Das TGT wird in den Cache des Benutzers abgelegt und berechtigt den Benutzer zur

Beschaffung SP-spezifischer Service Tickets, die für den Zugriff auf die SP wie AFS oder SSH verwendet werden. Das heißt, mit einem TGT kann auf M verschiedene SP zugegriffen werden.

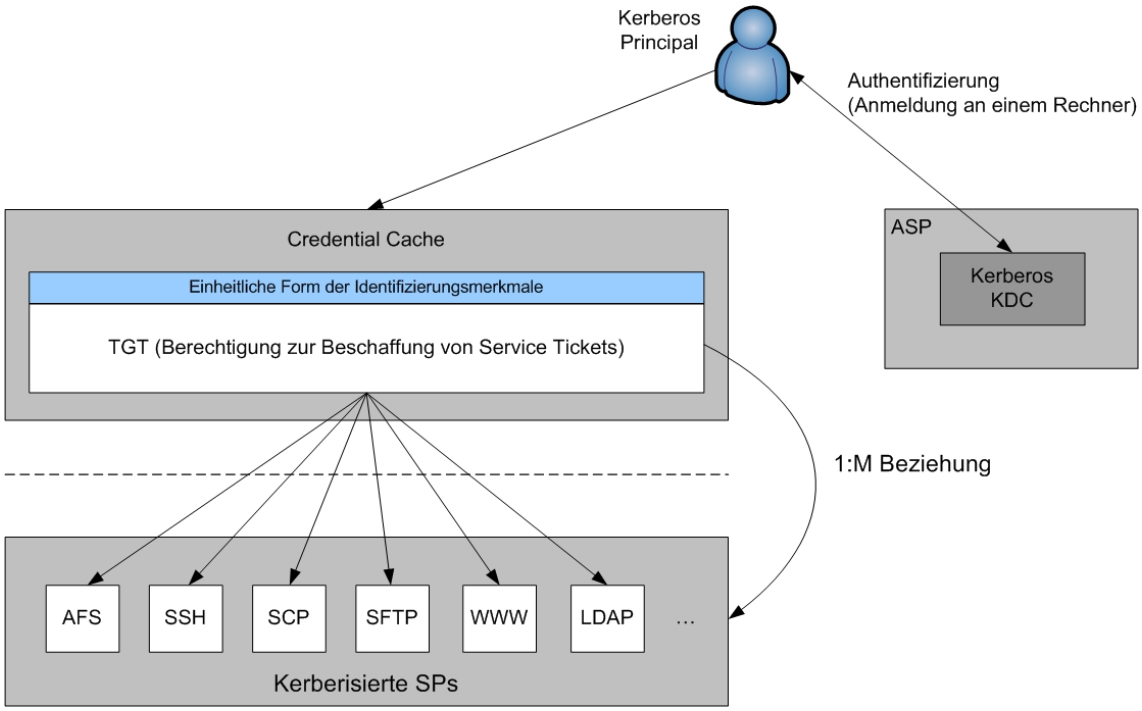

Abb. 29: Proxy-based True SSO für den Fachbereich durch Kerberos

5 Prototypische Implementierung

In diesem Kapitel erfolgt die prototypische Implementierung eines Kerberos-basierten True SSO-Szenarios auf Grundlage des Migrationskonzepts und seinen Anforderungen aus Kapitel 4. Dafür kommen einige kerberisierte Beispielanwendungen zum Einsatz, wobei die durchgeführten Schritte für deren Konfiguration beschrieben werden. Die Zielarchitektur entspricht Abb. 19 und Abb. 20 auf den Seiten 35 und 36, wobei das Gesamtbild des Prototyps durch Abb. 29 auf Seite 59 reflektiert wird. Zur Umsetzung der Schritte wurde insbesondere auf [Garm_03], [Klün_03] und den Dokumentation und Mailing Listen auf [MIT_06] und [OAFS_06] zurückgegriffen.

5.1 Testumgebung

Folgende Rechner wurden für die Testumgebung verwendet:

- `kerberos.krbrealm.krb`: Server-Rechner mit Linux-Betriebsystem zum Aufsetzen aller Server.
- `linuxclient.krbrealm.krb`: Client-Rechner, der als Linux-Client auftritt.
- `winclient.krbrealm.krb`: Client-Rechner, der als Windows-Client auftritt.

Die Rechner befinden sich innerhalb der Domäne `krbrealm.krb`, der Kerberos Realm `KRBREALM.KRB` und der AFS-Zelle `krbrealm.krb`. Die folgenden Kapitel 5.1.1 und 5.1.2 beschreiben die Hardware- und Netzkonfiguration und listen die installierte Software auf den einzelnen Rechnern auf.

5.1.1 Hard- und Software

Rechner / Ausst.	**kerberos (Server)**	**linuxclient (Client)**	**winclient (Client)**
Hardware			
Motherboard	Asus A7V266	Asus A7V266	Asus P4C800 Deluxe
CPU	AMD Athlon XP1700+	AMD Athlon XP1700+	Pentium 4 2.80 GHz
RAM	768 MB RAM	768 MB RAM	1024 MB RAM
Betriebssystem	Fedora Core 4 Kernel: 2.6.11-1.1369_FC4	Fedora Core (FC) 4 Kernel: 2.6.11-1.1369_FC4	Microsoft Windows XP Professional Service Pack 2 mit allen verfügbaren Updates (Stand: 12.02.2006)

Tabelle 5: Hardware und Betriebssysteme der Testsumgebung

Die folgende Tabelle beinhaltet aus Platzgründen nur die für das Zielszenario benutzten Anwendungen, die die notwendigen Werkzeuge für die Implementierung enthalten. Durch Einsatz des Betriebssystems Linux Fedora Core (FC) konnten für die Linux-Rechner durchgehend RPM Package Manager (RPM)-Pakete verwendet werden, die die Installation und Konfiguration der jeweiligen Anwendung vereinfachen und plattformspezifisch angepasst sind. Für das echte Szenario sollten jeweils die Quellpakete der Anwendungen verwendet werden, die plattformspezifisch angepasst werden können.

Rechner / Softw.	**kerberos (RPM) (Server)**	**linuxclient (RPM) (Client)**	**winclient (Client)**
Kerberos (MIT Distribution)	krb5-server-1.4.1-5 krb5-workstation-1.4.1-5 krb5-libs-1.4.1-5 krbafs-1.2.2-7 krbafs-devel-1.2.2-7	krb5-workstation-1.4.1-5 krb5-libs-1.4.1-5	MIT Kerberos for Windows 3.0 (KfW)
OpenAFS	openafs-1.4.0-1.1 openafs-kernel-1.4.0-2.6.11_1.1369_FC4_1 openafs-krb5-1.4.0-1.1 openafs-server-1.4.0-1.1 openafs-compat-1.4.0-1.1 openafs-kernel-source-1.4.0-1.1	openafs-1.4.0-1.1 openafs-kernel-1.4.0-2.6.11_1.1369_FC4_1 openafs-krb5-1.4.0-1.1 openafs-client-1.4.0-1.1 openafs-compat-1.4.0-1.1	OpenAFS for Windows-1.4.0
OpenLDAP	openldap-2.2.29-1.FC4 openldap-servers-2.2.29-1.FC4	openldap-2.2.29-1.FC4 openldap-clients-2.2.29-1.FC4	
Samba	samba-common-3.0.14a-2 samba-3.0.14a-2	samba-common-3.0.14a-2 samba-client-3.0.14a-2	
SASL	cyrus-sasl-2.1.20-5 cyrus-sasl-gssapi-2.1.20-5	cyrus-sasl-2.1.20-5 cyrus-sasl-gssapi-2.1.20-5	MIT GSS-API (in KfW enthalten)
SSH	openssh-4.0p1-3 openssh-server-4.0p1-3	openssh-4.0p1-3 openssh-clients-4.0p1-3	Putty 0.58 [VTM_06], WinSCP 3.8
PAM	pam-0.79-8 pam_krb5-2.1.7-3 (pam_krb5afs)	pam-0.79-8 pam_krb5-2.1.7-3	
NSS	nss_ldap-234-4	nss_ldap-234-4	

Tabelle 6: Software der Testumgebung

Zur Installation der RPM-Pakete wurde das Werkzeug Yellowdog Updater Modified (`yum`) verwendet, das aktuelle RPM-Pakete aus dem Internet bezieht, Abhängigkeiten der Anwendung auflöst und die erforderlichen Anwendungen installiert. Eine Ausnah-

me macht die OpenAFS-Software. Die erforderlichen Pakete mussten von [OAFS_06] bezogen werden.

5.1.2 Netzaufbau

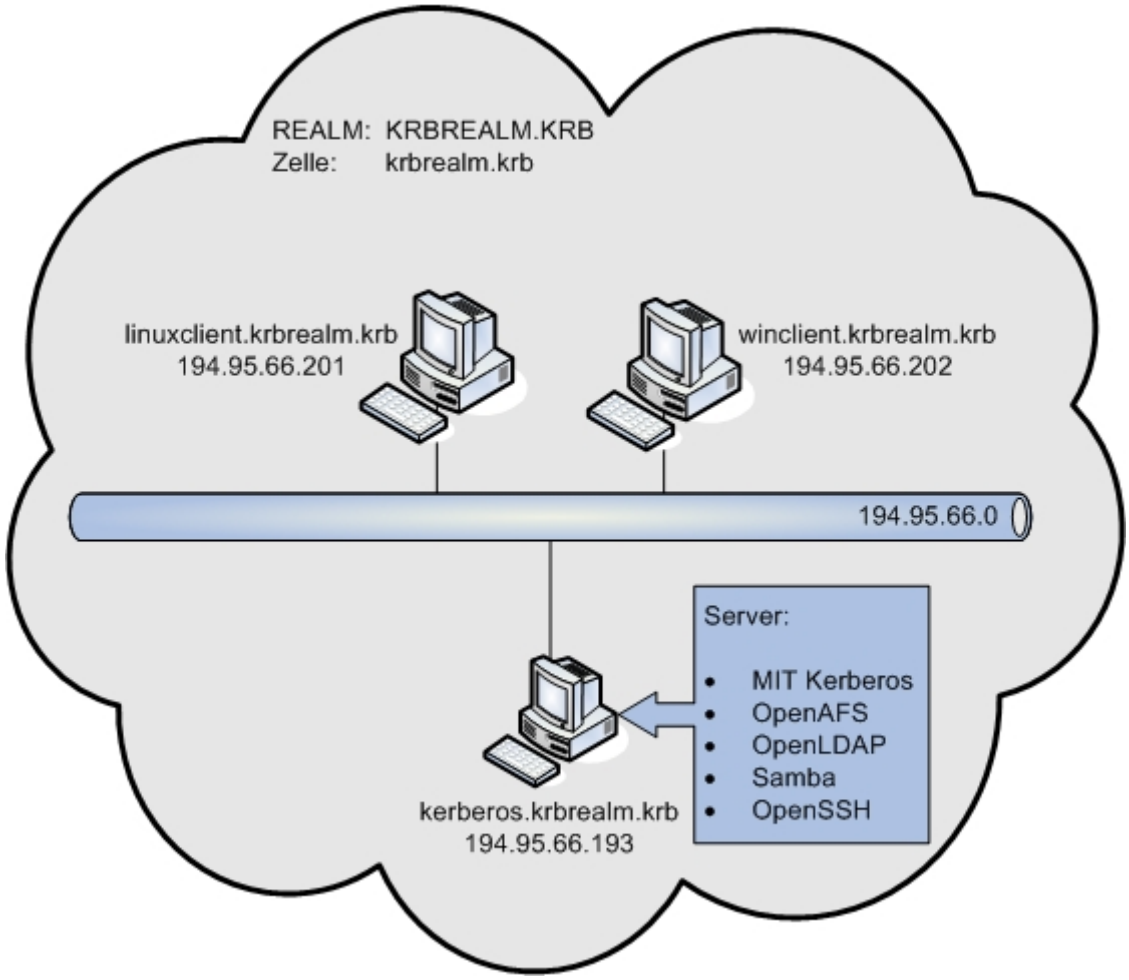

Abb. 30: Aufbau der Testumgebung

5.2 Vorbereitungen

Während der Installation von FC des Server-Rechners `kerberos` wurde exklusiv eine Partition der Größe 3,8 GB für AFS erstellt. Sie wurde mit dem Dateisystem ext3 formatiert und unter `/vicepa` gemountet. Auf den DNS-Dienst und jegliche Backup-Server wird verzichtet. Die Zeitsynchronisierung erfolgt manuell auf den lokalen Rechnern. Die in Tabelle 6 aufgelisteten Anwendungen wurden auf den jeweiligen Rechnern installiert. Im Folgenden wird die Konfiguration der Anwendungen näher beschrieben.

Zur Abbildung von IP-Adressen in Hostnamen wurde unter Linux die Datei

`/etc/hosts` und unter Windows ebenfalls die `hosts`-Datei verwendet. Der Rechnername in Fettschrifft und ein % (z.B. **`kerberos%`**) zeigen auf welchem Rechner ein Konfigurationsschritt durchgeführt wird. Ein Pfeil der Art ➔ deutet während eines Konfigurationsschritts auf eine Fortsetzung einer vorherigen Zeile hin. *`Kursive Schrift`* kennzeichnet Kommentare.

5.3 Konfiguration

5.3.1 MIT Master KDC

Der erste Schritt zur Konfiguration des Kerberos Realms besteht darin, Basisfunktionalitäten des KDCs einzurichten. Dafür werden auf dem Kerberos-Server `kerberos` die Dateien `kdc.conf` und `krb5.conf` (siehe Anhang A.1) konfiguriert. Die Datei `kdc.conf` konfiguriert die Einstellungen des Kerberos V5 KDC pro Realm, wie Datenbankangaben und unterstützte Schlüsseltypen. Hier muss besonders darauf geachtet werden, dass als unterstützter Schlüsseltyp `des-cbc-crc:normal` angegeben wird, der von Windows und seine Benutzer benötigt wird.

Die Datei `krb5.conf` ist die Kerberos Konfigurationsdatei und wird von den Kerberos V5 Bibliotheken und allen Clients benötigt. Sie enthält allgemeine Informationen, z.B. über den Kerberos Realm und den Standort der KDCs für bekannte Realms.

Initialisierung der Kerberos-Datenbank

Nachdem die Konfigurationsdateien angepasst wurden, kann die Kerberosdatenbank initialisiert werden. Dies erfolgt mit dem Befehl `kdb5_util`:

```
kerberos% kdb5_util create -r KRBREALM.KRB –s
Loading random data
Initializing database '/usr/local/var/krb5kdc/principal' for realm
➔ 'KRBREALM.KRB',
master key name 'K/M@KRBREALM.KRB'
You will be prompted for the database Master Password.
It is important that you NOT FORGET this password.
Enter KDC database master key: <Master Passwort>
Re-enter KDC database master key to verify: <Master Passwort>
```

Im vorigen Beispiel werden mit dem Kommando `kdb5_util` die Kerberos Realm `KRBREALM.KRB`, die Kerberos Datenbank-Dateien, einige Standard-Principals und eine so genannte Stash Datei (Schalter -s) erzeugt, die eine lokale Kopie des Master-

Passworts in verschlüsselter Form beinhaltet. Dieses Passwort wird in der Datei `.k5.KRBREALM.KRB` abgelegt und erlaubt dem KDC beim Starten der Kerberos-Dämonen sich gegenüber sich selbst zu authentifizieren.

Hinzufügen von Principals

Der Datenbank können nun mit dem Kerberos-Werkzeug `kadmin.local` Principals hinzugefügt werden. `kadmin.local` ist das lokale administrative Benutzerinterface zur Verwaltung der Kerberos-Datenbank. Für entfernte Zugriffe auf die Datenbank wird das Werkzeug `kadmin` benutzt.

Zunächst muss mindestens ein administrativer Principal und gemäß der Pfadangabe in der `kdc.conf` eine Zugriffskontrolllisten-Datei `kadm5.acl` für ihn erzeugt werden. `kadm5.acl` definiert die Zugriffsrechte von Principals auf die Kerberos-Datenbank. In diesem Beispiel haben alle Principals mit dem Zusatz `/admin` und der administrative Principal `dstojc2s/admin` durch die Angabe eines * vollen Zugriff auf die Datenbank. Der Inhalt der Datei `kadm5.acl`:

#Kerberos principal	*permissions*	*target_principal*
*/admin	*	
dstojc2s/admin@KRBREALM.KRB	*	

Den administrativen Principal `dstojc2s/admin` in die Datenbank einpflegen:

```
kerberos% kadmin.local

kadmin.local: addprinc dstojc2s/admin
WARNING: no policy specified for dstojc2s/admin@KRBREALM.KRB; defaulting to
➔ no policy
Enter password for principal "dstojc2s/admin@KRBREALM.KRB":<Passwort>
Re-enter password for principal "dstojc2s/admin@KRBREALM.KRB":<Passwort>
Principal "dstojc2s/admin@KRBREALM.KRB" created.
```

Die Kerberos-Dämonen `krb5kdc` (der KDC) und `kadmind` (zur Administration des KDC) können jetzt gestartet werden und fahren über entsprechende Scripte bei jedem Neustart des Rechners hoch:

```
kerberos% krb5kdc
kerberos% kadmind
```

Es soll nun ein TGT-Ticket für `dstojc2s/admin` geholt werden, um die Funktionalität des KDCs zu prüfen. Das Kommando `kinit` beschafft das TGT, `klist` zeigt Tickets aus dem Cache an:

```
kerberos% kinit dstojc2s/admin
Password for dstojc2s/admin@KRBREALM.KRB:<Passwort>

kerberos% klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: dstojc2s/admin@KRBREALM.KRB

Valid starting     Expires            Service principal
03/14/06 16:38:48  03/15/06 02:38:48  krbtgt/KRBREALM.KRB@KRBREALM.KRB
        renew until 03/15/06 16:38:48

Kerberos 4 ticket cache: /tmp/tkt0
klist: You have no tickets cached
```

Nun können normale Benutzer, Hosts und Dienste als Principals erzeugt werden. Zunächst wird der Test-Principal `sso` angelegt, dessen Schlüssel wegen der Windows-Kompatibilität mit einer DES-Verschlüsselung erzeugt wird:

```
kerberos% kadmin.local

kadmin.local:  addprinc -e des:normal sso
WARNING: no policy specified for sso@KRBREALM.KRB; defaulting to no policy
Enter password for principal "sso@KRBREALM.KRB":<Passwort>
Re-enter password for principal "sso@KRBREALM.KRB":<Passwort>
Principal "sso@KRBREALM.KRB" created.
```

Als nächstes können Host-Principals für die Rechner `kerberos`, `linuxclient` und `winclient` in die Datenbank eingepflegt werden, die Mitglieder der Realm KRBREALM.KRB sein sollen:

```
kerberos% kadmin.local

kadmin.local: addprinc -randkey host/kerberos.krbrealm.krb
kadmin.local: addprinc -randkey host/linuxclient.krbrealm.krb
kadmin.local: addprinc –e des:normal host/winclient.krbrealm.krb
```

Mit dem Kommando `ktadd` unter `kadmin.local` können Schlüssel aus der Kerberosdatenbank in eine so genannte Keytab-Datei extrahiert werden, die in der Regel in dem Verzeichnis `/etc` platziert wird. In einer Keytab-Datei sind Passwörter von Principals verschlüsselt abgelegt. Principals verwenden die Keytab, um sich gegenüber dem KDC zu authentifizieren. Es empfiehlt sich eine separate Keytab pro Dienst auf der Festplatte, auf dem der Dienst läuft, abzulegen und nur für diesen Dienst lesbar zu machen. Zunächst werden die Schlüssel der Rechner `kerberos` und `winclient` extrahiert und in die Datei `/etc/krb5.keytab` abgelegt:

```
kerberos% kadmin.local

kadmin.local: ktadd host/kerberos.krbrealm.krb
kadmin.local: ktadd –e des:normal host/winclient.krbrealm.krb
```

Um mit dem Werkzeug `kadmin` über das Netz die Kerberosdatenbank zu administrieren, muss eine Keytab namens `kadm5.keytab` für die administrativen Principals `kadmin/admin` und `kadmin/changepw`, die während der Initialisierung der Datenbank erzeugt wurden, generiert werden:

```
kerberos% kadmin.local

kadmin.local:  ktadd -k /var/kerberos/krb5kdc/kadm5.keytab kadmin/admin
➔ kadmin/changepw
```

Client-Konfiguration

Die oben beschriebene Minimalkonfiguration für die Realm reicht aus, um die Clients einzurichten. Dafür wurden folgende Schritte unternommen:

Auf dem Rechner `linuxclient`:

- Kopieren der `krb5.conf` von Rechner `kerberos` nach `/etc`.
- Eine Keytab für den Rechner `linuxclient` generieren:

```
linuxclient% kadmin

kadmin: ktadd host/linuxclient.krbrealm.krb
```

Auf dem Rechner `winclient`:

- Kopieren der `krb5.conf` nach `C:\Windows` und umbenennen in `krb5.ini`

Die Client- und Serverkonfiguration für die Kerberos Realm ist jetzt eingerichtet. Es wird mit der Konfiguration der AFS-Zelle fortgefahren.

5.3.2 OpenAFS

Das Dateisystem zur Datenablage für einen Benutzer soll in diesem Prototyp durch die AFS-Implementierung von OpenAFS gestellt werden. Dafür wird die AFS-Zelle `krbrealm.krb` eingerichtet, die die lokale Zelle für den Benutzer `sso` sein soll. Durch die Installation der OpenAFS RPM Pakete auf den jeweiligen Rechnern wurden alle nötigen Verzeichnisse erstellt. Zur Authentifizierung der Benutzer wird anstatt des OpenAFS KAS der im vorigen Kapitel konfigurierte MIT KDC benutzt (siehe Kapitel 4.3.3, Abb. 23).

Server-Konfiguration

Die auf dem Rechner `kerberos` exklusiv für AFS erstellte Partition wird durch die Datei `/etc/fstab` unter `/vicepa` gemountet. Der Zellename `krbrealm.krb` wird in den Textdateien `/usr/vice/etc/ThisCell` und `/usr/vice/etc/CellServDB` an erster Stelle eingetragen:

```
# Eintrag in ThisCell
krbrealm.krb

# Eintrag in CellServDB
>krbrealm.krb        # Zellen Name
194.95.66.193        # kerberos.krbrealm.krb (Datenbankserver)
[weitere Zellen …]
```

AFS beinhaltet einen eigenen Zeitmechanismus. Da hier die Rechnerzeit manuell eingestellt wurde – und im echten Szenario z.B. NTP benutzt werden soll –, muss dieser Mechanismus deaktiviert werden. Ebenso muss vorübergehend der so genannte Dynroot-Mechanismus deaktiviert werden, da es aufgrund dieses Mechanismus während der Konfiguration zu Problemen im Zugriff auf den AFS Namensraum kam. Diese Einstellung kann unter FC 4 in der Konfigurationsdatei `/etc/sysconfig/openafs` vorgenommen und nach der Konfiguration wieder rückgängig gemacht werden:

```
AFSD_ARGS="-nosettime -fakestat"
```

Nun wird ein Principal `afs` im MIT KDC erzeugt. Dadurch wird ein Kerberos V5 Schlüs-

sel und eine entsprechende Key Version Number für diesen Principal erzeugt, die für den nächsten Schritt wichtig ist. Der Schlüssel wird in eine Keytab-Datei `afs.keytab` extrahiert und durch das Kommando `asetkey` ausgelesen, in eine für den AFS-Server lesbare Form konvertiert und als Master-Key in der Datei `/usr/afs/etc/KeyFile` eingetragen. Die Datei `afs.keytab` kann danach gelöscht werden. Hier wird auch ein Principal in der Datenbank des KDC für den AFS-Administrator angelegt.

```
kerberos% kadmin.local

# AFS-Administrator erzeugen
kadmin.local: addprinc afs_admin
# AFS Principal erzeugen
kadmin.local: addprinc -randkey afs/krbrealm.krb
# Schlüssel in Datei afs.keytab extrahieren
kadmin.local: ktadd -k /tmp/afs.keytab afs/krbrealm.krb
# KVNO herausfinden
kadmin.local: getprinc afs/krbrealm.krb
...
Key: vno 3, DES cbc mode with CRC-32, no salt
Key: vno 3, Triple DES cbc mode with HMAC/sha1, no salt
Key: vno 3, DES with HMAC/sha1, no salt
...
kadmin.local: quit
# Schlüssel in KeyFile eintragen lassen
kerberos% asetkey add 3 /tmp/afs.keytab afs/krbrealm.krb
```

Der MIT Kerberos Dämon `krb524d` wird nun gestartet. Er sollte bei jedem Neustart des Rechners automatisch gestartet werden. `krb524d` übernimmt die Übersetzung von Kerberos V5- in AFS-Tickets.

```
kerberos% krb524d
```

Da noch keine AFS-Benutzer angelegt sind, muss der BOS-Serverprozess, der die AFS-Server bewacht, in einen Modus gestartet, der jedem Benutzer Zugriff auf AFS gewährt. Dafür wird der Schalter `–noauth` verwendet. Weiterhin werden mit dem Werkzeug `bos` in diesem Schritt der Zellenname definiert, die in Kapitel 4.3.3 beschriebenen Server-Prozesse initialisiert und der AFS-Administrator `afs_admin` der Textdatei `/usr/afs/etc/UserList` hinzugefügt. Dadurch kann `afs_admin` privilegierte `bos` und `vos` Kommandos ausführen. Mit dem Befehl `bos listhosts kerberos` wird überprüft, ob der Rechner `kerberos` als erster Datenbankserver der Zelle

registriert ist.

```
kerberos% bosserver –noauth &
kerberos% bos setcellname kerberos.krbrealm.krb krbrealm.krb –noauth
kerberos% bos create kerberos.krbrealm.krb buserver simple
➔ /usr/afs/bin/buserver -cell krbrealm.krb -noauth
kerberos% bos create kerberos.krbrealm.krb ptserver simple
➔ /usr/afs/bin/ptserver -cell krbrealm.krb -noauth
kerberos% bos create kerberos.krbrealm.krb vlserver simple
➔ /usr/afs/bin/vlserver -cell krbrealm.krb –noauth
kerberos% bos adduser kerberos.krbrealm.krb afs_admin -cell krbrealm.krb
➔ –noauth
kerberos% bos listhosts kerberos –noauth
Cell Name is krbrealm.krb
   Host 1 is kerberos.krbrealm.krb
```

Jetzt kommt das Werkzeug `pts` zum Einsatz, das die Protection Database verwaltet. `pts` legt vor allem AFS-Benutzer und -Gruppen an. Als erster Benutzer wird der AFS-Administrator angelegt:

```
kerberos% pts createuser -name afs_admin -cell krbrealm.krb -noauth
User afs_admin has id 1
kerberos% pts adduser afs_admin system:administrators -cell krbrealm.krb
➔ -noauth
kerberos% pts membership afs_admin -cell krbrealm.krb -noauth
Groups afs_admin (id: 1) is a member of:
  system:administrators
```

Im nächsten Schritt wird nach einem Neustart des BOS-Servers der Prozess `fs` gestartet. Er besteht aus den Prozessen File Server, Volume Server und Salvager. Danach wird mit dem Werkzeug `vos` das erste Volume namens `root.afs` (vgl. Abb. 6) erzeugt. Außerdem wird der Update Server (`upserver`) gestartet, der dafür sorgt, dass der Inhalt der Verzeichnisse auf dem AFS-Server anderen Rechnern dieser Zelle bekannt gemacht wird.

```
kerberos% bos restart kerberos.krbrealm.krb -all -cell krbrealm.krb -noauth
kerberos% bos create kerberos.krbrealm.krb fs fs /usr/afs/bin/fileserver
➔ /usr/afs/bin/volserver /usr/afs/bin/salvager -cell krbrealm.krb -noauth
kerberos% vos create kerberos.krbrealm.krb /vicepa root.afs -cell
➔ krbrealm.krb -noauth
```

```
Volume 536870912 created on partition /vicepa of kerberos.krbrealm.krb
kerberos% bos create kerberos.krbrealm.krb upserver simple
➔ "/usr/afs/bin/upserver -crypt /usr/afs/etc" -cell krbrealm.krb -noauth
```

Damit die AFS-Serverprozesse bei jedem Neustart des Server-Rechners automatisch hochfahren, wird unter FC 4 das graphische Service Configuration Tool verwendet und der Eintrag `openafs-server` markiert.

Nach einem Neustart des Rechners `kerberos` wird in einer Shell geprüft, ob der Benutzer `afs_admin` ein gültiges AFS-Ticket erhält. Dafür wird zunächst mit dem MIT Kommando `kinit` ein TGT-Ticket, dann mit dem Kommando `aklog` (alternativ `afs5log`) vom KDC ein AFS-Ticket geholt und mit `klist` der Ticketcache ausgelesen:

```
kerberos% kinit admin
Password for admin@KRBREALM.KRB:<Passwort>
kerberos% aklog
kerberos% klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: admin@KRBREALM.KRB

Valid starting     Expires            Service principal
02/13/06 21:33:52  02/14/06 07:33:52  krbtgt/KRBREALM.KRB@KRBREALM.KRB
        renew until 02/14/06 21:33:52
02/13/06 21:33:59  02/14/06 07:33:52  afs/krbrealm.krb@KRBREALM.KRB
        renew until 02/14/06 21:33:52

Kerberos 4 ticket cache: /tmp/tkt0
klist: You have no tickets cached
```

Mit dem AFS-Ticket können nun die AFS-Werkzeuge ohne den Schalter `-noauth` ausgeführt werden, da der AFS-Administrator im AFS eingepflegt ist und sich gegenüber den KDC authentifizieren kann. Dass die AFS-Server alle korrekt gestartet wurden, wird mit folgendem Kommando geprüft:

```
kerberos% bos status kerberos
Instance buserver, (type is simple) currently running normally.
Instance ptserver, (type is simple) currently running normally.
Instance vlserver, (type is simple) currently running normally.
Instance fs, (type is fs) currently running normally.
```

```
    Auxiliary status is: file server running.
Instance upserver, (type is simple) currently running normally.
```

Der nächste Schritt konfiguriert den Top Level des AFS-Namensraumes. Dafür werden Zugriffskontrolllisten auf das Verzeichnis `/afs` gesetzt, das Volume `root.cell` (vgl. Abb. 6, S. 16) erzeugt und Mountpunkte definiert. Außerdem wird ein symbolischer Link gesetzt, der die Länge des AFS-Pfadnamens verkürzt.

```
kerberos% fs setacl /afs system:anyuser rl
kerberos% vos create kerberos.krbrealm.krb /vicepa root.cell
Volume 536870915 created on partition /vicepa of kerberos.krbrealm.krb
kerberos% fs mkmount /afs/krbrealm.krb root.cell
kerberos% fs setacl /afs/krbrealm.krb system:anyuser rl
kerberos% cd /afs/
kerberos% ln -s krbrealm.krb krbrealm
kerberos% fs mkmount /afs/.krbrealm.krb root.cell –rw
```

Als nächstes werden die Volumes `root.afs` und `root.cell` mit dem Befehl `vos addsite` repliziert. Die Replikation eines Volumes bewirkt, dass zu dem Volume ein Klon existiert, auf das zurückgegriffen werden kann, wenn das originale Volume nicht erreichbar ist. Der Klon kann auf einem anderen Server liegen. Mit dem Befehl `vos release` werden die replizierten Volumes freigegeben und schließlich mit dem Befehl `fs checkvolumes` dem Cache Manager bekannt gemacht.

```
kerberos% vos addsite kerberos.krbrealm.krb /vicepa root.afs
Added replication site kerberos.krbrealm.krb /vicepa for volume root.afs
kerberos% vos addsite kerberos.krbrealm.krb /vicepa root.cell
Added replication site kerberos.krbrealm.krb /vicepa for volume root.cell
kerberos% vos release root.afs
Released volume root.afs successfully
kerberos% vos release root.cell
Released volume root.cell successfully
kerberos% fs checkvolumes
All volumeID/name mappings checked.
```

Die AFS-Zelle ist jetzt betriebsbereit, so dass normale Benutzer hinzugefügt werden können. Dafür wird mit `pts` zunächst eine Gruppe `testgroup` mit der GID 501 angelegt, der Administrator `afs_admin` der Gruppe hinzugefügt und der Eigentümer der Gruppe definiert:

```
kerberos% pts creategroup -name testgroup -id -501
group testgroup has id -501
kerberos% pts adduser -user afs_admin -group testgroup
kerberos% pts chown testgroup testgroup
```

Die Volumes für die Homeverzeichnisse sollen in der Zelle unter `/afs/krbrealm.krb/home` untergebracht werden. Dafür kann das UNIX-Kommando `mkdir` benutzt werden. Der im KDC erzeugte Test-Principal `sso` wird mit der UID 501 angelegt und der Gruppe `testgroup` zugewiesen.

```
kerberos% mkdir /afs/.krbrealm.krb/home
kerberos% vos release root.cell
Released volume root.cell successfully

kerberos% pts createuser -name sso -id 501
User sso has id 501
kerberos% pts adduser -user sso -group testgroup
```

Für den Benutzer `sso` werden nachfolgend ein Volume für das Homeverzeichnis erstellt, die Zugriffsrechte definiert und eine Quota von 2 GB festgelegt:

```
kerberos% vos create -name user.sso -server kerberos.krbrealm.krb –partition
➔ /vicepa
Volume 536870918 created on partition /vicepa of kerberos.krbrealm.krb

# Quota
kerberos% vos setfields -id user.sso -maxquota 2000000
kerberos% fs mkmount /afs/.krbrealm.krb/home/sso user.sso -rw
kerberos% vos release root.cell
Released volume root.cell successfully

# Zugriffsrechte
kerberos% fs setacl /afs/krbrealm.krb/home/sso sso rlidwka
kerberos% chown 501.501 /afs/krbrealm.krb/home/sso
kerberos% chmod 700 /afs/krbrealm.krb/home/sso
```

Die grundlegende Konfiguration der AFS-Server ist hiermit abgeschlossen. Es wird mit der Client-Konfiguration fortgefahren.

Client-Konfiguration

Die Client-Cachegröße wird in der Textdatei `/usr/vice/etc/cacheinfo` auf 100

MB festgesetzt. Damit die Clients `linuxclient` und `winclient` der Zelle `krbrealm.krb` zugewiesen werden, werden die Dateien `ThisCell` und `CellServDB` in die dafür vorgesehenen Verzeichnisse kopiert:

- `linuxclient`
  - `ThisCell` und `CellServDB` nach `/usr/afs/etc` kopieren.
  - Mit dem Service Configuration Tool den Dienst `openafs-clients` für das automatische starten aktivieren.

- `winclient`
  - Angaben über die Zelle erfolgen über die Installation oder die GUI des OpenAFS-Clients.
  - Die Datei `CellServDB` wird in `C:\Program Files\OpenAFS\Client` abgelegt.

Die Einstellungen in der Konfigurationsdatei `/etc/sysconfig/openafs` auf den Rechnern `kerberos` und `linuxclient` werden wie folgt geändert:

```
AFSD_ARGS="-dynroot -nosettime -fakestat"
```

Ein Anmelden an einem Rechner für den Benutzer `sso` ist noch nicht möglich, da an keiner Stelle Benutzerinformationen vorhanden sind. Nachfolgend wird der Benutzer `sso` im Verzeichnis von OpenLDAP eingepflegt, so dass sich die zur Anmeldung benötigten Benutzerinformationen wie UID, GID oder Homeverzeichnis aus dem Verzeichnisdienst auslesen lassen.

5.3.3 OpenLDAP

In diesem Kapitel wird die Konfiguration des Verzeichnisdienstes OpenLDAP beschrieben. Er dient der zentralen Ablage benutzerrelevanter Informationen, die von verschiedenen Anwendungen wie dem Anmeldedienst oder Samba ausgelesen werden.

Server-Konfiguration

Zunächst wird eine SASL-Konfigurationsdatei für den OpenLDAP-Dienst im Verzeichnis `/usr/lib/sasl2` namens `slapd.conf` erstellt, der den `slapd`-Dienst anweist die GSS-API zu benutzen. Die Datei `slapd.conf` weist folgende Zeile auf:

```
# Die Datei SASL-Konfigurationsdatei slapd.conf
mech_list: gssapi
```

Als nächstes wird ein Principal `manager` für den LDAP-Administrator und ein Principal `ldap` für den LDAP-Dienst erzeugt. Zudem wird der Schlüssel von `ldap` in die Keytab-Datei `krb5.keytab` extrahiert und für den Benutzer `ldap` lesbar gemacht, damit sich der Verzeichnisdienst gegenüber dem KDC authentifizieren kann.

```
kerberos% kadmin.local

kadmin.local: addprinc manager
kadmin.local: addprinc -randkey ldap/kerberos.krbrealm.krb
kadmin.local: ktadd ldap/kerberos.krbrealm.krb
kerberos% chgrp ldap /etc/krb5.keytab && chmod 640 /etc/krb5.keytab
```

Die Konfiguration des LDAP-Servers und Angaben über einzubeziehende Schemas erfolgen in der Datei `/etc/openldap/slapd.conf` (siehe Anhang A.2). Für den administrativen Zugriff auf den LDAP-Server werden zwei Benutzer verwendet:

- `dn: rootdn: cn=manager,dc=krbrealm,dc=krb:` Der Benutzer `manager` hat vollen Zugriff auf den Verzeichnisdienst. Dafür authentifiziert er sich gegenüber dem KDC und beschafft sich über die GSS-API ein LDAP-Ticket mit dem er OpenLDAP administrieren kann. Dadurch wird vermieden, dass das Passwort des LDAP-Administrators in der `slapd.conf` angegeben werden muss.
- `dn: uid=sambaroot,ou=People,dc=krbrealm,dc=krb:` Der Benutzer `sambaroot` hat beschränkten Zugriff auf den Verzeichnisdienst. Er hat nur vollen Zugriff auf Verzeichniszweige, die Windows-Informationen enthalten.

Ein weiterer Samba-Benutzer im Vezeichnis ist `root`, der benötigt wird um den Rechner `winclient` an der Windows-Domäne anzumelden.

Client-Konfiguration

Die LDAP-Clients verwenden die Datei `ldap.conf`, um erforderliche Informationen wie Servername oder IP-Addresse des LDAP-Servers auszulesen. Die LDAP-Client-Konfigurationsdatei `ldap.conf` des Rechners `linuxclient` wurde wie in Anhang A.2 konfiguriert.

Der nächste Schritt ist die Konfiguration des NSS-Dienstes, der dem Anmeldedienst und anderen Anwendungen angibt, dass Benutzerinformation vom LDAP-Server abgefragt werden sollen. Dafür wird die Datei `/etc/nsswitch.conf` konfiguriert. Zunächst soll geprüft werden, ob die Benutzerinformation lokal vorhanden sind (z.B. in der `passwd`-Datei). Falls nicht, wird der LDAP-Server befragt. Folgende Einstellungen wurden dafür in der `/etc/nsswitch.conf` unternommen:

```
[…]
passwd:   files ldap
shadow:   files ldap
group:    files ldap
[…]
```

Als nächstes wird geprüft, ob der Benutzer `manager` ein LDAP-Ticket erhält und die GSS-API funktioniert, indem er eine Suchoperation auf den Verzeichnisdienst durchführt. Die in Anhang A.2 festgelegten regulären Ausdrücke sorgen dafür, dass der Principal `manager@KRBREALM.KRB` in eine entsprechende LDAP-Syntax abgebildet wird, so dass er von OpenLDAP als zugriffsberechtigter Benutzer erkannt wird.

```
linuxclient% kinit manager
Password for manager@KRBREALM.KRB:<Passwort>
linuxclient% ldapsearch
SASL/GSSAPI authentication started
SASL username: manager@KRBREALM.KRB
...[leer, da noch keine Verzeichniseinträge vorhanden]
linuxclient% klist
Ticket cache: FILE:/tmp/krb5cc_0
Default principal: manager@KRBREALM.KRB

Valid starting     Expires            Service principal
02/13/06 21:33:52  02/14/06 07:33:52 krbtgt/KRBREALM.KRB@KRBREALM.KRB
        renew until 02/14/06 21:33:52
02/13/06 21:33:59  02/14/06 07:33:52 ldap/kerberos.krbrealm.krb@KRBREALM.KRB
        renew until 02/14/06 21:33:52
```

Wenn das LDAP-Ticket vorhanden ist, verlässt sich OpenLDAP auf die Authentifizierung von Kerberos und gewährt dem Principal `manager` den Zugriff auf das Verzeichnis. Dadurch kann die Verzeichnisdatenbank ohne Passworteingabe mit Daten populiert werden. Die dafür benutzten LDIF-Dateien sind in Anhang A.3 enthalten. Das OpenLDAP-Werkzeug `ldapadd` übernimmt das Anreichern der Datenbank:

```
kerberos% ldapadd –f rootdn.ldif
kerberos% ldapadd –f ounits.ldif
kerberos% ldapadd –f group.ldif
kerberos% ldapadd –f hosts.ldif
kerberos% ldapadd –f netgroup.ldif
kerberos% ldapadd –f passwd.ldif
kerberos% ldapadd –f sso.ldif
kerberos% ldapadd –f sambaDomainName.ldif
```

5.3.4 Samba

Der Samba-Server auf dem Rechner `kerberos` übernimmt die Rolle eines Windows PDC Domänen Controllers. Dafür wird die Windows-Domäne `krbrealm.krb` eingerichtet, in die der Rechner `winclient` beitreten kann. Samba verwaltet Benutzernamen, Passwörter, Zugriffsrechte, UID, GID und Freigaben für die Windows-Umgebung, die pro Benutzer im OpenLDAP-Server gelagert werden. Weitere Benutzerrelevante Informationen wie der Pfad zum Roaming Profile werden ebenfalls im OpenLDAP-Server abgelegt und von dort auch während der Anmeldung des Windows-Benutzers `sso` abgefragt.

Wegen den in Kapitel 4.4.4 aufgeführten Gründen, muss für den Benutzer `sso` ein separates Windows-Passwort angelegt werden, das im Verzeichnis in verschlüsselter Form aufbewahrt wird. Das Passwort ist dasselbe wie das des Principals `sso`, so dass der Benutzer `sso` sich während des Anmeldevorgangs an der Windows-Domäne anmeldet und sich durch entsprechende Software gegenüber dem Kerberos KDC authentifiziert (siehe Kapitel 5.4.2).

Die Konfiguration des Samba-Servers erfolgt über die Datei `/etc/samba/smb.conf`. Sie ist in Anhang A.4 enthalten. In der `smb.conf` ist eine Freigabe `[afs]` für den Zugriff auf im AFS abgelegte Daten eingerichtet. Home- sowie Profildaten aus dem AFS können somit erreicht werden, indem für den Benutzer `sso` im Verzeichnis die entsprechenden Einträge auf diese Freigabe verweisen (vgl. Anhang A.3, Datei `sso.ldif`).

5.3.5 PAM

In diesem Schritt wird das PAM-Modul `pam_krb5afs.so` für den Rechner `linuxclient` in die PAM-Konfiguration integriert, so dass während des Anmeldevorgangs

des Linux-Benutzers `sso` ein Kerberos- und ein AFS-Ticket vom KDC geholt werden. Dafür werden unter FC 4 in die Konfigurationsdatei `/etc/pam.d/system-auth` folgende Einträge den Anforderungen entsprechend hinzugefügt:

```
auth     sufficient  /lib/security/pam_krb5afs.so  try_first_pass tokens
account  sufficient  /lib/security/pam_krb5afs.so  try_first_pass
password sufficient  /lib/security/pam_krb5afs.so  try_first_pass
session  optional    /lib/security/pam_krb5afs.so  try_first_pass
```

Der Konfigurationsparameter `try_first_pass` weist das Modul an, das Passwort eines vorher aufgerufenen Moduls zu verwenden. Falls dies fehlschlägt, wird ein erneuter Versuch der Authentifizierung unternommen, indem der Benutzer zur Eingabe seines Passworts aufgefordert wird. Der Parameter `tokens` wird von Anwendungen wie Samba benötigt. `tokens` sorgt dafür, dass während des Anmeldevorgangs ein AFS-Ticket beschafft wird.

5.4 Anmeldevorgang

5.4.1 Linux

Der Testbenutzer `sso` kann sich nun über die Linux-GUI oder einer Shell gegenüber den MIT KDC auf dem Rechner `kerberos` authentifizieren. Der Vorgang läuft wie in Abb. 24, S. 48 ab.

Dies bestätigt nach dem Anmeldevorgang der Aufruf von `klist`, das anzeigt dass ein Kerberos- und ein AFS-Ticket erhalten wurden. Dass der Benutzer im AFS in seinem Homeverzeichnis gelandet ist, wird mit dem Kommando pwd geprüft.

```
linuxclient% klist
Ticket cache: FILE:/tmp/krb5cc_501_SyRdNB
Default principal: sso@KRBREALM.KRB

Valid starting     Expires            Service principal
03/17/06 15:18:22  03/18/06 01:18:22  krbtgt/KRBREALM.KRB@KRBREALM.KRB
        renew until 03/18/06 01:18:22

Kerberos 4 ticket cache: /tmp/tkt501_dXcpZz
Principal: sso@KRBREALM.KRB
```

```
  Issued              Expires             Principal
03/17/06 15:18:22  03/18/06 01:18:22  krbtgt.KRBREALM.KRB@KRBREALM.KRB
linuxclient% pwd
/afs/krbrealm.krb/home/sso
```

5.4.2 Windows

Der Anmeldevorgang auf `winclient` läuft wie in Abb. 25, S. 49 ab. Nachdem der Rechner hochgefahren ist, drückt der Benutzer die Tastenkombination `Ctrl+Alt+Entf` und kann die Windows-Domäne `KRBREALM.KRB` anwählen und sich mit dem in Kerberos und Samba eingerichteten Passwort an der Domäne anmelden.

Abb. 31: Der OpenAFS-Client

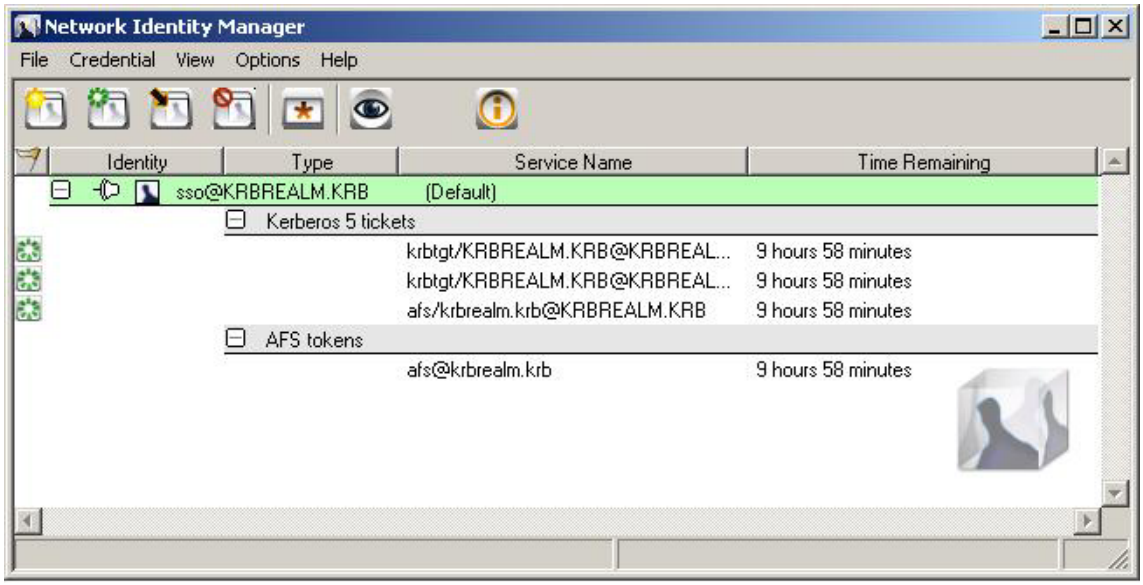

Abb. 32: Der Kerberos-Ticketverwalter Network Identity Manager

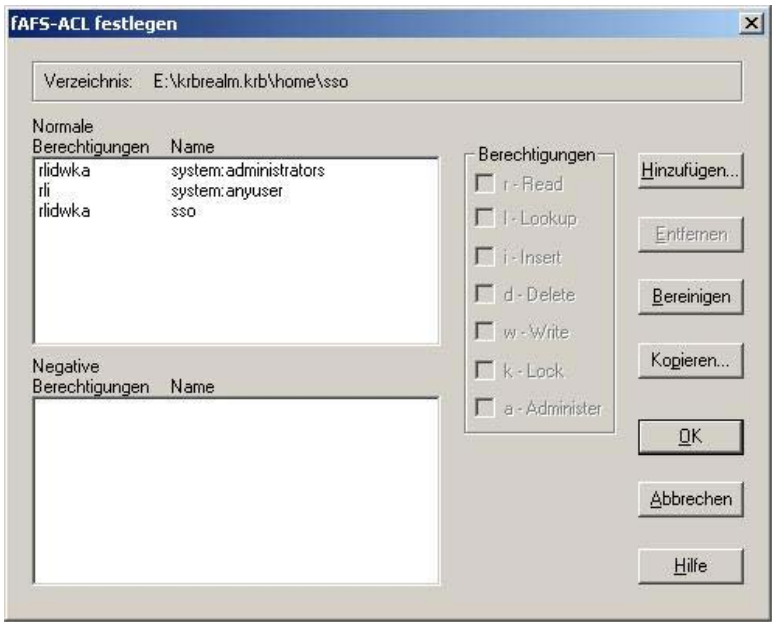

Abb. 33: GUI zur Definition von AFS Zugriffssteuerlisten

Über ein graphisches Benutzerinterface kann der Benutzer Zugriffskontrolllisten für sein Homeverzeichnis `/afs/krbrealm.krb/home/sso` definieren (siehe Abb. 33) und festlegen, wer Zugriff auf welche Verzeichnisse haben soll.

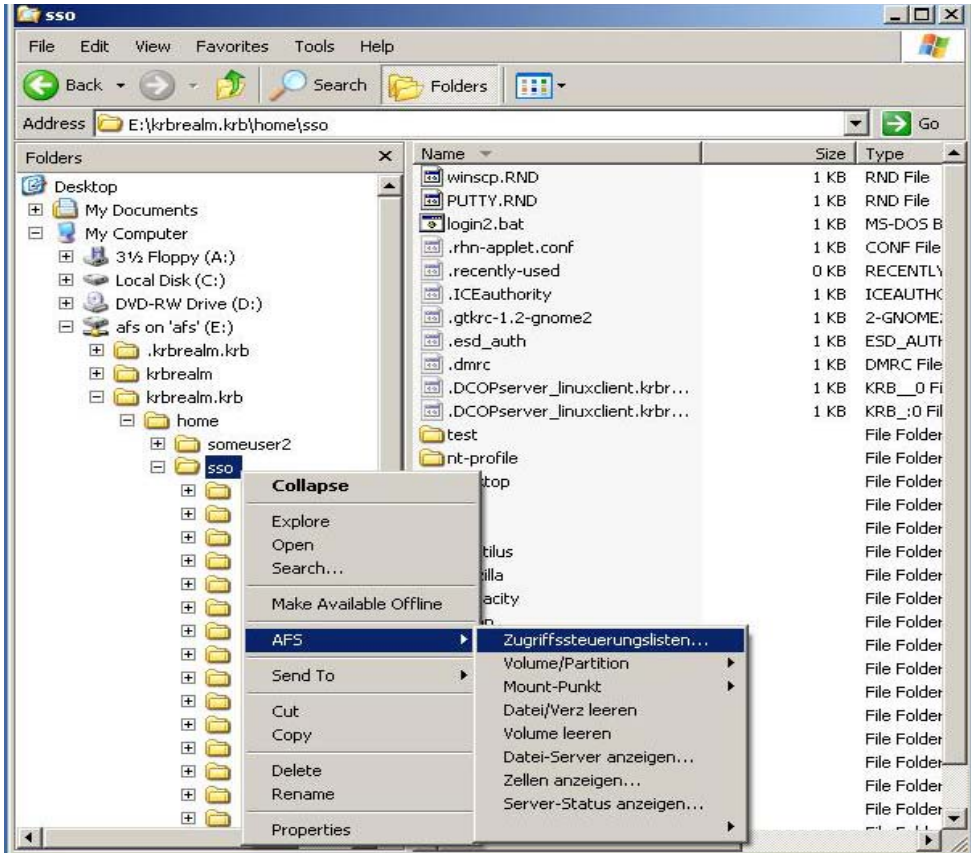

Abb. 34: Explorer-Menü von OpenAFS und eingehängtes AFS-Netzlaufwerk `E:\`

Die zweite Zugriffskontrollliste in der Abbildung definiert, dass nicht authentifizierte Benutzer (`system:anyuser`) die Verzeichnis- und Dateinamen lesen (l), den Inhalt von Dateien lesen (r) und Dateien in das Verzeichnis einfügen können (i). Diese ACL musste definiert werden, um das Roaming Profile-Verzeichnis `/afs/krbrealm.krb/home/sso/nt-profile` über den Samba-Server zu laden.

5.5 Kerberisierte Anwendungen

OpenLDAP wurde in Kapitel 5.3.3 kerberisiert. In diesem Kapitel wird beschrieben wie Anwendungen, die oft von Benutzern verwendet werden, kerberisiert wurden. Dazu gehören die SSH-basierten Anwendungen `ssh`, `scp` und `sftp`. Ausserdem wird auf einer theoretischen Basis erläutert, wie der Apache Server kerberisiert werden kann.

5.5.1 SSH-basierte Anwendungen

Zunächst wird der SSH-Server konfiguriert.

SSH-Server-Konfiguration

Die Konfigurationsdatei `/etc/ssh/sshd_config` des SSH-Servers auf dem Rechner `kerberos` wird um folgende Zeilen ergänzt:

```
# Passwort des Benutzers durch KDC verifizieren.
KerberosAuthentication yes
# Tickets nach Abmelden löschen
KerberosTicketCleanup yes

# GSSAPI-Konfiguration
GSSAPIAuthentication yes
GSSAPICleanupCredentials yes
```

Client-Konfiguration

Der SSH-Client auf dem Linuxrechner `linuxclient` erfolgt in der Konfigurationsdatei `/etc/ssh/ssh_config`.

```
# GSSAPI-Unterstützung aktivieren
GSSAPIAuthentication yes
# Weiterleitung des Kerberos-Tickets zum Zielrechner erlauben
GSSAPIDelegateCredentials yes
```

Durch die hier verwendete OpenSSH Version 4 ergibt sich ein Problem. Während die Weiterleitung des Kerberos-Tickets von OpenSSH unterstützt wird, ist dies mit dem AFS-Ticket nicht möglich, da OpenSSH ab Version 3 das AFS-Ticket erst nach der Authentifizierung zum Zielrechner weiterleitet. Dadurch ist es nicht möglich den Benutzer in sein Homeverzeichnis umzuleiten, da er durch das fehlende AFS-Ticket nicht die Zugriffsberechtigung für sein Homeverzeichnis hat. Stattdessen landet der Benutzer im Wurzelverzeichnis `/` des Zielrechners und muss sich mittels der Werkzeuge `aklog` oder `afs5log` (ohne Passworteingabe) ein AFS-Ticket beschaffen und mit den Kommando `cd ~` in sein Homeverzeichnis wechseln. Ein möglicher Ansatz zur Lösung dieses Problems besteht darin, PAM zur Generierung eines AFS-Ticket aus dem weitergeleiteten Kerberos-Ticket zu benutzen. Weitere Lösungsansätze für dieses Problem sind z.B. in [Coge_05] beschrieben.

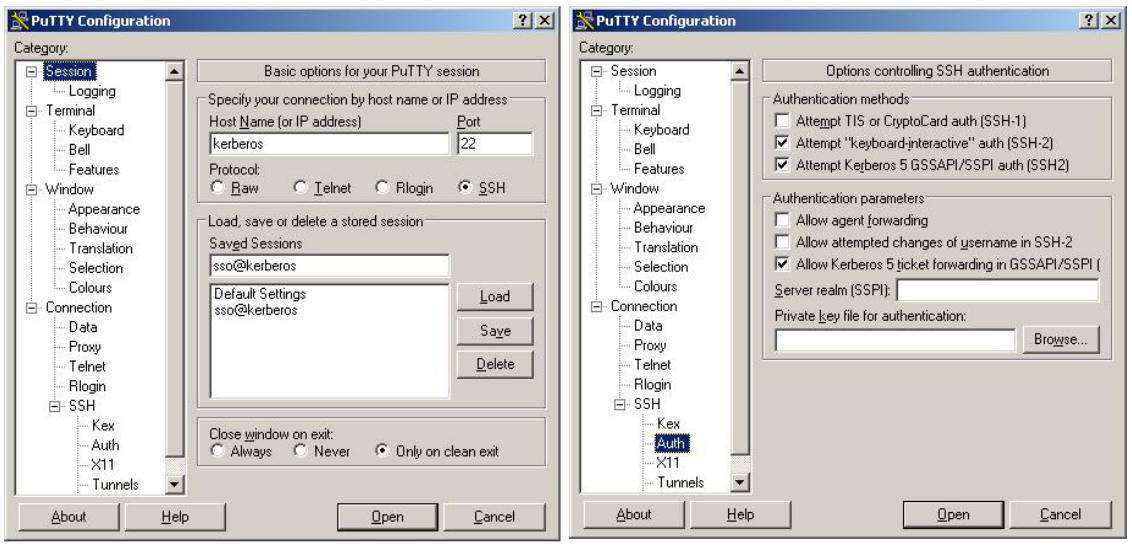

Abb. 35: Konfiguration von Putty

Auf den Windowsrechner `winclient` wurden die Anwendungen Putty und WinSCP wie in den Abbildungen Abb. 35 und Abb. 36 konfiguriert. Die Einstellungen aktivieren die GSS-API-Unterstützung für die jeweilige Anwendung. Putty und WinSCP setzen eine installierte MIT GSS-API voraus. Die Werkzeuge `scp` und `sftp`, die in der Putty-Distribution enthalten sind, können über die Windows-Shell aufgerufen und ohne Passworteingabe verwendet werden.

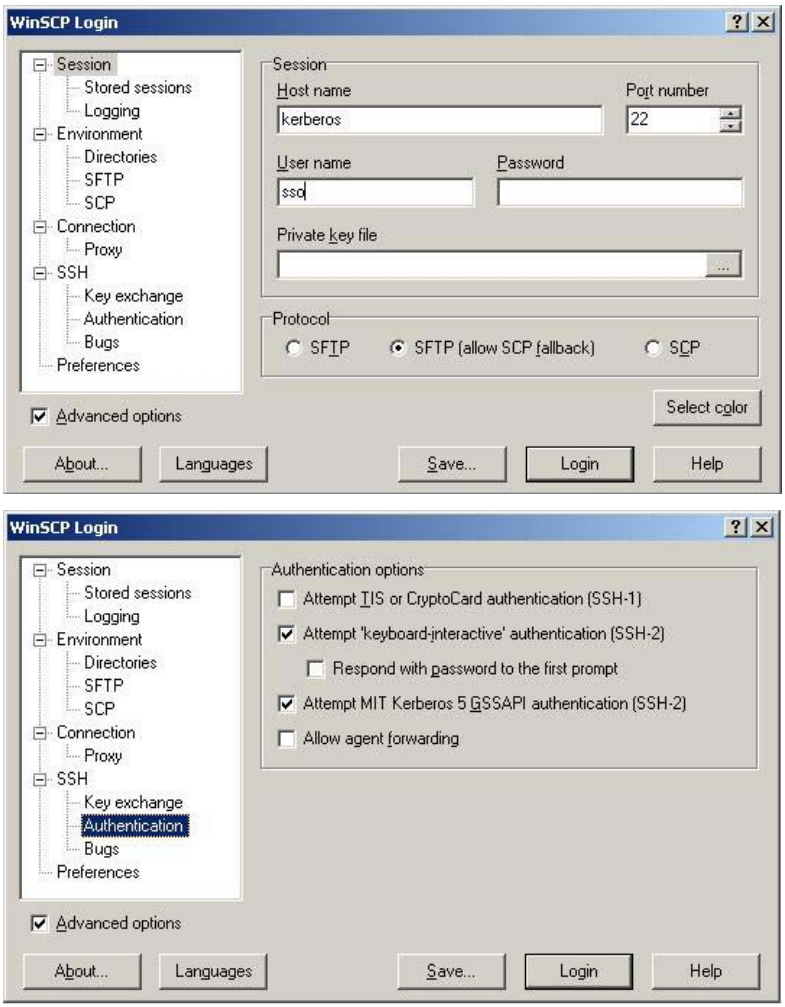

Abb. 36: Konfiguration von WinSCP

5.5.2 Apache Web-Server

Die in diesem Kapitel beschriebene Apache-Konfiguration wurde aus zeitlichen Gründen nicht praktisch umgesetzt. Sie zeigt jedoch die wesentlichen Einstellungen zur Kerberisierung des Web-Servers. Die Ausgangssituation ist die folgende: Ein Benutzer möchte auf geschützte Web-Ressourcen auf dem WWW-Server `www_server` zugreifen, für die eine Authentifizierung verlangt wird. Das Ziel ist die Authentifizierung des Benutzers gegenüber einen KDC mit Hilfe des SPNEGO/GSS-API-Mechanismus. Es wird das Apache-Modul von [Kour_06] verwendet. Zur Umsetzung müssen folgende Schritte durchgeführt werden:

(1) Installation des Moduls.

```
www_server% ./configure –with-krb5=<Kerberos-Verzeichnis> --with-krb4=no
➔ --with-apache=<Apache-Verzeichnis>
www_server% make && make install
```

(2) Service Principal für den Web-Server erzeugen und Schlüssel in eine Keytab apache.keytab extrahieren (Keytab sollte nur von apache lesbar sein) .

```
www_server% kadmin
# Principal mit zufälligem Schlüssel erstellen
kadmin: addprinc -randkey HTTP/www_server.krbrealm.krb
# Schlüssel extrahieren und in Apache-Konfigurations-Verzeichnis ablegen
kadmin: ktadd -k /etc/apache/apache.keytab HTTP/www_server.krbrealm.krb
```

(3) Die Konfiguration des Apache-Moduls `mod_auth_kerb` kann in der Apache-Konfigurationsdatei `http.conf` mittels Directory-Direktiven vorgenommen werden.

```
# Die Datei http.conf

# Laden des Moduls mod_auth_kerb
LoadModule auth_kerb_module
# Directory-Direktive
<Directory /var/www/private>
      AuthType Kerberos                      # Kerberos-Unterstützung
      AuthName "Kerberos Login"
      KrbMethodNegotiate on                  # SPNEGO aktivieren
      Krb5Keytab  /etc/apache/apache.keytab  # Pfad zum Apache-Keytab
      KrbAuthRealms KRBREALM.KRB             # Kerberos Realm
      KrbServiceName HTTP                    # Principal Name des WWW-Servers
      Require valid-user
</Directory>
```

(4) Konfiguration der Browser.

- IE ab Version 5.5. Detallierte Hinweise zur Konfiguration sind in [Sura_02] beschrieben. Der Hostname des Web-Servers muss in der Local Intranet Sicherheitszone angegeben werden (Tools→Internet Options→Security Tab→Local Intranet). Ausserdem muss im Advanced Tab die Option Integrated Authentication aktiviert sein.
- Mozilla Firefox ab Version 1.5. In der Adressleiste wird `about:config` eingegeben. In den dann erscheinenden Einträgen können die Einstellungen zur SPNEGO (Filter nego) oder GSS-API (Filter gss) vorgenommen warden. Unter dem Eintrag `network.negotiate-auth.trusted-uris` müssen URLs eingetragen werden, für die SPNEGO verwendet werden soll, z.B. "`https://www_server.krbrealm.krb`".

6 Evaluierung des Szenarios

Dieses Kapitel soll einen Überblick über die Unterschiede zwischen dem Alt- und dem in dieser Arbeit erarbeiteten Neusystem verschaffen. Darüber hinaus wird die Ausbaufähigkeit des Neusystems beschrieben und dessen Vor- und Nachteile aufgelistet und kurz erläutert.

6.1 Unterschiede zwischen altem und neuem Konzept

Die Einführung von Kerberos ist ausschlaggebend für den wesentlichen Unterschied zwischen Alt- und Neusystem. Kerberos erweitert hierbei das Altsystem um eine Ticket-basierte Proxy-based True SSO-Architektur mit gegenseitiger Client und Server Authentifizierung. Das einmalige Authentifizieren während des Anmeldevorgangs an einem Rechner reicht aus, um ein Ticket zu erhalten und mit diesem weitere kerberisierte Dienste zu verwenden. Im Idealfall werden keine Passwörter, sondern Tickets als Zugriffsberechtigung auf die Dienste über das Netz übertragen. Der Datenaustausch zwischen Client und Server kann mit Hilfe der Kerberos Sessionkeys und AFS-Tickets verschlüsselt werden.

	Altsystem	Neusystem
Anzahl Passwörter	2 (Linux + Windows)	2 (Linux + Windows)
Single Sign-On	Nein	Ja: True SSO
Anzahl Authentifizierungsvorgänge	M: Bei jeder Anwendung (für die vorgestellten Anwendungen dieser Arbeit)	1: beim Anmeldevorgang (für die vorgestellten Anwendungen dieser Arbeit)
Authentifizierungsarchitektur	Passwort-basiert	Ticket-basiert nach einmaliger Eingabe eines Passworts
Anzahl Benutzer-Datenbanken	1: OpenLDAP	Bei Einsatz von: • MIT Kerberos: 3 (LDAP/KDC/AFS) • Heimdal Kerberos: 2 (OpenLDAP/AFS)

Tabelle 7: Hauptunterschiede zwischen Alt- und Neusystem

Für das Authentifizieren von Benutzern im Neusystem müssen nach wie vor zwei

Passwörter verwendet werden. Dies ergibt sich aus der nicht gegeben Interoperabilität zwischen den Linux- und Windows-Betriebssystemen, insbesondere durch die unverträglichen Passwortformate der beiden Betriebssysteme (vgl. auch Kapitel 4.4.4).

Aufbauend auf Kerberos, erweitert OpenAFS das Altsystem um kryptographische Authentifizierung und Datenübertragung und eine fein abgestufte Zugriffskontrolle auf abgelegte Daten im einheitlichen AFS-Namensraum. Dem Benutzer wird weltweit und transparent der Zugriff auf die AFS-Daten ermöglicht.

6.2 Vorteile und Nachteile des neuen Konzepts

In diesem Kapitel sollen die wichtigsten Vor- und Nachteile, die sich durch das in dieser Arbeit vorgestellte SSO-Konzept für den Fachbereich Informatik ergeben, kurz vorgestellt und erläutert werden. Der Einsatz von Kerberos verspricht folgende

Vorteile:

- Proxy-based True SSO mit einer 1:M Beziehung, d.h. mit einem einheitlichen Identifizierungsmerkmal, können Ticket-basiert M verschiedene kerberisierte Anwendungen verwendet werden.
- Sicherheitsgewinn, da die Anzahl der über das Netz transportierten Passwörter vermindert wird und eine gegenseitige Client-Server Authentifizierung stattfindet.
- Benutzerfreundlichkeit. Ergibt sich aus dem ersten Punkt.

Nachteile:

- In der Regel legt Kerberos die Tickets in einer Datei im Verzeichnis `/tmp` des Benutzers ab. Gelingt es einem Angreifer auf diese Datei zuzugreifen, kann der Angreifer im schlimmsten Fall mit dem Ticket auf alle Dienste zugreifen, für die der Benutzer zugriffsberechtigt ist.
- Die Authentifizierungs-Infrastruktur muss homogen sein. In dieser Hinsicht bedeutet homogen, dass jeder Service Provider in einer Kerberos Realm das Kerberos-Protokoll unterstützen muss, indem er z.B. durch die GSS-API kerberisiert oder durch einen kerberisierten Service Provider ersetzt.
- Das separate Windows-Passwort bricht das Konzept des True SSO. Auf längere Sicht sollte die Abschaffung dieses Passworts angestrebt werden, indem Windows-Benutzer gegen eine Kerberos Realm und nicht gegen eine durch

Samba emulierten Windows-Domain Controller authentifiziert werden.

Da AFS auf Kerberos aufsetzt, gelten oben genannte Kerberos Vor- und Nachteil auch für OpenAFS. Durch den Einsatz von OpenAFS ergeben sich neben anderen weitere

Vorteile:

- Skalierbarkeit. Diese bezieht sich vor allem auf verschiebbare Volumes zur Lastverteilung, Erweiterung von AFS durch zusätzliche Server, ohne sein Verhalten zu ändern, Replikation von Daten und Zugriffskontrolllisten
- Ausfallsicherheit. Im AFS können Daten auf mehrere Fileserver repliziert werden, so dass der Zugriff auf Daten stets gewährleistet ist.
- Transparenz. Bei Ausfall eines AFS Servers greifen Clients benutzertransparent automatisch auf verbliebene Fileserver zu.
- Auf Kerberos basierende Authentifizierung. Der Benutzer kann nur dann auf seine Daten zugreifen, wenn er sich gegenüber Kerberos authentifizieren kann.
- Verschlüsselte Datenübertragung. Alle über AFS-Client und –Server ausgetauschten Daten werden netzweit verschlüsselt.
- Flexibilität. Wenn Server gewartet werden müssen oder sich Fehler ankündigen, können Daten im laufenden Betrieb ohne Unterbrechungszeiten auf andere Server verschoben werden. Daneben ist eine flexible Vergabe von Zugriffsrechten möglich.
- Lastverteilung. Da Volumes in einer AFS-Zelle auf verschiedene Server repliziert werden, kann ein Client je nach Lastsituation entscheiden, von welchem Server er die Daten eines Volumes bezieht. Volumes, auf die oft zugegriffen wird, können auf Server verschoben werden, die nicht stark belastet sind.
- Caching auf dem Client. Führt zu beschleunigtem Datenzugriff und geringere Netzlast.
- Globalität. Der einheitliche AFS-Namensraum ist unter Linux und Windows weltweit verfügbar.
- Gute Zusammenarbeit mit dem Network Identity Manager (z.B. Verwaltung der Tickets, Integrated Login).

Nachteile:

- Höherer Administrationsaufwand. Der Einsatz von OpenAFS steigert die Anzahl zu administrierender Server erheblich.
- Neue Kommandos und Zugriffsrechtesteuerung. Administratoren und Benutzer

müssen neue AFS-spezifische Kommandos erlernen.

- Zugriff auf Dateien, die größer sind als der Client-Cache ist problematisch.
- Die Zugriffsrechtesteuerung auf AFS-Dateien ist unter Windows nicht über ein Samba Netzlaufwerk, sondern nur über das AFS-Netzlaufwerk oder per SSH-Sitzung über eine Konsole möglich.
- Benutzerakzeptanz. Aufgrund neuer Kommandos, muss beobachtet werden, ob AFS von den Benutzern als neues Dateiablagesystem akzeptiert wird.

6.3 Ausbaufähigkeit des neuen Konzepts

Das hier vorgestellte SSO-Konzept ist durch Flexibilität und Skalierbarkeit geprägt. Dadurch ist es möglich immer mehr kerberisierte Anwendungen in das SSO-Umfeld zu integrieren. Dies gilt auch für FH-weite Dienste. Es erlaubt neben lokalem einen hochschulübergreifenden skalierbaren Datenaustausch mit anderen Hochschulen und Organisationen.

Neben Kerberos Cross-Realms, die in Kapitel 4.3.2 kurz vorgestellt wurden, unterstützt das SSO-Konzept dieser Arbeit AFS Cross-Cell Authentifizierung. Cross-Cell Authentifizierung bedeutet, dass eine Zelle `CELL1` ein AFS-Ticket einer anderer Zelle `CELL2` erkennt, so dass der Zugriff auf AFS-Daten in `CELL2` möglich ist. Dadurch könnten in der FH fachbereichsspezifische Zellen oder Kerberos Realms implementiert werden, die miteinander in Beziehung gebracht werden können. Das SSO-Konzept ließe sich somit FH-weit implementieren, wobei die Verwaltung der Benutzer auf die einzelnen Fachbereiche der FH delegiert werden können, die je eine eigene Zelle unterhalten können.

Die Sicherheit von Kerberos baut teilweise darauf auf, dass Benutzer ein schwer erratbares Passwort wählen. Genau das ist selten der Fall. Um die Sicherheit weiter zu steigern können Smart-Cards in Verbindung mit Kerberos eingesetzt werden. Der Kerberos-Schlüssel des Benutzers würde auf die Smart-Card ausgelagert und durch eine PIN geschützt. Dies hätte zur Folge, dass der Anmeldevorgang an einem Rechner nicht mehr passwortbasiert, sondern Smart-card-basiert ist und somit eine Passworteingabe bei der Anmeldung an einem Rechner entfällt. Die hier vorgestellte Ticketbasierte Infrastruktur könnte somit um eine Public Key Infrastruktur erweitert werden. Dafür müssen die FH-internen Rechner mit Schnittstellen für Smart Card-Reader ausgestattet werden.

7 Fazit

Die Implementierung eines Single Sign-On ist für eine Hochschule eine anspruchsvolle Aufgabe. Diese Arbeit verdeutlicht, dass die Einführung von Kerberos und dem Andrew File System als Ausgangspunkt für ein Single Sign-On einen grundlegenden Einschnitt in die stabil funktionierende IT-Infrastruktur des Fachbereichs zur Folge hat. Dafür fügt das Konzept aber der heterogenen Umgebung mehr Interoperabilität in Hinsicht auf die Authentifizierung der Benutzer und Dateiablage hinzu. Die Betrachtung der technischen Rahmenbedingungen hat gezeigt, dass der Fachbereich bereits ein effektives und zentrales Benutzermanagement durch einen LDAP-Verzeichnisdienst realisiert hat und somit eine wichtige Grundlage zur Weiterentwicklung der IT-Struktur hin zum Single Sign-On gegeben ist. Die Untersuchung anderer Hochschulen hat gezeigt, dass immer mehr Hochschulen das Thema Single Sign-On verfolgen, in vielen Variationen entwickeln und in vielen Anwendungsbereichen gebrauchen und erfolgreich einsetzen.

7.1 Zusammenfassung der Arbeit

Der Schwerpunkt dieser Arbeit liegt unter Berücksichtigung einer gegebenen IT-Infrastruktur in der Entwicklung einer auf Kerberos aufsetzenden SSO-Lösung. Auf Grundlage der Authentifizierungs-Infrastruktur des Fachbereichs Informatik und der in Kapitel 2.2.2 vorgestellten SSO-Architekturen wurde darauf aufbauend ein Migrationskonzept sowie ein prototypischer Entwurf abgeleitet und vorgestellt.

Das dabei entworfene prototypische System bietet einen Ausgangspunkt für Weiterentwicklungen und unterstützt die Administratoren des Fachbereichs bei der Gestaltung der vorhandenen Architektur und Auswahl relevanter Anwendungen. Es konnte gezeigt werden, dass die vorhandene Infrastruktur durch einen schrittweisen Migrationsablauf und durch Modifizierungen an den entsprechenden Stellen zu einer SSO-Architektur erweitert werden kann.

Eine prototypische Umsetzung des Konzepts wurde in einem ausführlichen Kapitel 5 aufgezeigt, das die gewonnenen Erkenntnisse der vorherigen Kapitel adressiert und reflektiert. Es zeigt die Machbarkeit eines Proxy-based SSO-Szenarios unter Berücksichtigung der Anforderungen des Fachbereichs.

Ein zwischen Alt- und Neusystem vergleichendes Kapitel 6 spiegelt den Nutzen und die Eigenschaften des Neusystems wider. Die Administratoren erhalten ein sehr flexib-

les und skalierbares, der Benutzer ein benutzerfreundliches, sicheres und auf Tickets basierendes System.

7.2 Ausblick

Diese Arbeit bietet Ideen und ein Konzept für eine Proxy-basierte SSO-Architektur und stellt ein lauffähiges, aber noch prototypisches System zur Vorführung der Herangehensweise an die Umsetzung des Konzeptes vor. Für den realen Einsatz in einer sicherheitskritischen Umgebung sind weitere Untersuchungen bestimmter Elemente und deren Verhalten in der vorhandenen Gesamt-Systemumgebung erforderlich.

Vor allem muss genauer untersucht werden, wie in der Windows-Umgebung der Samba-Dienst mit dem Andrew File System interagiert und wie sich insbesondere die Rechtevergabe auf Verzeichnisse und Dateien auswirkt. Um dieser Problematik aus dem Weg zu gehen, könnte alternativ der OpenLDAP Verzeichnisdienst durch ein Microsoft Active Directory-Verzeichnisdienst ersetzt werden. Erfahrungen anderer in dieser Arbeit nicht vorgestellten Organisationen und Hochschulen haben gezeigt, dass dadurch systemweit ein True SSO ermöglicht und erleichtert wird. Das liegt daran, dass das Active Directory einen Kerberos KDC integriert hat, das gleichermaßen von Windows- und UNIX-Umgebungen (vor allem von Samba) genutzt werden kann (vgl. hierzu [Dyer_04]). Auch ließe sich OpenAFS leichter in die Windows-Umgebung integrieren.

Die in der Arbeit vorgeführten praktischen Anwendungsbeispiele dienten lediglich der Vorführung des Umgangs mit dem Prototyp. In einem realen Szenario müsste eine praxistaugliche Testumgebung aufgebaut werden, die die tatsächliche Interoperabilität zwischen den Anwendungen aufzeigt. Diese erlaubt eine Evaluierung des Gesamtsystems und seinen Kommunikationswegen und der Übergabe von Tickets zwischen den Anwendungen. Vor allem müssen Auswirkungen des Systems untersucht werden, wenn das Konzept über lokale Domänen und Firewalls hinaus auf das Internet zur weltweiten Nutzung ausgedehnt wird.

A Anhang

A.1 MIT Kerberos Konfigurationsdateien

Die KDC-Konfigurationsdatei `kdc.conf`.

```
[kdcdefaults]
       # Definitionen zu Ports auf denen der KDC lauscht
       kdc_ports = 88, 750
[realms]
       # Definitionen zu Pfaden, Lebenszeit der Tickets und Schlüsseltypen
       KRBREALM.KRB = {
              database_name = /var/kerberos/krb5kdc/principal
              admin_keytab = /var/kerberos/krb5kdc/kadm5.keytab
              acl_file = /var/kerberos/krb5kdc/kadm5.acl
              key_stash_file = /var/kerberos/krb5kdc/.k5.KRBREALM.KRB
              kadmind_port = 749
              max_life = 10h 0m 0s
              max_renewable_life = 7d 0h 0m 0s
              master_key_type = des3-hmac-sha1
              supported_enctypes  =  des-cbc-crc:normal  des3-hmac-sha1:normal
              des-hmac-sha1:normal des-cbc-md5:normal
 }
```

Die Kerberos Client-Konfigurationsdatei `krb5.conf`.

```
[libdefaults]
       # Angabe des Default Realm Namen/Tickets können weitergeleitet werden
       default_realm = KRBREALM.KRB
       forwardable = true
[realms]
       KRBREALM.KRB = {
              # Definition über Standort der Server und Ports
              kdc = kerberos.krbrealm.krb:88
              admin_server = kerberos.krbrealm.krb:749
              default_domain = krbrealm.krb
       }
[domain_realm]
       # Abbildung der DNS-Domäne auf die Kerberos Realm
       .krbrealm.krb = KRBREALM.KRB
       krbrealm.krb = KRBREALM.KRB
```

A.2 OpenLDAP Konfigurationsdateien

Die LDAP-Server Konfigurationsdatei `slapd.conf`.

```
# Schemas, die integriert werden sollen
Include     /etc/openldap/schema/core.schema
Include     /etc/openldap/schema/cosine.schema
include     /etc/openldap/schema/inetorgperson.schema
include     /etc/openldap/schema/nis.schema
include     /etc/openldap/schema/krb5-kdc.schema
include     /etc/openldap/schema/samba.schema

pidfile     /var/run/slapd.pid
argsfile    /var/run/slapd.args

########################################################################
# ldbm database definitions
########################################################################

Database    bdb
suffix      "dc=krbrealm,dc=krb"
rootdn      "cn=manager,dc=krbrealm,dc=krb"
# rootpw    {CRYPT}6Ocyjs4tzxN6M

directory   /var/lib/ldap

# Indices to maintain for this database
index objectClass                                       eq,pres
index ou,cn,mail,surname,givenname                      eq,pres,sub
index uidNumber,gidNumber,loginShell                    eq,pres
index uid,memberUid                                     eq,pres,sub
index nisMapName,nisMapEntry                            eq,pres,sub
index sambaSID,sambaPrimaryGroupSID,sambaDomainName     eq

# Log Level
loglevel    256

# SASL Configuration for Kerberos authentication
# This is only necessary if you are using GSSAPI (Kerberos) authentication.
sasl-realm KRBREALM.KRB

# Dieser regulärer Ausdruck bildet bildet den Kerberos Principal manager@KRBREALM.KRB
# auf cn=manager,dc=krbrealm,dc=krb ab, der laut rootdn der LDAP-Administrator ist.
sasl-regexp
    uid=manager,cn=krbrealm.krb,cn=gssapi,cn=auth
    cn=manager,dc=krbrealm,dc=krb
# Dieser regulärer Ausdruck bildet Principals auf die uids innerhalb von People ab.
sasl-regexp
    uid=(.*),cn=krbrealm.krb,cn=gssapi,cn=auth
```

```
            uid=$1,ou=People,dc=krbrealm,dc=krb

# Leserecht für alle anderen erlauben.
access to *
    by * read

# Erlaube uid=manager auf alle children zu schreiben
access to dn.children="dc=krbrealm,dc=krb"
        by dn="cn=manager,dc=krbrealm,dc=krb" write
        by * read

# Beispiel für eine ACL für den Benutzer uid=Sambaroot
# Hier darf er auf die Samba-Einträge und objecClass von ou=People schreiben.
access to dn.children="ou=People,dc=krbrealm,dc=krb "
attrs=objectClass,sambaSamAccount
by dn="uid=sambaroot,ou=People,dc=krbrealm,dc=krb " write
        by * read
```

Die LDAP-Client Konfigurationsdatei `ldap.conf`.

```
# Your LDAP server. Must be resolvable without using LDAP.
host 194.95.66.193

# The distinguished name of the search base.
base dc=krbrealm,dc=krb

# Provide an uri with the server name.
uri ldap://kerberos.krbrealm.krb
ssl no
tls_cacertdir /etc/openldap/cacerts
```

A.3 LDIF-Dateien

Die Datei `rootdn.ldif`.

```
# Labor C015
dn: dc=krbrealm,dc=krb
objectClass: dcObject
objectClass: organization
dc: krbrealm
o: Labor C015 - Netzlabor
description: Labor C015 - LDAP Database

# Administrative user for C015 Ldap database
dn: cn=manager,dc=krbrealm,dc=krb
objectClass: organizationalRole
cn: manager
description: Super-User for Ldap Services
```

Die Datei `ounits.ldif`.

```
# Organizational Unit for autmount service
dn: ou=Automount,dc=krbrealm,dc=krb
ou: automount
objectClass: top
objectClass: organizationalUnit

# Organizational Unit for ethers
dn: ou=Ethers,dc=krbrealm,dc=krb
ou: Ethers
objectClass: top
objectClass: organizationalUnit

# Organizational Unit for groups
dn: ou=Groups,dc=krbrealm,dc=krb
ou: Groups
objectClass: top
objectClass: organizationalUnit

# Organizational Unit for hosts
dn: ou=Hosts,dc=krbrealm,dc=krb
ou: Hosts
objectClass: top
objectClass: organizationalUnit
```

```
# Organizational Unit for samba Idmaps
dn: ou=Idmap,dc=krbrealm,dc=krb
ou: Idmap
objectClass: top
objectClass: organizationalUnit

# Organizational Unit for computers in the edugroup
dn: ou=Netgroup,dc=krbrealm,dc=krb
ou: Netgroup
objectClass: top
objectClass: organizationalUnit

#  Organizational Unit for accounts
dn: ou=People,dc=krbrealm,dc=krb
ou: People
objectClass: top
objectClass: organizationalUnit

# Organization Unit for Profiles
dn: ou=Profile,dc=krbrealm,dc=krb
ou: Profile
objectclass: top
objectclass: organizationalUnit
description: Super-User for Ldap Services
```

Die Datei `group.ldif`.

```
dn: cn=users,ou=Groups,dc=krbrealm,dc=krb
objectClass: posixGroup
objectClass: top
cn: users
gidNumber: 500

dn: cn=testgroup,ou=Groups,dc=krbrealm,dc=krb
objectClass: posixGroup
objectClass: top
cn: testgroup
gidNumber: 501
memberUid: sso
```

Die Datei `hosts.ldif`.

```
# Our first machine entry (the "$" at the end of the uid is neccesary)
dn: uid=kerberos$,ou=Hosts,dc=krbrealm,dc=krb
uid: kerberos$
cn: NT Trust Account
```

```
objectClass: account
objectClass: posixAccount
objectClass: top
userPassword: e1NTSEF9RmF6U3R4TXJIUHlpU3lNZk5LYmdnVEpVNWw2VjBUbXk=
loginShell: /dev/null
uidNumber: 20000
gidNumber: 502
homeDirectory: /dev/null
gecos: NT Trust Account

# The second test machine.
dn: uid=linuxclient$,ou=Hosts,dc=krbrealm,dc=krb
uid: linuxclient$
cn: NT Trust Account
objectClass: account
objectClass: posixAccount
objectClass: top
userPassword: e1NTSEF9am9CTGc2UlpReUJsSXpvVjVFSFRGRCthVjRWU01PQTE=
loginShell: /dev/null
uidNumber: 20001
gidNumber: 502
homeDirectory: /dev/null
gecos: NT Trust Account

# The third test machine.
dn: uid=winclient$,ou=Hosts,dc=krbrealm,dc=krb
uid: winclient$
cn: NT Trust Account
objectClass: account
objectClass: posixAccount
objectClass: top
userPassword: e1NTSEF9d3JWc2UzbTBmYVNoejBiMnVsdm1FVGlvOXZtL2RsYU4=
loginShell: /dev/null
uidNumber: 20001
gidNumber: 502
homeDirectory: /dev/null
gecos: NT Trust Account
```

Die Datei `netgroup.ldif`.

```
# The equivalent of our nis netgroup
dn: cn=kerberosgroup,ou=Netgroup,dc=krbrealm,dc=krb
objectClass: nisNetgroup
objectClass: top
cn: kerberosgroup
```

```
nisNetgroupTriple: (kerberos,,krbrealm.krb)
nisNetgroupTriple: (kerberos.krbrealm.krb,,krbrealm.krb)
nisNetgroupTriple: (linuxclient,,krbrealm.krb)
nisNetgroupTriple: (linuxclient.krbrealm.krb,,krbrealm.krb)
nisNetgroupTriple: (winclient,,nitro)
nisNetgroupTriple: (winclient.krbrealm.krb,,krbrealm.krb)
```

Die Datei `passwd.ldif`.

```
# The entry for uid=Samba root, our Samba administrator user.
dn: uid=sambaroot,ou=People,dc=krbrealm,dc=krb
uid: sambaroot
cn: sambaroot
objectClass: account
objectClass: posixAccount
userPassword: polo89
uidNumber: 502
gidNumber: 502
homeDirectory: /dev/null

# The entry for samba's root user that can add workstations to the domain
dn: uid=root,ou=People,dc=krbrealm,dc=krb
uid: root
objectClass: account
objectClass: sambaSamAccount
sambaSID: S-1-5-21-3966423353-351671230-3527248093-1000
sambaLMPassword: polo89
sambaNTPassword: polo89
```

Die Datei `sso.ldif`.

```
# sso, People, krbrealm.krb
dn: uid=sso,ou=People,dc=krbrealm,dc=krb
uid: sso
cn: Daniel Stojceski
objectClass: posixAccount
objectClass: top
objectClass: sambaSamAccount
objectClass: account
loginShell: /bin/tcsh
uidNumber: 501
gidNumber: 501
homeDirectory: /afs/krbrealm.krb/home/sso
gecos: Daniel Stojceski
sambaSID: S-1-5-21-3966423353-351671230-3527248093-41005
sambaPrimaryGroupSID: S-1-5-21-3966423353-351671230-3527248093-41005
```

```
sambaPwdCanChange: 1140465691
sambaPwdMustChange: 2147483647
sambaPasswordHistory:
➔ 0000000000000000000000000000000000000000000000000000000000000000
sambaPwdLastSet: 1140465691
sambaLMPassword: 6B34DE5EAB09D958AAD3B435B51404EE
sambaNTPassword: 8A01F6B29BED845101BFC15610C84EB8
sambaAcctFlags: [U         ]
sambaProfilePath: \\SAMBA\afs\krbrealm.krb\home\sso\nt-profile
sambaHomePath: \\SAMBA\afs\krbrealm.krb\home\sso
sambaHomeDrive: z:
```

Die Datei `sambaDomainName.ldif`.

```
dn: sambaDomainName=KRBREALM.KRB,dc=krbrealm,dc=krb
objectclass: sambaDomain
objectclass: sambaUnixIDPool
objectclass: top
sambaDomainName: KRBREALM.KRB
sambaSID: S-1-5-21-3966423353-351671230-3527248093
uidNumber: 550
gidNumber: 550
```

A.4 Samba Konfigurationsdatei

Die Datei `smb.conf`.

```
[global]
netbios name = samba
workgroup = KRBREALM.KRB
server string = SAMBA-LDAP

# security = ads ginge auch
security = user
encrypt passwords = yes
use kerberos keytab = yes

realm = KRBREALM.KRB
afs username map = %U@krbrealm.krb

log level = 1
log file = /var/log/samba/log.%m

local master = yes
os level = 33
domain master = yes
preferred master = yes
domain logons = yes

logon script = scripts\login2.bat

###################### Ldap Configuration ###############
# Instruct samba to store the passwords in your ldap server
passdb backend = ldapsam:ldap://kerberos.krbrealm.krb
# This defines which backend to use for group id mappings
idmap backend = ldap:ldap://kerberos.krbrealm.krb
ldap admin dn = "uid=sambaroot,ou=People,dc=krbrealm,dc=krb"
# Define where are the machine accounts stored in your ldap server
ldap suffix = dc=krbrealm,dc=krb
ldap machine suffix = ou=Hosts
ldap user suffix = ou=People
ldap group suffix = ou=Groups
ldap idmap suffix = ou=Idmap
#########################################################
```

```
[netlogon]
path = /etc/samba/netlogon
public = no
writeable = no
browseable = no

[afs]
path = /afs/
afs share = yes
afs username map = %u@krbrealm.krb
read only = no
guest ok = yes
```

Literaturverzeichnis

[Apac_06] The Apache Software Foundation – HTTP Server Project, Forest Hill 2006 – URL: http://httpd.apache.org.

[Baiz_98] Baize, E.; Pinkas, D.: The Simple and Protected GSS-API Negotiation Mechanism, RFC 2478, o.O., Dezember 1998 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc2478.txt.

[Balk_85] Balkovich, E.; Lerman, S.; Parmelee, R. P.: Computing in Higher Education: The Athena Experience, in: Communications of the ACM, 28 (1985) 11, S. 1214-1224 – URL: http://cds.icu.ac.kr/course/ice601/reading_list/p1214-balkovich.pdf.

[Bart_05] Bartlett, A.: Samba 4 – Active Directory, Thesis, o.O., 5. Januar 2005 – URL: http://us1.samba.org/samba/news/articles/abartlet_thesis.pdf.

[Blum_02] Blumenburg, S.: Sicherheitsrisiko: Passwort, Stuttgart 2002.

[Buch_03] Buchholz, C.; Paulus, S.: Identity Management, in: Datenschutz und Datensicherheit, 27 (2003) 9, S. 536-540.

[Chem_06] Technische Universität Chemnitz, AFS an der TU Chemnitz – URL: http://www.tu-chemnitz.de/urz/afs/index.html.

[Cler_02] Clercq, J. D.: Singe Sign-On Architectures, in: Davida, G.; Frankel, Y.; Rees, O. (Hrsg.), Infrastrucure Security, LNCS 2437, International Conference, InfraSec 2002, Proceedings, Springer-Verlag, Bristol (UK) 2002, S. 40-58.

[Coge_05] Cogel, B.: SSH und AFS, Vortrag am AFS Workshop, 24.-25. November, Paderborn 2005 – URL: http://imt.uni-paderborn.de/fileadmin/imt/afs-workshop-2005/K%F6ln/AFSPaderborn-SSH.pdf.

[CSS_06] Certified Security Solutions: PuTTY with GSSAPI and Kerberos v0.56b2, o.O. 2006 – URL: http://www.certifiedsecuritysolutions.com/cgi-

bin/dnld_form.pl.

[Cyru_06] Project Cyrus, Carnegie Mellon University, Pennsylvania (USA), Pittsburgh 2006 – URL: http://cyrusimap.web.cmu.edu.

[Dyer_04] Dyer, R. M.; AFS on Windows, OpenAFS Best Practices Workshop, Stanford, California, March 24-26, 2004 – URL: http://www-conf.slac.stanford.edu/AFSBestPractices/Slides/AFSonWindows.pdf.

[File_06] FileZilla, o.O. 2006 – URL: http://filezilla.sourceforge.net.

[Fire_06] Mozilla Foundation: Firefox Web Browser, o.O. 2006 – URL: http://www.mozilla.com/firefox.

[Fisc_01] Fischer-Hübner, S.: IT-Security and Privacy, in: Lecture Notes in Computer Science, Vol. 1958, Springer-Verlag, Berlin et al. 2001.

[Frank_99] Franks, J. et al.: HTTP Authentication: Basic and Digest Access Authentication, RFC 2617, o.O. 1999 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc2617.txt.

[Fumy_05] Fumy, W.; Sauerbrey, J.: Identity & Access Management, in: Kuhlin, B.; Thielmann, H. (Hrsg.), The Practical Real-Time Enterprise, Springer, Heidelberg 2005, S. 259-274.

[Garm_03] Garman, J.: Kerberos - The Definitive Guide, O'Reilly, Sebastopol (USA) 2003.

[Gulb_00] Gulbrandsen, A.; Vixie, P.; Esibov, L.: A DNS RR for specifying the location of services (DNS SRV), RFC 2782, o.O. 2000 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc2782.txt

[Hegd_05] Hegde, V.; Premalatha; Escalante, M.: MIT Kerberos Enhancements for Samba, Vortrag vom 28. April, o.O. 2005 – URL: http://web.mit.edu/jaltman/Public/Samba-XP-Presentation.pdf.

[Heim_06] Heimdal Homepage, o.O. 2006 – URL: http://www.pdc.kth.se/heimdal.

[Hewl_05] Hewlett Packard Company: Select Access - Overview and Features, o.O 2005 – URL: http://managementsoftware.hp.com/products/select/index.html.

[Hodg_02] Hodges, J.; Morgan, R.: Lightweight Directory Access Protocol (v3): Technical Specification, RFC 3377, o.O. 2002 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc3377.txt.

[Hors_99] Horster, P.; Kraaibeek, P.; Wohlmacher, P.: Sicherheitsinfrastrukturen - Basiskonzepte, in: Horster, P. (Hrsg.), Sicherheitsinfrastrukturen, Vieweg Verlag, Braunschweig/Wiesbaden 1999, S. 1-16.

[Hueb_01] Huebsch, C.; Schwarze, T.: AFS – ein verteiltes Dateisystem, Vortrag am 61. Treffen der CLUG, o.O., 9. November 2001 – URL: www.clug.de/vortraege/openafs/afsvortrag.pdf.

[IBM_05] IBM Corporation: IBM Tivoli Software, New York (USA) 2005 – URL: http://www-306.ibm.com/software/tivoli.

[Klün_03] Klünter, D.; Laser, J.: LDAP verstehen – OpenLDAP einsetzen, dpunkt.verlag, Heidelberg 2003.

[Kohl_93] Kohl, J.; Neuman, C.: The Kerberos Network Authentication Service (V5), IETF RFC 1510, o.O. September 1993 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc1510.txt.

[Köln_04] Universität zu Köln (Hrsg.): Jahresbericht 2004, Köln 2004 – URL: http://www.uni-koeln.de/rrzk/dokumentation/jb/04/all.pdf.

[Konq_06] KDE e.V.: Konqueror Homepage, o.O., 2006 – URL: http://www.konqueror.org.

[Kour_06] Kouril, D.: Kerberos Module for Apache, o.O., 2006 – URL: http://sourceforge.net/projects/modauthkerb.

[LDAP_06] The OpenLDAP Foundation, o.O. 2006 – URL: http://www.openldap.org.

[Lend_04] Lendecke, V.: Samba 3 und AFS: Storage wirklich skalierbar, Beitrag im Freien Vortragsprogramm des LinuxTag 2004, o.O., 7. Juni 2004 – URL: http://www.free-it.de/archiv/talks/paper-10199/paper.pdf.

[Libe_06] Liberty Alliance Project, o.O. 2006 – URL: http://www.projectliberty.org.

[Linn_00] Linn, J.: Generic Security Service Application Program Interface, RFC 2743, o.O., Januar 2000 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc2743.txt.

[Mill_88] Miller, S.; Neuman, C.; Schiller, J.; Saltzer, J.: Kerberos Authentication and Authorization System, Project Athena Technical Plan, Section E.2.1, Cambridge, Massachusetts 27. Oktober, 1988 – URL: http://web.mit.edu/afs/athena.mit.edu/user/other/a/Saltzer/www/publications/athenaplan/e.2.1.pdf.

[Mill_92] Mills, D. L.: Network Time Protocol (Version 3) - Specification, Implementation and Analysis, RFC 1305, o.O., März 1992 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc1305.txt.

[MIT_06] Massachusetts Institute of Technology (MIT): Kerberos: The Network Authentication Protocol, o.O. 2006 – URL: http://web.mit.edu/kerberos.

[Möll_06] Möller, M.: Apache Kerberos/SPNEGO module, o.O., 2006 – URL: http://sourceforge.net/projects/modgssapache.

[Morr_86] Morris, J. H. et al.: Andrew: a distributed personal computing environment, in: Communications of the ACM, 29 (1986) 3, S. 184-201 – URL: http://portal.acm.org/citation.cfm?id=5671&jmp=review&dl=portal&dl=ACM .

[Mozi_06] Mozilla Foundation, o.O., 2006 – URL: http://www.mozilla.com.

[MSIE_06] Microsoft Corporation, Microsoft Internet Explorer, 2006 – URL: http://www.microsoft.com/windows/ie/default.mspx.

[Myer_97] Myers, J.: Simple Authentication and Security Layer (SASL), RFC 2222, o.O., Oktober 1997 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc2222.txt.

[Myer_99] Myers, J.: SMTP Service Extension for Authentication, RFC 2554, o.O., März 1999 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc2554.txt.

[Need_78] Needham, R. M.; Schroeder, M. D.: Using Encryption for Authentication in Large Networks of Computers, in: Communications of the ACM, 21 (1978) 12, S. 993-999 – URL: http://www-inst.eecs.berkeley.edu/~cs268/sp02/cached_papers/needham.pdf.

[Nego_06] Negotiateauth project, o.O., 2006 – URL: http://negotiateauth.mozdev.org/index.html.

[Neum_05] Neuman, C.; Yu, T.; Hartman, S.; Raeburn, K.: The Kerberos Network Authentication Service (V5), IETF RFC 4120, o.O. Juli 2005 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc4120.txt.

[NTP_06] The NTP Public Services Project, o.O, 2006 – URL: http://ntp.isc.org/bin/view/Main/SoftwareDownloads.

[Nyst_06] Nyström, T.: Kerberos Telnet for Windows 95/98/ME/NT/2000/2003/XP, o.O. 2006 – URL: http://www.stacken.kth.se/~thn/ktelnet/index.html.

[OAFS_06] OpenAFS, o.O. 2006 – URL: http://www.openafs.org.

[OASI_06A] Organization for the Advancement of Structured Information Standards, o.O. 2006 – URL: http://www.oasis-open.org.

[OASI_06B] Security Assertion Markup Language (SAML), OASIS Standard, März 2006 – URL: http://www.oasis-open.org/committees/security/docs.

[Oden_05] Odenbach, C.: Server- und Dienstestruktur an der Uni Paderborn, Vortrag am AFS Workshop, 24.-25. November, Paderborn 2005 – URL: http://imt.uni-paderborn.de/fileadmin/imt/afs-workshop-2005/Paderborn/Server-

%20und%20Dienstestruktur%20an%20der%20Uni%20Paderborn.ppt.

[OSSH_06] OpenBSD Project: OpenSSH, Calgary, Alberta (Canada) 2006 – URL: http://www.openssh.com.

[Pade_04] IMT der Universität Paderborn (Hrsg.): Jahresbericht 2004, Paderborn 2004 – URL: http://imt.uni-paderborn.de/fileadmin/imt/imt-aktuell/2005/IMT-Jahresbericht-2004.pdf

[Pash_03] Pashalidis, A.; Mitchell, C. J.: A taxonomy of single sign-on systems, in: Lecture Notes in Computer Science, 2727 (2003), S. 249–264 – URL: http://citeseer.ist.psu.edu/cache/papers/cs/31075/http:zSzzSzwww.isg.rhul.ac.ukzSz~xrtczSzcvzSzssotax.pdf/pashalidis03taxonomy.pdf.

[Pass_06] Microsoft .NET Passport, o.O. 2006 – URL: http://www.passport.net.

[Pine_06] University of Washington: Pine - a Program for Internet News & Email, Washington (USA) 2006 – URL: http://www.washington.edu/pine.

[Přik_06] Přikryl, M.: WinSCP, Freeware SFTP and SCP client for Windows, o.O., 2006 – URL: http://winscp.net/eng/docs/lang:de.

[Putt_06] Tatham, S.: PuTTY: A Free Telnet/SSH Client, o.O. 2006 – URL: http://www.chiark.greenend.org.uk/~sgtatham/putty.

[Roes_02] Roessler, G.: Identification and Authentication in Networks enabling Single Sign-On, Graz University of Technology, Graz 2002 – URL: http://www.iaik.tu-graz.ac.at/teaching/11_diplomarbeiten/archive/roessler.pdf.

[Sama_95] Samar, V.; Schemers, R.: Unified Login with Pluggable Authentication Modules (PAM), RFC 86.0, Open Software Foundation, o.O. 1995 – URL: http://www.opengroup.org/tech/rfc/mirror-rfc/rfc86.0.txt.

[Schu_98] Schulze, N.: AFS-Kerberos-5-Migration, Studienarbeit, Technische Universität Chemnitz, Fachbereich Informatik, Meinersdorf 1998 – URL:

http://archiv.tu-chemnitz.de/pub/1998/0042/data/starbeit.ps.

[Secu_06] Secure Endpoints, New York (USA) 2006 – URL: https://www.secure-endpoints.com.

[Send_06] Sendmail Homepag, o.O. 2006 – URL: http://www.sendmail.org.

[Stei_88] Steiner, J. G.; Neuman, C.; Schiller, J. I.: Kerberos: An Authentication Service for Open Network Systems, o.O., 1988 – URL: http://citeseer.ist.psu.edu/cache/papers/cs/7156/http:zSzzSzwww.cs.utah.eduzSz~retraczSzcs6961zSzkerberos88.pdf/steiner88authentication.pdf.

[Sten_05] Stenzel, H.: Verzeichnisdienste und Identity Management in Hochschulen - ein Überblick, in: Praxis der Informationsverarbeitung und Kommunikation (PIK), 28 (2005) 3, S. 176-181.

[Sura_02] Surati, S.; Muckin, M.: HTTP-Based Cross-Platform Authentication via the Negotiate Protocol, o.O., Dezember 2002 – URL: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsecure/html/http-sso-1.asp

[Thun_06] Mozilla Foundation: Thunderbird Email Client, o.O. 2006 – URL: http://www.mozilla.com/thunderbird.

[Voyd_83] Voydock, V. L.; Kent, S. T.: Security mechanisms in highlevel network protocols, in: ACM Computing Surveys., 15 (1983) 2, S.135-171 – URL: http://www.cs.jhu.edu/~rubin/courses/sp03/papers/voydock.kent.pdf.

[VTM_06] GSSAPI/SSPI for PuTTY, o.O., 2006 – URL: http://sweb.cz/v_t_m/.

[Wahl_97A] Wahl, M.; Howes, T.; Kille, S.: Lightweight Directory Access Protocol (v3), IETF RFC 2251, o.O. 1997 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc2251.txt.

[Wahl_97B] Wahl, M.; Kille, S.; Howes, T.: Lightweight Directory Access Protocol (v3): UTF-8 String Representation of Distinguished Names, IETF RFC 2253,

o.O. 1997 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc2253.txt.

[Webb_05] Webb, K.: Considerations when Choosing a Backup System for AFS, AFS & Kerberos Best Practices Workshop at Carnegon-Mellon University, Pittsburgh, Pennsylvania (USA), Juni 20-24, 2005 – URL: http://www.pmw.org/afsbpw05/talks/AFSBackupConsiderations.pdf.

[Welt_04] Welte, R. et al.: Der zentrale Webserver der Universität Freiburg, in: Praxis der Informationsverarbeitung und Kommunikation (PIK), 27 (2004) 1, S. 37-43.

[Zhu_05] Zhu, L. et al.: The Simple and Protected Generic Security Service Application Program Interface (GSS-API) Negotiation Mechanism, RFC 4178, o.O. 2005 – URL: ftp://ftp.rfc-editor.org/in-notes/rfc4178.txt.

[ZKI_06] Zentren für Kommunikation und Informationsverarbeitung in Lehre und Forschung e.V., Heilbronn 2006 – URL: http://www.zki.de.

Gültigkeit der verwendeten Hyperlinks zuletzt überprüft am 26.03.2006.

Glossar

Da Begriffe in verschiedenen Kontexten verschiedene Bedeutungen haben können und zum Teil verschieden verwendet werden, sollen in diesem Glossar Begriffe, die durchgehend in dieser Arbeit verwendet werden, im Sinne dieser Arbeit alphabetisch definiert werden.

- **Authentifizierung**. Im Kontext sicherer Kommunikation in einem Netz bedeutet Authentifizierung die gegenseitige Verifikation der Identität von kommunizierenden Partnern, wobei ein Partner auch eine Softwarekomponente wie ein Service Provider sein kann.

- **Credential**. Legitimations-, Berechtigungsnachweis eines Clients, der ihn berechtigt, Dienste anzufordern.

- **Domäne**. Eine logisch abgeschlossene Netzinfrastruktur. Im Umgangssprachgebrauch auch Arbeitsgruppe genannt.

- **Identifizierungsmerkmale (ID-M)**. Beschreiben und identifizieren einen Benutzer in elektronischer Form. ID-M können verschiedene Formen haben, z.B. Benutzername/Passwort, X.509-Zertifikat, Kerberos-Ticket, usw. Wird in einer IT-Infrastruktur stets dieselbe Form von ID-M verwendet, z.B. nur Kerberos-Tickets zum authentifizieren gegenüber Service Provider, so wird von einheitlichen ID-M gesprochen.

- **Service Provider (SP)**. Ein Dienstanbieter, der Clients authentifizieren kann oder Clients einen bestimmten Dienst oder Inhalt anbietet.

- **Principal**. Ein Client (z.B. Benutzer) oder ein Server (ein Dienst oder ein Host) in einer Realm, dessen Identität verifiziert werden soll und ein Ticket zugewiesen werden kann. Um Kerberos-Clients von Clients anderer Services zu unterscheiden, wurde der Begriff Principal eingeführt [Steiner, 1988].

- **Realm**. Eine Kerberos-Authentifizierungsdomäne, die von mindestens einem KDC verwaltet wird und in der eine definierte Gruppe an Benutzern, Hosts und Dienste

beheimatet sind.

- **Ticket**. Verschlüsseltes Datenpaket, das ein Principal als Identitätsnachweis gegenüber einem Service Provider benutzen kann.

Eidesstattliche Erklärung

Anhand dieser eidesstattlichen Erklärung versichere ich die Bachelor–Abschlussarbeit:

„Konzeption einer Kerberos-basierten Single Sign-On Lösung für ein ausgewähltes Szenario im Hochschulbereich“

im Fachbereich Informatik der Fachhochschule Bonn–Rhein–Sieg, Sankt Augustin 2006, vollständig eigens verfasst und dazu keine anderen, als die angegebenen Quellen und Hilfsmittel verwendet zu haben. Die Arbeit hat in gleicher oder ähnlicher Form noch keinem anderen Prüfungsamt vorgelegen.

Sankt Augustin	31.03.2006	
Ort	Datum	Unterschrift