Universität Leipzig

Fakultät für Mathematik und Informatik

Institut für Informatik

Integration von Methoden und Verfahren zur gesicherten Übertragung von Daten und zur Authentifizierung in ein webbasiertes System

Diplomarbeit

Aufgabenstellung und Betreuung:
Prof. Dr. K. Irmscher
Dipl.-Math. J. Hotzky

vorgelegt von
Jan Falkenreck
geb. am: 28.03.1970
Studiengang (Informatik)

Leipzig, Juli 2002

1 Motivation

Das Internet stellt heute ein weltumspannendes Netz dar, welches sowohl professionell, wissenschaftlich, kommerziell als auch privat genutzt wird. In dieser Umgebung des interpersonellen Datenaustausches werden in zunehmendem Maße Sicherheit und Privatsphäre relevant.

Die Struktur des heutigen Internets ist im Wesentlichen aus Forschungsprojekten des US Verteidigungsministerium und Initiativen von Universitäten entstanden. Im Vordergrund stand die Funktion des Netzes, Authentifizierung und Geheimhaltung waren sekundärer Natur. In dem kommerziell nutzbar gewordenen Netz werden an die Sicherheit und die Privatsphäre hohe Anforderungen gestellt. Mit der zunehmenden Kommerzialisierung durch Firmen und Gesellschaften wird die Integration von Mechanismen wie Authentifizierung und Verschlüsselung von Daten notwendig, um kritische Daten und unternehmensinterne Informationen über das Internet austauschen zu können.

Diese Arbeit beschäftigt sich mit der Integration von Methoden und Verfahren zur gesicherten Übertragung und zur Authentifizierung in eine bestehende Applikation. Es werden Realisierungsmöglichkeiten aufgezeigt und diskutiert.

Sicherheitsmechanismen in verteilten Rechnernetzen, wie sie z.B. in webbasierten Applikationen einsetzbar sind, werden behandelt. Die Grundlagen und Entwicklungen der heutigen Krypto- und Netzwerkstandards werden auf Basis des OSI-Referenzmodells dargestellt und behandelt.

Die erforderlichen Entwicklungen, Methoden und Verfahren zur Integration in eine webbasierte Anwendung werden abgeleitet. Die Arbeit zeigt die Sicherheitsanforderungen an die Anwendung auf und evaluiert bestehende Technologien bezüglich ihrer Verwendbarkeit für eine Authentifizierung in der

Anwendung. Die Integration der Methoden und Verfahren erfolgt prototypisch in das Content Administration Tool (CAT), dem administrativen Teilbereich des get-portals. Besonderer Wert wird dabei auf die Authentifizierung von Nutzern am System und die verschlüsselte Übertragung von Daten über nicht private Netze gelegt.

Zu Beginn der Arbeit werden die kryptographischen Grundlagen und die verwendeten Verfahren und Standards aufgeführt. Anschließend wird das Anwendungsgebiet beschrieben und eine Nutzungsstrategie der Anwendung dargelegt.

Die Integration der Verfahren und Mechanismen in die Anwendung erfolgt unter Berücksichtigung der vorher gesetzten Ziele und der Lösungsansatz wird prototypisch implementiert.

1.1 Zugriffskontrolle

Um Systeme und Anwendungen vor unbefugtem Zugriff zu schützen, muss die Sicherheitspolitik eine Zugriffskontrolle auf die sensiblen Bereiche beinhalten. Es muss sichergestellt werden, dass sich ein zugreifender Nutzer vorher dem System gegenüber als berechtigt ausgewiesen hat. Dazu ist primär die Überprüfung seiner Identität notwendig. Dieser Vorgang wird auch als Authentifizierung oder Authentisierung[1] bezeichnet. Wird die Identität des Nutzers bestätigt, folgt in einem zweiten Schritt die Autorisierung. Durch die Autorisierung wird festgelegt, welche Ressourcen ein Nutzer beanspruchen darf.

[1] In der Literatur wird auch häufig der Begriff authentisieren anstelle von authentifizieren verwendet. Laut Duden [DU 22] bedeutet authentifizieren „die Echtheit bezeugen, beglaubigen“ während authentisieren im Sinne von „glaubwürdig, rechtsgültig machen“ verwendet wird. In dieser Arbeit wird der Begriff authentifizieren verwendet, da dieser dem Kerngedanken der Arbeit, Nutzer anhand von Wissen und/oder Besitz zu identifizieren, näher kommt.

1.2 Authentifizierung

Authentifizierung ist der Prozess der Überprüfung, ob eine Person wirklich diejenige ist, die sie vorgibt zu sein. Dieses Verfahren ist nicht zwangsläufig auf Personen beschränkt. Auch ein Rechner kann sich gegenüber anderer Hardware authentifizieren. Authentifizierung impliziert, dass ein Kommunikationspartner ausreichend Informationen besitzt oder erhält, um zu überprüfen, dass sein Kommunikationspartner auch derjenige ist, der er vorgibt zu sein.

Es gibt verschiedene Methoden der Authentifizierung. Ihnen gemeinsam ist, dass sie im Wesentlichen auf Faktoren wie Wissen und/oder Besitz von bestimmten Eigenschaften oder Objekten basieren. So kann zum Beispiel ein Identitätsnachweis durch verschiedene Faktoren wie der Eingabe eines Passwortes, durch Besitz eines Schlüssels oder einer Token Card oder durch biometrische Eigenschaften oder eine Kombination der verschiedenen Faktoren erfolgen.

Wird die Authentifizierung nur durch einen dieser Faktoren durchgeführt, so spricht man auch von der Einzelfaktorauthentifizierung (Nach [ML00]). Werden zwei Faktoren zur Authentifizierung kombiniert, so spricht man auch von Doppelfaktorauthentifizierung. Dies ist zum Beispiel beim Einsatz von Transaction Number (TAN) Token der Fall. Hier erfolgt die Authentifizierung über den Besitz der Token Card und dem Wissen über die TAN.

Im Weiteren wird zwischen schwacher und starker Authentifizierung unterschieden. Die Einzelfaktorauthentifizierung wird nach [ML00] als schwach und die Doppelfaktorauthentifizierung als stark definiert. Bei schwacher Authentifizierung kommt nur einer der oben beschriebenen Faktoren zum Einsatz, während bei der starken zwei unterschiedliche Faktoren kombiniert werden.

Im Regelfall folgt auf die Authentifizierung die Autorisierung.

1.3 Autorisierung

Autorisierung ist der Prozess der Erlaubnisvergabe an eine Person, gewisse Aktionen ausführen zu dürfen. Aus Gründen der Vertraulichkeit, Verfügbarkeit oder dem Schutz der Integrität von Daten werden Nutzern unterschiedliche Berechtigungen in einem System gewährt. Autorisierung kann in zwei Teilaspekte untergliedert werden. Zum einen werden dem Nutzer gewisse Rechte durch eine höhere Autorität vergeben, z.B. durch einen Administrator. Zum anderen identifiziert die Applikation oder das System die nutzbaren Ressourcen, nachdem sich der Nutzer angemeldet hat. Beide Aspekte sind Bestandteil der Autorisierung. Der Prozess der Autorisierung beruht auf verschiedenen Zugriffsschutzmodellen. Das Modell des diskreten Zugriffschutz (Discrete Access Control, DAC) regelt die Zugriffsmodalitäten zwischen Subjekten und Objekten in Form einer Tabelle [KH 95] S. 106. So wird festgelegt, ob ein Subjekt auf ein Objekt zugreifen darf oder nicht. Beim Zuständigkeitsmodell [KH 95] S. 107 existiert eine regelbasierte Zugriffskontrolle. Ist eine Regel gültig, wird der Zugriff gewährt. Ein Beispiel wäre: Existiert ein Pfad von einem Objekt zu einer Gruppe, so hat diese Gruppe Zugriff auf das Objekt.

1.4 Anforderungen an Systeme zur Authentifizierung

Die Authentifizierung dient dem Nachweis der Identität einer Person oder eines Rechners. Sie kann in der Verbindung mit der Autorisierung zur Zugangsbeschränkung bestimmter Bereiche eingesetzt werden. Ein klassisches Szenario ist der Ausweis für den Zutritt zum Firmengelände. Dieser Ausweis wird einer Authentifizierungsinstanz präsentiert, in diesem Fall dem Pförtner, welcher nun abhängig vom Ausweis eine Autorisierung für den Zutritt zum Gelände erteilt oder nicht.

Die Anforderungen an Systeme zur Authentifizierung können bedingt durch die notwendige Vertrauenswürdigkeit sehr unterschiedlicher Natur sein. Verschiedene Authentifizierungen hemmen den Zugang zu Systemen stärker oder schwächer. Dies

kann durch die Verwendung unterschiedlicher Verfahren und auch deren Kombination erreicht werden.

Die Wirksamkeit von Sicherheitsmechanismen im Allgemeinen wird über den Einsatz an Ressourcen definiert, die ein Angreifer anwenden muss, um die Schutzmechanismen zu durchdringen. Die Vertrauenswürdigkeit wird in der Literatur ([HK 95] S.137) häufig als Kombination der beiden Eigenschaften Wirksamkeit und Korrektheit beschrieben. Die Vertrauenswürdigkeit in ein System steigt mit höherer Sicherheitsfunktionalität und beschreibt somit eine Form des Aufwandes, welcher nötig ist, um eine Manipulation an übertragenen Daten vorzunehmen oder Zugriff auf vertrauliche Informationen zu erhalten. Die Vertrauenswürdigkeit von Systemen lässt sich in Abhängigkeit der Sicherheitskriterien für die Übertragung der Daten sowie auch für die Authentifizierung unterschiedlich einstufen. Daten können unverschlüsselt, kodiert oder verschlüsselt übertragen werden. Authentifizierung kann durch Präsentation eines Gegenstandes (Besitz), Präsentation von Wissen oder das Vorweisen bestimmter Eigenschaften oder aber einer Kombination dieser Faktoren erfolgen. Kombinierte Authentifizierungsanforderungen können einen stärkeren Schutzmechanismus darstellen, als eine Einzelfaktorauthentifizierung, da hier mehrere Schutzmechanismen überwunden werden müssen.

Die sicherheitstechnische Anforderung an ein System zur Authentifizierung hängt von dessen Schutzbedürftigkeit ab. Eine generelle Forderung ist die Korrektheit der Implementierung. Weiter muss der implementierte Mechanismus auch wirksam sein, d.h. für eine Authentifizierung geeignet. Letztlich sind die Mechanismen bedingt durch die technische Weiterentwicklung nach einer gewissen Zeit dahingehend zu überprüfen, ob sie nicht durch den Einsatz neuer Mittel überwunden werden können und somit ihre Sicherheitsfunktionalität verlieren bzw. eingeschränkt wird.

1.5 Sichere Übertragung

Durch die sichere Übertragung wird ein verschlüsselter Kanal für die private Kommunikation zur Verfügung gestellt. Dies ermöglicht den Austausch geheimer Daten über nicht private Netze. Obwohl der Datenverkehr mitgelesen werden kann, können diese nicht eingesehen und ausgewertet werden, da sie verschlüsselt sind.

2 Grundlagen, Verfahren und Standards

2.1 Kryptographische Grundlagen

Alice will Bob eine Nachricht übermitteln, deren Inhalt nur Bob bekannt sein soll. Hierzu bedienen sich die beiden eines nur Alice und Bob bekannten Schlüssels, mit dem Alice die Nachricht codiert. Bob decodiert die Nachricht mit dem Schlüssel und erhält die gewünschten Informationen. Fängt nun ein Dritter die Nachricht von Alice an Bob ab, so sieht er nur den codierten Datenstrom. Ist es ihm jedoch möglich, in den Besitz des Schlüssels zu kommen, so kann er ebenfalls die Nachricht entschlüsseln. Eine andere Möglichkeit besteht darin, die Nachricht mit allen Schlüsseln des verwendeten Schlüsselraumes zu entschlüsseln und das Resultat auf Sinnhaftigkeit zu überprüfen und so die Informationen zu extrahieren. Die Geheimhaltung steht und fällt also mit der verwendeten Verschlüsselung der Daten und der Geheimhaltung des Schlüssels.

2.1.1 Verschlüsselung

2.1.1.1 Symmetrische Verschlüsselung

Bei der symmetrischen Verschlüsselung wird für die Verschlüsselung und Entschlüsselung der gleiche Schlüssel verwendet. Eine Nachricht N wird mit einem Schlüssel K durch eine Operation Op in ein Chiffrat C verwandelt. Der gleiche Schlüssel K dient dazu, C über Op^{-1} (evtl.Op = Op^{-1}) wieder in N zu überführen ([KF98] S. 78). Das Problem bei der symmetrischen Verschlüsselung ist die Geheimhaltung des Schlüssels. Beide Kommunikationspartner müssen im Besitz des Schlüssels K sein. Wird der Schlüssel beim Schlüsselaustausch von einer dritten Person abgefangen, so kann diese das Chiffrat ebenfalls dekodieren. Die Nachricht ist nicht mehr privat.

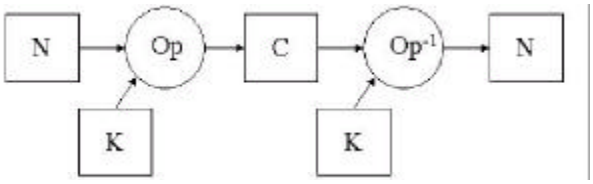

Abbildung 2-1 Ablauf der symmetrischen Verschlüsselung

Der Vorteil dieses Verfahrens ist die performante Ver- und Entschlüsselung. Die Verschlüsselung selbst findet durch Konfusion oder Diffusion der Bitfolgen statt. Diese Operationen werden häufig mehrfach hintereinander ausgeführt. Unterschieden wird zwischen den vornehmlich eingesetzten Blockchiffrierern und Stromchiffrierern. Bei der Blockchiffrierung wird die zu verschlüsselnde Bitfolge in Blöcke gleicher Länge zerlegt und jeder Block verschlüsselt. Die Stromchiffrierung hingegen verschlüsselt den Datenstrom fortlaufend.

Bekannte Verfahren für die symmetrische Verschlüsselung ([BS96] S. 309 ff.) sind der Data Encryption Standard (DES), Triple DES (3DES), International Data Encryption Algorithm (IDEA), Rivest's Cipher[1] 2,4,5 (RC2, RC4, RC5), und Rijndael der Gewinnerkandidat für den Advanced Encryption Standard (AES). Die symmetrische Verschlüsselung wird in der Literatur ([KF98] S. 78) auch häufig als Private-Key-Algorithmus bezeichnet.

2.1.1.2 Asymmetrische Verschlüsselungsverfahren

Das Problem des Schlüsseltausches der symmetrischen Verfahren wird bei den asymmetrischen Verfahren umgangen, indem zum Chiffrieren und Dechiffrieren verschiedene Schlüssel zum Einsatz kommen [EA01]. Die Ver- und Entschlüsselung der Daten findet mittels eines Schlüsselpaares statt, welches aus einem geheimen (privaten) und einem öffentlichen Schlüssel besteht. Über mathematische Formeln wird das Chiffrat C der Nachricht N mit dem öffentlichen Ko und der Klartext mit dem privaten Schlüssel Kp berechnet oder umgekehrt. Um eine Nachricht N zu verschlüsseln, muss

der Benutzer A ein Schlüsselpaar „Privater Schlüssel (Kp(A)) / öffentlicher Schlüssel (Ko(A))“ erzeugen. Ko(A) wird dazu an den Kommunikationspartner B verteilt. Kp(A) verbleibt vor fremdem Zugriff geschützt beim Benutzer. Um eine verschlüsselte Nachricht an seinen Kommunikationspartner A zu versenden, verschlüsselt B die N mit dem öffentlichen Schlüssel Ko(A) von A. Das Chiffrat C kann wiederum nur mit dem privaten Schlüssel Kp(A) von A entschlüsselt werden. Es ist auch für B nicht möglich, C(Ko(A)) zu decodieren. N kann nur wieder erzeugt werden, indem Kp(A) auf C(Ko(A)) angewendet wird.

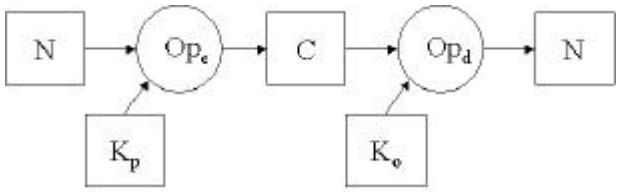

Abbildung 2-2 Ablauf der Asymmetrischen Verschlüsselung

Dreht man dieses Verfahren um, so kann Kp(A) auch zum Signieren von Verträgen und Nachrichten dienen. Verschlüsselt A einen Vertrag V mit Kp(A), so kann dieser nur mit Ko(A) entschlüsselt werden. Da nur A im Besitz von Kp(A) ist, kann nur er das Dokument verschlüsselt haben. Die Entschlüsselung von V(Kp(A)) ist durch den öffentlich verfügbaren Schlüssel Ko(A) natürlich beliebigen Personen möglich. In diesem Fall ist die Verschlüsselung als Signierung des Dokumentes V durch A zu werten.

Der öffentliche Schlüssel Ko kann und sollte öffentlich zugänglich gemacht werden. Dies kann zum Beispiel durch einen Verzeichnisdienst erfolgen. Asymmetrische Verfahren werden aus diesem Grund auch häufig als Public.Key-Verfahren bezeichnet. Somit wird das Dilemma der Secret-Key-Algorithmen gelöst, die für die sichere Kommunikation erst einen sicheren Kanal für den Schlüsseltausch benötigen.

Die Schlüsselerzeugung wird durch eine mathematische Funktion realisiert, welche in einem Vorgang die zwei Schlüssel Ko und Kp generiert. Die beiden Schlüssel sind wie

[1] RC wird auch als „Ron’s Code“ bezeichnet, nach dem Entwickler Ron Rivest

schon beschrieben, voneinander abhängig und getrennt zu verwenden. Die wesentliche Forderung dieses Verfahrens ist Geheimhaltung des privaten Schlüssels. Ferner sollte Kp nicht aus Ko berechenbar oder bestimmbar sein.

Das folgende Beispiel berechnet die Schlüssel für den Rivest-Shamir-Adleman (RSA) Algorithmus:

- Finde P und Q, zwei große Primzahlen (z.B.1024-bit)
- Wähle E so, dass E größer als 1 ist und kleiner als PQ und so, dass E und (P-1)(Q-1) relativ prim sind. (E und (P-1)(Q-1) haben keine gemeinsamen Primfaktoren). E muss nicht prim sein, aber ungerade.
- Berechne D so, dass (DE - 1) geradzahlig teilbar durch (P-1)(Q-1). Also DE = 1 (mod (P-1)(Q-1))

Der öffentliche Schlüssel Ko ist das Paar (PQ, E). Der private Schlüssel Kp ist D. PQ ist der Modulo. E ist der öffentliche Exponent. D ist der geheime Exponent.

- Verschlüsselt wird nun mit C = (N^E) mod PQ, gefordert: N < PQ. (N Klartextnachricht als Integer, Chiffretext als Integer)
- Enschlüsselt wird mit N = (C^D) mod PQ.

Abbildung 2-3 Beispiel für einen asymmetrischen Verschlüsselungsalgorithmus: Der RSA-Algorithmus von RSA Security Inc.

Die asymmetrischen Verfahren stellen in der Praxis keinen Ersatz für die symmetrischen dar. Es gibt zwei Gründe, die dies verdeutlichen:

1. Public-Key-Verfahren benötigen einen höheren Rechenaufwand und sind aus diesem Grund im Regelfall um ein Vielfaches weniger performant als symmetrische Verfahren. Da die asymmetrische Verschlüsselung um Faktoren 100 bis 1000 rechenintensiver als die symmetrische ist, wird sie häufig nur zum Austausch von kleinen Datenmengen, wie zum Beispiel einem symmetrischen Sitzungsschlüssel verwendet.

2. Ein weiterer Nachteil der asymmetrischen Verfahren ist Ihre Verwundbarkeit gegen chosen-plain-text Angriffe. Ist N eine Nachricht aus einer Menge von n möglichen Klartextnachrichten, und das Chiffrat C=E(N), mit E als Enkryptfunktion, dann muss ein Kryptanalytiker nur alle n Nachrichten verschlüsseln und mit C vergleichen. Symmetrische Verfahren sind hierdurch nicht gefährdet, da Chiffrierversuche mit unbekannten Schlüsseln nicht durchgeführt werden können. Bei den Public-Key-Verfahren ist jedoch der öffentliche Schlüssel bekannt.[BS96] S.39

Die Sicherheit der Verfahren beruht im Wesentlichen auf mathematischen Problemen der Komplexitätstheorie, wie zum Beispiel der Komplexität der Primfaktorzerlegung großer Zahlen (vgl. Abbildung 2-3) oder der Berechnung diskreter Logarithmen An dieser Stelle sei auf [BS 96] S.306 verwiesen. Asymmetrische Verschlüsselungsverfahren sind beispielsweise RSA und ElGamal.

2.1.1.3 Hybride Verschlüsselungsverfahren

Um das Problem des Schlüsselaustausches bei der symmetrischen Verschlüsselung zu umgehen und dennoch die Vorteile des geringeren Rechenaufwandes zu nutzen, kombinieren hybride Verschlüsselungsverfahren, häufig auch als hybride Kryptosysteme bezeichnet, die vorher erläuterten Methoden. Diese zeichnen sich dadurch aus, dass der symmetrische Schlüssel mittels eines asymmetrischen Verfahrens verschlüsselt wird. Im Regelfall wird dazu der öffentliche Schlüssel des Kommunikationspartners genutzt. Nun kann der kodierte, symmetrische Schlüssel sicher zwischen den Kommunikationspartnern ausgetauscht werden. Die anschließende Kommunikation kann dann performant verschlüsselt mittels des symmetrischen Schlüssels ablaufen.

Ein Beispiel:
A und B möchten miteinander sicher kommunizieren. A schickt B seinen öffentlichen Schlüssel $K_o(A)$. B generiert einen symmetrischen Sitzungsschlüssel K_{sym} , verschlüsselt diesen mit $K_o(A)$, und sendet das Chiffrat $C=E_{Ko(A)}(K_{sym})$ an A. A dechiffiriert nun C mittels $K_p(A)$,so das $K_{sym} = D_{Kp(A)}(C)$. Die weitere Kommunikation zwischen A und B wird nun mit K_{sym} symmetrisch verschlüsselt.

Der Vorteil bei diesem Protokoll ist, dass der Sitzungsschlüssel erst erzeugt wird, wenn er benötigt wird. Anschließend kann er wieder vernichtet werden. Dadurch wird das Risiko einer Kompromittierung des Verfahrens verringert.

2.1.2 Wirksamkeit der Algorithmen

2.1.2.1 Zufallszahlen

Computer sind deterministische Automaten, die bei Eingabe einer Nachricht von einem Zustand in einen vorhersagbaren Zustand wechseln. Bei identischem Startzustand sind identische Maschinen nach der gleichen Eingabe im gleichen Zustand. Die Schlüssel in einem Verschlüsselungssystem werden mit Hilfe von Zufallszahlen festgelegt. Da es in deterministischen Automaten keine Zufälle gibt, können in Computern nur Pseudo-zufallszahlen erzeugt werden ([BS96] S. 53). Sind diese Zufallszahlen nicht hinreichend zufällig, sondern beinhalten versteckte Regelmäßigkeiten, ist es möglich, einen Schlüssel zu erraten.
„So wurde z.B. bei einer der ersten Versionen des Netscape Navigators der für die SSL-Verschlüsselung notwendige Schlüssel aus Parametern des Rechners abgeleitet, die bei genauerer Betrachtung nicht so zufällig waren, wie die Programmierer angenommen hatten.“ ([KF98] S. 125)

2.1.2.2 Angriffstechniken

Eine einfache Methode, einen symmetrisch verschlüsselten Text zu dechiffrieren, ist die Dechiffrierung mit allen möglichen Schlüsseln des Schlüsselraumes. Gegen diesen Angriff gibt es keine Abwehrmöglichkeit. Aus diesem Grund sollte der Schlüsselraum ausreichend dimensioniert sein. Durch Kryptoanalyse ist es unter Umständen möglich, aus einem zumindest teilweise bekannten Klartext und zugehörigem Chiffretext den Schlüssel zu ermitteln. In ([BS96] S.6 ff.) sind verschiedene Angriffstechniken beschrieben.

Für asymmetrische Verfahren gibt es im Wesentlichen 3 Angriffstechniken:

1. Umkehren der Gleichung der Einweg-Hashfunktionen (vgl. 2.1.3) durch sehr leistungsfähige Rechner. Lösung: größere Schlüssellängen
2. Finden von Methoden durch die die Umkehrung der Einweg-Hashfunktionen sehr viel schneller möglich ist. Lösung: keine. Es müssen aktuelle Informationen besorgt werden, ob ein Verfahren noch als sicher eingestuft werden kann.
3. Man-in-the-Middle Angriff: Verhindern des korrekten Austausches des öffentlichen Schlüssels der Kommunikationspartner. Lösung: persönliche Übergabe der Schlüssel oder Zertifizierung.

2.1.2.3 Wirksamkeit von Verschlüsselungssystemen:

Die Wirksamkeit von Verschlüsselungssystemen hängt wesentlich von den folgenden vier Punkten ab:

1. Wirksamkeit der Algorithmen
2. Sicherheit bei der Bedienung des Systems
3. Korrektheit der Implementierung
4. Sicherheit bei der Verteilung der Schlüssel

Wie schon in Kapitel 1.4 erläutert, wird die Wirksamkeit von Sicherheitsmechanismen über den Einsatz an Ressourcen definiert, die ein Angreifer anwenden muss, um die Schutzmechanismen zu durchdringen ([KH95] S.137). Sind die Algorithmen unwirksam oder schwach oder ist die Implementierung inkorrekt, ist auch das System schwach. Die Wirksamkeit der Algorithmen bei der Verschlüsselung hängt auch von der Länge der verwendeten Schlüssel ab. Eine größere Schlüssellänge ist im Regelfall ein Garant für mehr Sicherheit. Die Sicherheit bei der Verteilung der Schlüssel wird noch ausführlicher im Abschnitt über Zertifikate aufgegriffen.

Ein elementarer Punkt der Wirksamkeit der Systeme ist der sicherheitsbewußte Umgang der Nutzer mit dem System. Das beste Verschlüsselungssystem ist unbrauchbar, wenn die Nutzer sich der Verantwortung nicht bewusst sind und beispielsweise durch unsachgemäßen Umgang private Schlüssel publiziert werden.

2.1.3 Hashfunktionen

Hashfunktionen werden verwendet, um jeder beliebig langen Nachricht eine Nachricht fester Länge zuzuordnen. Eine Einweg-Hashfunktion[1] funktioniert nur in einer Richtung. Aus einer Nachricht lässt sich durch die Einweg-Hashfunktion der Hashwert berechnen. Es soll jedoch nur mit erheblichem Aufwand möglich sein, zu dem Hashwert eine passende Nachricht zu finden. Der Zweck dieser Funktionen liegt darin, einen Fingerabdruck einer Nachricht zu erzeugen, um mit einer gewissen Wahrscheinlichkeit aussagen zu können, ob eine Nachricht mit dem Original übereinstimmt. Hieraus ergibt sich folgendes Problem: Wenn Nachrichten beliebiger Länge auf Zahlen mit fester Bitzahl, z.B. 128 oder 160 Bit, abgebildet werden, gibt es mehrere Nachrichten, die zu einem Hashwert passen, man spricht auch von einer Kollision. Eine Einweg-Hashfunktion muss deshalb so beschaffen sein, dass sich aus der Nachricht leicht der Hashwert berechnen lässt, sich jedoch zu dem Hashwert keine Nachricht finden lässt. Da beliebig lange Nachrichten auf kurze Hashwerte abgebildet werden, existieren zu jedem Hashwert beliebig viele Nachrichten. Wichtig ist die Kollisionsfreiheit der Hashfunktion, das heißt, es muss ausreichend schwierig sein, zwei Nachrichten zu demselben Hashwert zu generieren ([BS96] S. 36 ff.). Ferner sollte sich durch jedes geänderte Bit der Nachricht im Mittel die Hälfte aller Bits im Hashwert ändern. Jedes Bit in der Nachricht beeinflusst jedes Bit der kryptografischen Prüfsumme.

Typische Verfahren sind Message Digest 2 (MD2), Message Digest 4 (MD4), Message Digest 5 (MD5), Secure Hash Algorithm (SHA) 1, RACE Integrity Primitives Evaluation Message Digest (RIPEMD-160). Neben diesen Verfahren existieren weitere Kompressionsfunktionen, die zur Erzeugung und Prüfung eines Hashwertes einen Schlüssel erfordern. Diese Art der kryptografischen Prüfsummen wird auch als Message Authentification Code (MAC) bezeichnet. Hierzu gehört auch der vom National Institute

[1] Einweg-Hashfunktionen werden auch als One-Way-Hashfunktionen, Message Digest, Kompressionsfunktion oder Fingerabdruck bezeichnet.

of Standards and Technologies (NIST) 2001 veröffentlichte Entwurf des Keyed-Hash Message Authentication Code, kurz HMAC (RFC2104).

2.1.4 Signaturen

Eine Signatur soll die Authentizität eines Dokumentes beweisen. Im Normalfall ist dies eine Unterschrift auf einem Blatt Papier. Eine digitale Signatur stellt ebenfalls die Authentizität eines Dokumentes sicher und kann zusätzlich die Integrität garantieren.

Wie schon im Kapitel über die asymmetrischen Verfahren angesprochen, eignen sich diese auch zum Signieren von Dokumenten. Für große Datenmengen sind die Verfahren nicht leistungsfähig genug. Darum wird üblicherweise nur ein Fingerabdruck signiert.

Mit den Einweg-Hashfunktionen ist es möglich, einen Fingerabdruck der Nachricht zu erzeugen. Dieser Fingerabdruck kann auf Grund seiner Länge effizienter verschlüsselt werden als die Nachricht.

Um die Authentizität und/oder die Integrität der Nachricht zu prüfen, wird der verschlüsselte Fingerabdruck mit dem öffentlichen Schlüssel entschlüsselt. Nun wird der Fingerabdruck der Nachricht erneut berechnet und mit dem entschlüsselten Fingerabdruck verglichen. Sind beide identisch, ist die Authentizität und auch die Integrität der Nachricht nachgewiesen ([BB99] S. 104).

Es ist ersichtlich, wie wichtig die Kollisionsfreiheit der Hashfunktion für die Sicherheit des Verfahrens ist. Da die Nachricht nicht selbst verschlüsselt wird, darf es nicht möglich sein, eine Nachricht zu generieren, die den gleichen Fingerabdruck erzeugt wie im folgenden Beispiel beschrieben. A signiert eine Nachricht N mit seinem privaten Schlüssel $K_{p(A)}$, in der er unterschreibt, dass er B 10 Euro schuldet. B entschlüsselt die Signatur $S_{Kp(A)}(N)$ mit dem öffentlichen Schlüssel $K_{o(A)}$ und erhält so die Signatur S(N). Nun sucht B eine Nachricht, die zu dieser Signatur passt und aussagt, dass A B 1000 Euro schuldet. Da beide Nachrichten die gleiche Signatur liefern, kann nicht festgestellt

werden, welches das Original ist. Das ist bei finanziellen Transaktionen nicht wünschenswert.

2.2 Zertifikate

Im Abschnitt 2.1.2 wurde das Problem des sicheren Schlüsselaustausches schon angesprochen. Wie kann A sicher sein, den öffentlichen Schlüssel von B zu erhalten, und nicht einen von C, welcher wiederum den abgefangenen Schlüssel von B für einen Man-in-the-middle Angriff nutzt?

Hierfür existieren sogenannte Zertifikate. Ein Zertifikat ist ein Ausweis, welcher von einem vertrauenswürdigen Dritten T unterzeichnet wird. Vertrauen A und B T, so können sie ihre öffentlichen Schlüssel von T zertifizieren lassen. T unterzeichnet mit seinem Zertifikat die Zertifizierungsanforderungen von A und B und bestätigt damit, dass er den von A und B gemachten Angaben vertraut. Nun kann A B vertrauen, ohne ihn zu kennen, da das von B präsentierte Zertifikat von T unterzeichnet ist und A T traut.
Ein Zertifikat ist also ein digitales Dokument, welches einen öffentlichen Schlüssel an ein Individuum oder ein Objekt bindet.

Eine Zertifikatanforderung ist ein öffentlicher Schlüssel, der an Attribute gebunden wird, zum Beispiel den Distinguish Name (DN) (vgl. Abschnitt 2.2.2) des Servers. Die Zertifikatanforderung wird einer Zertifizierungsstelle[1] (CA) (vgl. Abschnitt 2.2.1) zur Signierung übergeben. Vertraut die CA den gemachten Angaben, evtl. durch vorgelegte Dokumente oder persönliche Ausweisung, so signiert sie die Anforderung mit ihrem privaten Schlüssel.
Mit dem Zertifikat der CA, welches den öffentlichen Schlüssel der CA enthält, kann das so erstellte Zertifikat überprüft werden.

[1] Zertifizierungsstellen werden auch als Trustcenter oder CA (Certification Authority, eine Instanz, die Zertifikate unterschreibt) bezeichnet

2.2.1 Zertifizierungsstellen

Zertifizierungsstellen sind vertrauenswürdige Dritte, welche Zertifikate ausgeben. Die CA signiert mit ihrem privaten Schlüssel die Zertifikatanforderung. Dadurch garantiert ihr öffentlicher Schlüssel, dass sich der Inhaber des durch sie signierten Zertifikates ihr gegenüber authentifiziert hat und das Zertifikat damit gültig ist. Ein durch eine CA signiertes Zertifikat bindet die Identität, z.B. eines Servers, an ein Public-Private-Key[1] Paar. Mit diesem Schlüsselpaar können nun elektronische Daten ver- oder entschlüsselt werden. Durch das Zertifikat kann der Kommunikationspartner überprüft werden.
Die Sicherheitspolitik einer CA wird im Regelfall der Allgemeinheit zugänglich gemacht. Ein übliches Verfahren ist die Publizierung über Webserver.

Zu der Zertifizierungsstelle gehört meistens noch eine Registration Authority. Die CA unterzeichnet die Zertifikatanforderungen. Dies ist aus Sicherheitsgründen häufig ein alleinstehendes, gut gesichertes System, da es die privaten Schlüssel der CA beinhaltet. Die Registration Authority (RA) dient dazu, Zertifikatanforderungen entgegen zu nehmen, an die CA weiterzuleiten und die von der CA unterzeichneten Zertifikate an die Nutzer zurückzugeben.

2.2.2 Bestandteile eines Zertifikates

In seiner einfachsten Form enthält ein Zertifikat nur einen öffentlichen Schlüssel und einen Namen. Im Allgemeinen sind jedoch mehr Attribute integriert. Elementare Bestandteile von Zertifikaten sind daher häufig ,

- eine Seriennummer, z.B. eine laufende Nummmer
- die Laufzeit mit dem Beginn und dem Ende der Gültigkeit,
- der Distinguish Name (DN) , dies kann der Name des Inhabers sein oder ein unverwechselbares Pseudonym,
- der öffentliche Schlüssel des Signaturinhabers,
- der Name der Zertifizierungsstelle,

- die digitale Signatur der Zertifizierungsstelle,
- weitere Angaben z.B. für die Nutzung des Zertifikates.

Die Seriennummer ist eine von der Aussteller-CA ausgegebene, eindeutige Nummer.

Der DN ist eine Zeichenkette, die dieses Zertifikat eindeutig identifiziert. Der DN setzt sich wiederum aus Zeichenketten zusammen, die eine Person, ein System (Server) oder eine Organisation identifizieren. DN's werden im allgemeinen in Verzeichnissen wie Lightweight Directory Access Protokoll (LDAP)[2] für X.509 konforme Zertifikate zur Identifizierung verwendet. Der DN identifiziert einen Eintrag im Verzeichnis eindeutig. Das Verzeichnis ist hierarchisch strukturiert, so dass der DN die Position (Pfad) innerhalb der Hierarchie angibt.

Der DN setzt sich aus den folgenden Komponenten zusammen:

- CN=Common Name
- E= Email address
- OU=organizational unit
- O=organization
- L=locality
- ST=state or province
- C=country name

Je nach Verwendungszweck des Zertifikates enthalten die Attribute unterschiedliche Elemente.

---

[1] Public-Private-Key Paar: Ein Paar Schlüssel bestehend aus öffentlichem und zugehörigem privaten Schlüssel.

[2] LDAP: Lightweight Directory Access Protokoll, Teil der X.500 Protokollfamilie

In Tabelle 2-1 und Tabelle 2-2 sind Beispielwerte für verschiedene Zertifikate angegeben:

Zertifikat Verwendung	CN	E	OU	O	L	ST	C
Server zertifikat	Rechner-name	Email des Administrators	Abteilung, die den Server betreibt	Firma*	Standort des Servers*	Bundes-land	Land
Personen zertifikat	Name der Person	Email der Person	Abteilung	Firma*	Stadt*	Bundes-land	Land
CA Zertifikat	Name der CA	Email des Administrators	CA-Name oder Beschreibung	Firma			Land

Tabelle 2-1 DNs für unterschiedliche Zertifikate (* bedeutet die Felder sind Plichtfelder)

Zertifikat Für	CN	E	OU	O	L	ST	C
Person	Jan Falkenreck	Jan.falkenreck @get-ag.com	Entwicklung	get AG	Leipzig	Sachsen	de

Tabelle 2-2 DN für ein Personenzertifikat

Zertifikate haben außerdem Erweiterungen, mit denen sich die Verwendung des Zertifikates steuern lässt. Diese Erweiterungen werden als *Zertifikaterweiterungen* oder certificate *extensions* bezeichnet. Zertifikaterweiterungen werden noch in Bezug auf X.509 konforme Zertifkate in Kapitel 2.3 behandelt.

2.2.3 Zertifikatwiderruflisten

Zertifikate haben eine vorbestimmte Laufzeit. Damit die Gültigkeit der Zertifikate vor Ablauf des Zeitraumes beschränkt werden kann, existieren Zertifikatwiderruflisten (CRL)[1]. Die CRL enthält alle Zertifikate einer CA, die ausgegeben aber ungültig sind.

[1] Zertifikatwiderruflisten werden auch als Certificate Revocation List (CRL) bezeichnet

Wird ein Zertifikat ungültig, z.B. durch Bekanntwerden des zugehörigen privaten Schlüssels, wird es in der CRL veröffentlicht. Die CRLs werden abhängig von der Policy der CA in bestimmten Abständen, oder falls erforderlich auch früher, veröffentlicht.

2.2.4 Zertifikatketten

Ketten von Zertifikaten können durch mehrfache Zertifizierung erstellt werden. Wird A von T zertifiziert und B wird von S zertifiziert, so herrscht zwischen A und B kein Vertrauen. Lassen sich jedoch T und S von V zertifizieren und signieren die Zertifikatanforderungen von A und B mit den von V zertifizierten Schlüsseln, so existieren Zertifikatketten von V nach A und von V nach B. A kann nun B vertrauen, da T und S von V zertifiziert sind und somit V vertrauen. Das Zertifikat von V stellt in diesem Fall das Rootzertifikat in der Zertifikatkette dar.

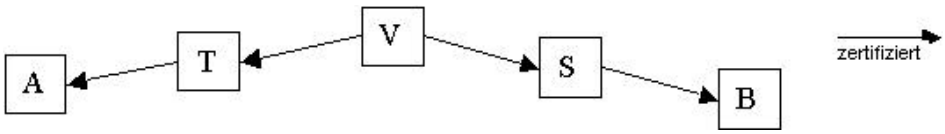

Abbildung 2-4 Zertifikatketten

Ein Problem stellt sich allerdings. Existiert in der Zertifikatkette eine „schlechte“ oder nicht vertrauenswürdige CA, so ist die gesamte Kette affektiert. Ist S eine „Bad CA“, eine schlechte CA, dann vertraut A B, obwohl B kein Vertrauen verdient. Durch Anerkennung eines CA Zertifikates wird die Entscheidung über das Vertrauen eines Kommunikationspartners auf die CA verlagert. Es ist bei der Zertifizierung also sehr genau darauf zu achten, wer zertifiziert wird.

2.2.5 Verzeichnisdienste

Die von einer CA ausgegebenen Zertifikate müssen nach dem Signaturgesetz[1] (SigG) in Verzeichnissen veröffentlicht werden. Diese Verzeichnisdienste sind Pflichtdienstleistungen der Zertifizierungsstelle. Soll also eine SigG-konforme CA aufgebaut werden, muss ein Verzeichnisdienst betrieben werden.

2.2.6 Kodierung von Zertifikaten

Zertifikate und Zertifikatanforderungen können in den unterschiedlichsten Formaten vorliegen. Tabelle 2-3 zeigt die wesentlichen Formate und die dazugehörigen Dateisuffixe. Auf die PKCS-Standards wird im Weiteren noch eingegangen.

Format	**Dateiendung**
PEM, ASCII Format (auch als „Base64 kodiert“ oder „ASCII armored“ bezeichnet)	pem, arm
DER, Binäres Dateiformat	der
Unbestimmtes Format, kann Binär oder ASCII sein	cer, crt
PKCS#12 Format. Dieses enthält den privaten Schlüssel und sollte mit Passphrase geschützt sein.	p12 (früher pfx)
PKCS#7 Format. Wird von S/MIME verwendet	p7c

Tabelle 2-3 Formate von Zertifikaten und typische Endungen

[1] Gesetz zur digitalen Signatur (Signaturgesetz - SigG) (Artikel 3 des Gesetzes Regelung der Rahmenbedingungen für Informations- und Kommunikationsdienste vom 13. Juni 1997; Stand: 01.05.2001)

2.3 Standards

2.3.1 X.509

Der X.509 Standard von der International Telecommunications Union (ITU) entworfen, ist eine Beschreibung der Struktur von Zertifikaten und wurde im Zusammenhang mit den Empfehlungen zu X.500 Verzeichnisdiensten veröffentlicht.
Die erste Version X.509 wurde 1988 veröffentlicht. Durch Revision von X.500 1993 folgte die für Verzeichniszugriffskontrolle erweiterte Version 2 1993. Die aktuelle Version X.509v3 von 1994 wurde schließlich entwickelt, da mehr Felder für Erweiterungen benötigt wurden, unter anderem auch durch die Nutzung von Privacy Enhanced Mail (PEM)[1]. Eine genaue Beschreibung der Entwicklung ist in [HFP99] zu finden.

X509v3 Zertifikate bestehen aus den folgenden Attributen:

- Die Versionsnummer (version), 3 bei X.509v3
- Eine eindeutige Seriennummer (serialNumber)
- ID des Signaturalgorithmus (signature)
- Den Namen der Zertifizierungsstelle (issuer)
- Die Gültigkeitsdauer mit dem Beginn und dem Ende der Gültigkeit (validity)
- Den Namen des Inhabers oder ein Pseudonym (subject)
- Den öffentlichen Schlüssel des Inhabers und der Algorithmus, mit dem er erstellt wurde (subjectPublicKeyInfo)
- Eine eindeutige ID des Inhabers (subjectUniqueID)
- Eine eindeutige ID der Zertifizierungsstelle (issuerUniqueID)
- Erweiterungen (extensions)

Die Erweiterungen sind für Zertifikate von großer Bedeutung, da mit ihnen die Verwendung der Zertifikate gesteuert werden kann. Voraussetzung ist jedoch, dass die Anwendungssoftware die Erweiterungen kennt. Es gibt Erweiterungen, die zum Beispiel ein Zertifikat eindeutig als Herausgeberzertifikat identifizieren. Andererseits kann die

[1] PEM: Privacy Enhanced Mail, RFC 1421- 1424

Verwendung eines Zertifikates für die Signierung von Code, die Authentifizierung oder zum Versenden von Emails beschränkt werden.

Viele Erweiterungen können als *critical* markiert werden. Dazu wird ein Flag gesetzt, das *Critical Bit*. Ist dieses für eine Erweiterung gesetzt, so muss laut [HFP99] Kap. 4.2 jede Anwendung das Zertifikat zurückweisen, die die Erweiterung nicht kennt. Ist die Erweiterung nicht als *critical* markiert, kann sie übergangen werden.

Die Standard Extensions in X.509 sind (nach [HFP99]):

Extension	Beschreibung
Authority Key Identifier	Unterstützung der Überprüfung von Zertifikatketten
Subject Key Identifier	Enthält den Hashwert eines PublicKeys eines Zertifikates
Key Usage	Beschränkt die Benutzung eines Zertifikates
Private Key Usage Period	Erlaubt einem Herausgeber einen vom Zertifikat abweichenden Zeitraum für die Nutzung des Schlüssels anzugeben
Certificate Policies	OID (Object Identifier) bei OID-Listen oder URI auf Policy der CA
Policy Mappings	Wird in CA-Zertifikaten genutzt. Paare von OIDs. IssuerDomainPolicy wird an subjectDomainPolicy gemappt. (Policies sind equivalent)
Subject Alternative Name	Dient zur Angabe weiterer Bezeichner für ein Subjekt in das Zertifikat
Issuer Alternative Names	Dient zur Angabe weiterer Bezeichner für einen Herausgeber des Zertifikats
Subject Directory Attributes	Sollte nur in lokalen Umgebungen gesetzt werden. Muss non-critical sein.
Basic Constraints	Sagt aus, ob es sich um ein CA Zertifikat handelt oder nicht. Sollte als CA Zertifikat immer critical sein.
Name Constraints	Indiziert einen Namensraum für Subjekte. Alle untergeordneten Zertfikate eines Pfades müssen hierdrin enthalten sein.
Policy Constraints	2 Möglichkeiten: 1. Verbieten von Policymappings 2. Jedes Zertifikat im Pfad muss einen Policy Identifier haben
Extended key usage field	Beschränkt wie auch Key Usage die Nutzung des Zertifikates, z.B. serverAuth
CRL Distribution Points	Lokalität, wo CRLs der Herrausgeber CA abgerufen werden können.

Tabelle 2-4 X.509 Standard Extensions

Jede Organisation kann ihre eigenen Erweiterungen registrieren lassen, um die Verwendung zu steuern oder weitere Informationen bereitzustellen. Bekannt sind die Netscape Certificate Extensions, mit denen man die Verwendung von Zertifikaten (zu mindest für Netscape Browser) beeinflussen kann. Diese werden teilweise auch von OpenSSL unterstützt.

2.3.2 Public Key Cryptography Standards

Die “Public Key Cryptography Standards” (PKCS) werden von der Firma RSA Data Security Inc. in Zusammenarbeit mit anderen Firmen als Entwurf eines Industrie-standards für eine Schnittstelle der Public-Key-Kryptographie herausgegeben. Die PKCS sind keine Standards im eigentlichen Sinne, sie wurden nicht durch ein Standardisierungskomitee standardisiert. Allerdings finden sie häufig in Public-Key-Infrastrukturen Anwendung. Anhang 7.10 gibt einen Überblick über die Standards.

Für die Verwendung von Zertifikaten in webbasierten Systemen sind insbesondere PKCS#7 für die Verwendung kryptographischer Verfahren, PKCS#10 für den Austausch von Zertifikaten, und PKCS#12 für die Speicherung persönlicher Zertifikate mit privatem Schlüssel interessant.

2.3.3 Secure Multipurpose Internet Mail Extensions

Secure Multipurpose Internet Mail Extensions (S/MIME)[1] ist ein von RSA und anderen Firmen entwickelter, auf Multipurpose Internet Mail Extensions (MIME)[2] basierender Standard für den Versand mehrteiliger, multimedialer oder binärer Dateien.
S/MIME nutzt PKCS#7 für die Verwendung kryptographischer Verfahren und PKCS#10 zur Übermittlung von Zertifikaten. Die symmetrische Verschlüsselung wird durch RC2 und 3DES, die asymmetrische mittels RSA realisiert. Als Hashfunktionen kommen MD5 und SHA-1 zum Einsatz. S/MIME wird beispielsweise vom Netscape Messanger 4.5 und

[1] Secure Multipurpose Internet Mail Extensions, RFC 2311 und RFC 2312

[2] Multipurpose Internet Mail Extensions, RFC 2045, RFC 2046

Microsoft Outlook verwendet. Durch Integration von X.509 Zertifikaten in diese Programme können Emails mit Attachments signiert und verschlüsselt übertragen werden.

2.4 Protokolle für verschlüsselte Übertragung

Für die verschlüsselte Übertragung von Daten sind verschiedene Protokolle entwickelt worden. Zu diesen Protokollen gehört das Secure Request Technologie (SRT) 1.0 Protokoll von BROKAT, welches dem Secure Socket Layer (SSL) 3.0 Protokoll ähnelt, jedoch für geringe Bandbreiten und den Einsatz in Java-Clients optimiert wurde. Ein weiteres Protokoll, welches heute auf Grund mangelnder Verbreitung keine Bedeutung mehr hat, ist das Secure Hypertext Transfer Protocol (SHTTP).
Ein für die verschlüsselte Übertragung in HTTP basierten Systemen häufig verwendetes Protokoll ist das SSL Protokoll und sein Nachfolger, das Transport Layer Security (TLS) V1.0 Protokoll.

2.4.1 Secure Socket Layer Protokoll

Das SSL Protokoll wurde von Netscape Communications mit dem Ziel entwickelt, eine verschlüsselte Kommunikation zwischen einem Client und einem Webserver zu etablieren und findet aktuell in der Version 3 Verbreitung. Das SSL-Protokoll ist zwischen der Anwendungsschicht und der Transportschicht angesiedelt. Es besteht aus 5 Protokollen, die unterschiedliche Aufgaben haben. Das SSL-Application-Data-Protokoll, SSL-Alert-Protokoll, SSL-Change-Cipherspec-Protokoll und das SSL-Handshake-Protokoll basieren auf dem SSL-Record Protokoll, welches wiederum auf der Transportschicht aufsetzt. Durch das SSL-Handshake-Protokoll werden die Kommunikationspartner authentisiert und die kryptographischen Verfahren und Parameter ausgehandelt.

Folgende Abbildung verdeutlicht die Funktion der einzelnen Protokolle:

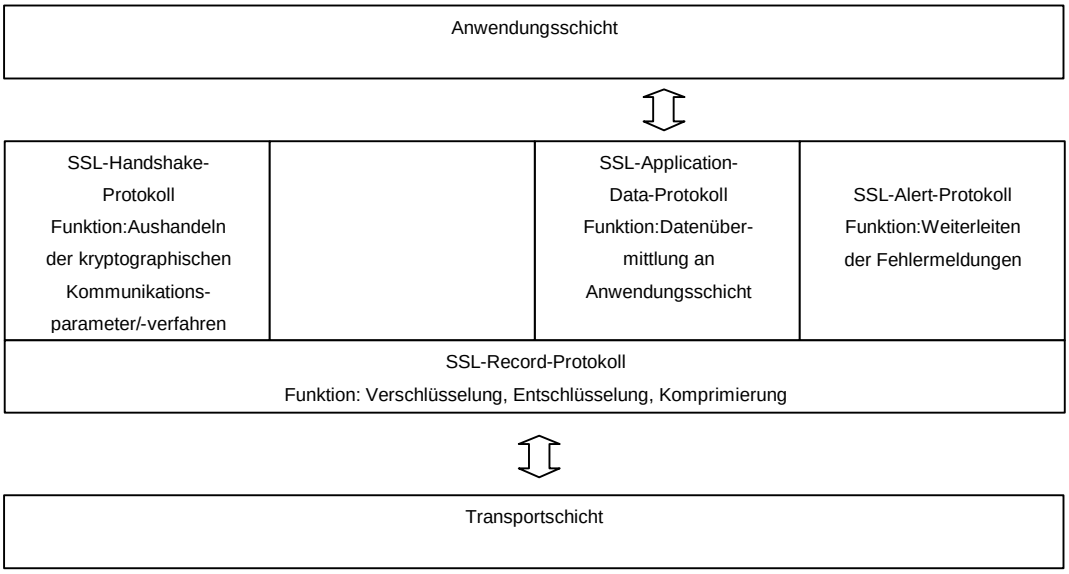

Tabelle 2-5 Schema des SSL Protokolls

SSL unterstützt in der Version 3 nach [NE98] für die Verschlüsselung DES, 3DES, SKIPJACK, RC2, RC4 und RSA und MD5 und SHA-1 als Hashverfahren. Die Authentifizierung wird mit RSA und DSA durchgeführt. KEA und RSA key exchange werden für den Schlüsseltausch verwendet.

Der Server benötigt für den Betrieb ein Zertifikat. Je nach Authentifizierungsanforderung des Webservers benötigt auch der Client ein Zertifikat.

Der Ablauf der Kommunikation des SSL-Handshake erfolgt nach folgendem Schema:

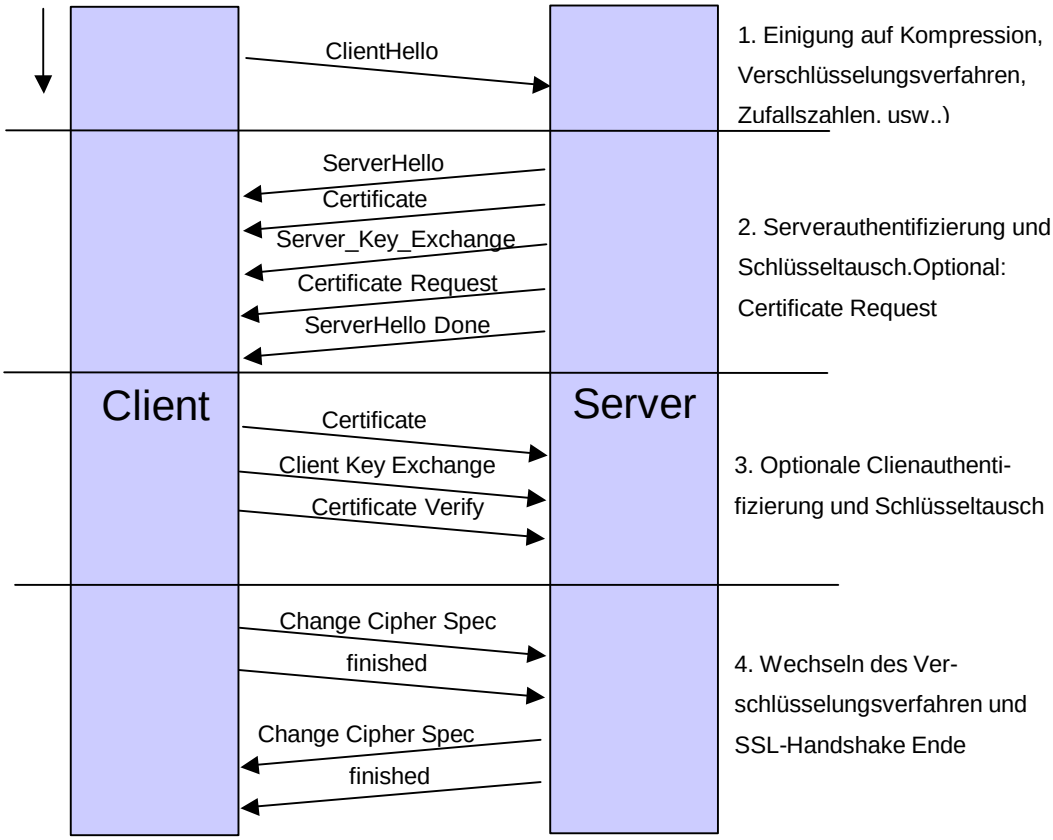

Abbildung 2-5 Zeitlicher Ablauf des SSL Handshake

Nach dem Ablauf des SSL-Handshake Protokolls existiert ein sicherer Übertragungskanal zwischen Browser und Server über den weitere Daten übertragen werden können.

Der Einsatz von SSL in Webservern

Durch die optionale Clientauthentifizierung ist die Möglichkeit gegeben, Nutzer, welche auf diesen Server zugreifen, zu authentifizieren. Der Browser präsentiert das Nutzerzertifikat und der Server akzeptiert oder verweigert den Zugang. Kann das präsentierte Zertifikat nicht erfolgreich überprüft werden, so wird der Zugang abhängig

von der Zugriffspolicy verweigert oder gestattet. Es gibt Möglichkeiten Webserver so zu konfigurieren, dass generell keine, eine optionale oder eine Clientauthentifizierung erforderlich ist.

Die Nutzung von SSL in Bezug auf virtuelle Webserver (VW) unterliegt jedoch Restriktionen bezüglich der Flexibilität. Werden auf einem Webserver mehrere VW betrieben, so sendet der Client (Browser) den Namen des angewählten Webservers an den Server. Zur Zeit des SSL Handshakes sind jedoch nur IP Adresse und der Port und nicht der Name des VW bekannt, da der Browser den Hostnamen noch nicht gesendet hat. So kann pro IP bzw. pro Port nur ein virtueller Webserver mit SSL betrieben werden.

Sollen mehrere virtuelle Server mit SSL betrieben werden, so gibt es zwei Varianten der Realisierung. Zum einen können mehrere ipbasierte Webserver aufgesetzt werden. In diesem Fall wird jedem VW eine eigene IP zur Verfügung gestellt. Das SSL-Handshake findet also mit der zu dem VW gehörigen IP statt.

Eine weitere Möglichkeit ist die Differenzierung der VW durch Zuordnung unterschiedlicher Ports. Das SSL-Handshake findet nun mit der gleichen IP aber mit differenzierten Ports statt. Da das Hypertext Transfer Protocol Secure (HTTPS) jedoch üblicherweise auf dem wellknown[1] Port 443 läuft, ist dieses Szenario weniger anwenderfreundlich, da zur URL noch der Port angegeben werden muss.

2.4.2 Transport Layer Security Protokoll

Das Transport Layer Security Protokoll (TLS) V1.0[2] basiert auf der von Netscape entworfenen Protokollspezifikation des SSL 3 Protokolls. Die Unterschiede zum SSL sind nicht sehr dramatisch. So ist auch TLS ein zweischichtiges Protokoll bestehend aus

---

[1] welknown: wohlbekannter Port. Diese Portnummer wird intuitiv mit einem bestimmten Protokoll assoziiert

[2] Transport Layer Security, Protokoll Version 1.0, RFC 2246

TLS Record Protokoll und TLS Handshake Protokoll. Allerdings wurde dem Protokoll ein optionales Session Caching Schema spendiert. Dies ermöglicht es, die Anzahl der neu aufgebauten Verbindungen zu reduzieren. Die so eingesparten rechenzeitintensiven Public-Key-Operationen wirken sich wiederum auf die Performance aus. Ein weiterer Designschwerpunkt liegt auf der Reduzierung des Netzwerkverkehrs.

SSL und TLS können nicht ohne weiteres miteinander kommunizieren. TLS beinhaltet jedoch Mechanismen, die zu SSL 3 abwärtskompatibel sind und so eine Kommunikation ermöglichen können. Standardmäßig sendet TLS V1.0 im Versionsfeld als Major Version 3 und als Minor Version 1 {3,1} und wird auch als SSL 3.1 angezeigt, SSL sendet {3,0}.

Die Erzeugung von Pseudozufallszahlen ist in TLS V1.0 im Gegensatz zur SSL-Spezifikation definiert und wird durch ein iteratives Verfahren realisiert. Statt des MAC bei SSL wird der Standard HMAC verwendet.

2.5 Sessiontracking

Unter einer Session oder auch Sitzung versteht man die eindeutige Zuordnung zweier oder mehrerer Kommunikationsendpunkte zu einer bestehenden Interaktion. Im Sinne dieser Arbeit wird es sich hierbei im Folgenden in der Regel um die Kommunikation zwischen einem Webbrowser als Client und einem Webserver handeln.

Das Hypertext Transfer Protocol (HTTP), Version 1.1 (RFC 2068, RFC 2616) ist vom Design her ein zustandsloses Protokoll. Das bedeutet, dass mehrere Anfragen von demselben Client nicht ohne weitere Hilfsmittel eindeutig einer Session zugeordnet werden können. Um nun eine Sessionverfolgung[1] in ein HTTP-basiertes System zu integrieren, bedarf es einiger Hilfsmittel seitens der Programmierung. Es gab in der

[1] Sessionverfolgung wird auch als Sessiontracking bezeichnet

Vergangenheit verschiedene Strategien, ein derartiges Sessionverfolgungssystem zu implementieren. Üblicherweise wird eine Session erzeugt, indem ein Client einen Verbindungsaufbau zu einem Server erzeugt. Abhängig von der Sessionverfolgung werden Optionen gesetzt oder ausgehandelt, die es dem Server erlauben, weitere Anfragen des Clients eindeutig zu identifizieren und einer Session zuzuordnen. Im Folgenden werden die wesentlichen Strategien kurz beschrieben.

2.5.1 URL Encoding

URL Encoding[1] ist eine einfache Methode, eine Sessionverfolgung eines Clients durchzuführen. Hierbei wird die URL um Daten erweitert, die es dem Server ermöglichen, diese Anforderung im Kontext einer Session zu interpretieren. Zum Beispiel wird eine Session ID in die URL kodiert, die dann von dem aufgerufenen CGI Programm oder Servlet interpretiert werden kann. Diese Methode kann von dem Server verwendet werden, wenn der Client zum Beispiel keine Cookies akzeptiert.

2.5.2 Cookies

Ein elementarer Nachteil des HTT Protokolls ist, wie schon erwähnt, seine Zustandslosigkeit. Das bedeutet, dass jede Anfrage von einem Browser unabhängig von den vorangegangenen Anfragen behandelt wird. Cookies können diesen Makel beseitigen, in dem sie Informationen über die Verbindung Client-Server und somit den Zustand der Verbindung speichern ([KM97]).

Cookies sind Paare aus einem Schlüssel und einem zugehörigen Wert und weiteren Attributen wie „path“ und „expires“, welche im Speicher oder in einer Datei auf dem Client gespeichert werden. Cookies sind per Definition an eine URL gebunden. Wird eine erneute Anfrage an einen Server gerichtet, so wird die URL und der Pfad des gespeicherten Cookies mit dem des Servers verglichen und bei Übereinstimmung wird der Cookie an den Server übermittelt, womit diesem die in dem Cookie gespeicherten

[1] URL Encoding oder auch URL Rewriting genannt

Informationen, z.B. Warenkorb oder NutzerID, wieder zur Verfügung stehen. Über Cookies kann eine Anwendung auf dem Server nun ermitteln, welcher Rechner, nicht aber welcher Nutzer auf das Angebot zugegriffen hat. Dafür ist eine weitere Applikationslogik notwendig.

Allerdings lassen sich Cookies in den meisten Webbrowsern deaktivieren. Aus Sicherheitsgründen wird davon auch Gebrauch gemacht, da entsprechend formulierte Cookies Sicherheitslücken in den Browsern ausnutzen können. Cookies sind ansonsten ein effizientes Mittel, wenn diese für das Sessiontracking eingesetzt werden.

Transiente Cookies

Bei Transienten Cookies handelt es sich um eine echte Teilmenge der Cookies. Ihre besondere Eigenschaft ist, dass sie nicht auf Datenträgern gespeichert, sondern im Arbeitsspeicher gehalten werden. Wird der verwendete Browser geschlossen, werden auch die transienten Cookies zerstört. Dieser Mechanismus bietet so einen erweitertern Schutz vor Sichtung des Inhalts dieser speziellen Cookies. Transiente Cookies werden beispielsweise in dem Produkt Keon Webpassport der Firma RSA Data Security Inc. Eingesetzt [RS01].

2.5.3 Secure Socket Layer Sessions

Das Secure Socket Layer (SSL) Protokoll hat integrierte Mechanismen, die es dem Serverprozess erlauben, eine eindeutige Zuordnung mehrerer Anforderungen eines Clients vorzunehmen [NE98]. Die Daten, die aus dem Verbindungsaufbau von Client und Server verhandelt werden, können für eine Sessionverfolgung durch die SSL_Session_ID verwendet werden (vgl. 7.5). Im Gegensatz zu den oben beschriebenen Methoden die auf HTTP aufsetzen, ist das Sessionhandling durch SSL auf Basis der Transportschicht des OSI-Modell angesiedelt.

2.6 Methoden und Verfahren zur Authentifizierung

Wie bereits erwähnt, existieren unterschiedliche Verfahren zur Authentifizierung. Es gibt keine generische Authentifizierungsmethode, die allen Anforderungen entspricht. Für die Wahl der Methoden sind unter anderem das Einsatzgebiet und der Aufwand entscheidend.

2.6.1 Passwörter

Passwörter stellen hinsichtlich der Authentifizierung die einfachste Methode dar, sind aus diesem Grund sehr verbreitet und beruhen auf der Festlegung einer geheimen Zeichenkette zwischen den Kommunikationsparteien. Sie stellen eine Einzelfaktorauthentifizierung dar (Wissen um das Passwort). Bei der Authentifizierung eines Nutzers gegenüber einem System wird üblicherweise eine Nutzerkennung und das Passwort in verschlüsselter oder unverschlüsselter Form an den Rechner übertragen, welcher dann anhand gespeicherter Informationen durch Vergleich die Richtigkeit der Nutzerkennung und des dazugehörigen Passwortes überprüft. Das Passwort kann, wie schon erwähnt, in unverschlüsselter und verschlüsselter Form auf dem Rechner gespeichert sein, wobei letztere Variante vorzuziehen ist.
Bei Unix Rechnern z.B. wird die vom Nutzer eingegebene Zeichenkette mittels einer Crypt-Funktion verschlüsselt und mit der auf dem Rechner verschlüsselt gespeicherten Zeichenkette verglichen. Sind beide Zeichenketten identisch, wird die Authentizität des Nutzers als gegeben angenommen und der Zugriff auf das System entsprechend der Autorisierung des Nutzers gestattet.

In einem webbasierten System kann ein Webserver bestimmte Verzeichnisse vor nicht autorisiertem Zugriff schützen, indem nur ein Zugriff mittels Nutzerkennung und Passwort auf dieses Verzeichnis möglich ist. Dazu werden die Zugriffsberechtigungen für das Verzeichnis festgelegt und bei Zugriff überprüft, ob eine Berechtigung vorliegt. Der IBM HTTP Server hat z.B. die Option, den Verzeichniszugang über Passworte zu schützen. Hier wird in jedes zu schützende Verzeichnis eine Datei mit bestimmten Authentifizierungsrichtlinien und dem Pfad für die Nutzerpasswortdatei gelegt. Erfolgt ein

Zugriff auf das Verzeichnis, wird gegen diese Datei geprüft und bei Berechtigung Zugriff gewährt.

Die Probleme bei der Authentifizierung sind ersichtlich. Die Sicherheit steht und fällt mit der Komplexität der Passwörter. Sind die Passwörter leicht zu erraten, weil sie z.B. auf Worten aus Wörterbüchern basieren, mitzulesen oder sogar unverschlüsselt abgelegt sind, gewähren diese im Grunde keine Sicherheit, da eine Erlangung der Kenntnis des Passwortes nur einen geringen Aufwand für einen Angreifer bedeutet. Damit ein System, welches über Passwortauthentifizierung verfügt, eine gewisse Sicherheit vor unbefugtem Zugriff gewährleistet, müssen Passwörter bei Eingabe unsichtbar, bei Übertragung und Speicherung verschlüsselt sein. Ferner sollten Mindestlängen und Restriktionen bezüglich der Passwortwahl vergeben werden.

2.6.2 Zertifikate

Zertifikate können zur Realisierung einer Doppelfaktorauthentifizierung eingesetzt werden. Es ist dabei zu beachten, dass ein Zertifikat immer nur den öffentlichen Schlüssel an eine Person bindet. Die Person muss natürlich auch im Besitz des zum Zertifikat gehörenden privaten Schlüssels sein. Der Zugriff auf den privaten Schlüssel ist häufig durch ein Passwort geschützt.

2.6.2.1 Softwarezertifikate

Softwarezertifikate sind Zertifikate, die als Datei auf einem Datenträger oder in einem anderweitigen Speicher gehalten werden. Softwarezertifikate finden z.B. in Web-browsern Anwendung. Der Netscape Communicator 4.78, Opera 5 und der MS Internet Explorer 5.5 unterstützen die Verwendung von Zertifikaten und können diese importieren und exportieren. Es können persönliche Zertifikate nach PKCS#12 mit privatem Schlüssel behandelt werden. Mit diesen ist es möglich, sich gegenüber einem SSL Server zu authentifizieren. Es können mit diesen Zertifikaten auch Emails signiert und verschlüsselt werden.

Softwarezertifikate sind sicherer als Passworte, da sie auf der Grundlage von Wissen und Besitz basieren. Für die Authentifizierung ist der Besitz eines privaten Schlüssels erforderlich. Ferner ist der Zugriff auf diesen Schlüssel durch ein Passwort geschützt (Wissen).

2.6.2.2 Smartcardzertifikate

Smartcards sind elektronische Karten, die als kleine Rechner oder Informationsspeicher meist in Scheckkartengröße implementiert sind. Je nach Design dienen sie verschiedenen Anwendungszwecken. Als Rechner implementierte Smartcards bestehen mindestens aus RAM, ROM und CPU. Optional ist bei Smartcards, welche Daten auf der Karte verschlüsseln können, noch eine Cryptounit und ein Random Number Generator vorhanden. Auch ein EEprom zur Speicherung von Daten ist häufig implementiert. Der Zugriff auf die Karte erfolgt über einen Smartcard Reader. Ein Beispiel für eine Smartcard, mit eigener Virtueller Maschine nach JavaCard 2.1 Standard, ist die GemXpresso 211PK Card von Gemplus.

Smartcardzertifikate sind Zertifikate, die mit dem privaten Schlüssel auf externen Karten untergebracht sind. Der Vorteil gegenüber den Softwarezertifikaten besteht darin, das nicht direkt auf sie zugegriffen werden kann. Sie können nicht einfach kopiert werden. Ein Beispiel ist die Smartcard eToken RSA Security Inc.

2.6.3 PIN/TAN Kombination

Die Authentifizierung über eine Kombination aus Personal Identification Number und Transaction Number (PIN/TAN) ist ein zweistufiges Authentifizierungsverfahren und aus dem Online-Banking geläufig. Der Zugriff auf das Konto erfolgt über die geheime PIN. Für jede weitere Aktion muss ein Einmalpasswort, die TAN, eingegeben werden. Diese ist nur dem Nutzer bekannt und nur für eine Aktion gültig. Danach wird sie ungültig. Für mehrere aufeinanderfolgende Aktionen besitzt der Nutzer eine Liste mit TANs. Wird die PIN bekannt, so wird für die Manipulation immer noch eine TAN benötigt. Ohne die Kenntnis der PIN ist die TAN Liste wertlos.

2.6.4 Biometrische Verfahren

Biometrische Verfahren nutzen bestimmte, einmalige Eigenschaften von Individuen zur Authentifizierung aus. So sind zum Beispiel die Struktur der Netzhaut des Auges oder der Fingerabdruck eindeutig einer Person zuzuordnen. Verfügt ein System über entsprechende Geräte für die Überprüfung, so kann ein Individuum anhand dieser Charakteristika identifiziert werden. Biometrische Verfahren stehen am Anfang der Marktreife und sind kapitalintensiv. Durch Ausnutzen der subjektgebundenen Eigenschaften lassen sich bisher scheinbar keine vertrauenswürdigen Authentifizierungssysteme aufbauen ([CT11] S.123)

2.6.5 Einschätzung der Verwendbarkeit und Wirksamkeit der Methoden zur Authentifizierung

Wie schon erwähnt, hängt der Einsatz der Authentifizierungsmethoden von dem Einsatzfeld und der Umgebung ab. In einem webbasierten System macht es häufig keinen Sinn, biometrische Verfahren zur Authentifizierung zu nutzen. Diese erfordern kapitalintensive Geräte, wie z.B. Netzhautscanner oder Lesegeräte für Fingerabdrücke. Außerdem müssen diese Geräte an jedem Client vorhanden sein.

Eine konkrete, wissenschaftliche Studie zum Vergleich der verschiedenen Verfahren konnte nicht ermittelt werden. Dies liegt vermutlich auch daran, dass es eine nichttriviale Aufgabe ist, eine geeignete Metrik zur Bewertung von Sicherheit aufzustellen [CE10]. Da die Kosten für ein Authentifizierungssystem in Abhängigkeit der gewünschten Schutzstufe, dem Anwendungszweck, den Umgebungsbedingungen und anderen Faktoren variieren, skizziert das folgende Diagramm eine Einschätzung der Relation Sicherheit/Kosten. Dieses Diagramm wird auch durch diverse Artikel über Authentifizierungsmechanismen untermauert.

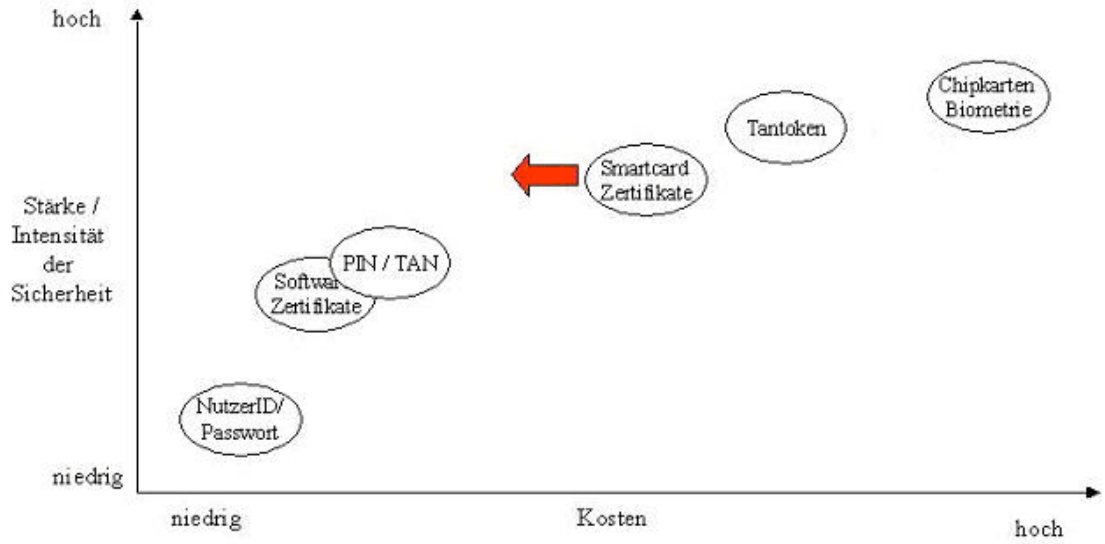

Abbildung 2-6 Skizze zur Einschätzung der Wirksamkeit von Authentifizierungsverfahren (Verändert übernommen aus [BK 00])

Die Abbildung soll den Sprung von Intensität der Sicherheit in Bezug auf die Kosten zwischen Passwortabfrage und Smartcardzertifikaten verdeutlichen. Die Kosten für Smartcardzertifikate werden vermutlich mit stärkerer Verbreitung noch weiter sinken.

Eine angemessene Lösung in einem webbasierten System ist die Authentifizierung über Zertifikate. Diese lassen sich wie schon beschrieben softwareseitig als auch auf Smartcards verwenden. Für die Realisierung des Prototypen werden Softwarezertifikate eingesetzt. Diese lassen sich in die Webbrowser wie Microsoft Internet Explorer 5.5, Opera 6.0 und Netscape Communicator 4.78 und 6.2 integrieren. Die Verwendung von Smartcardzertifikaten erfordert wiederum Lesegeräte bei den Nutzern, sowie Chipkarten und Software zum Portieren der Zertifikate auf die Chipkarten.

Eine Lösung über eine PIN/TAN-Kombination scheint zur Authentifizierung von Nutzern wegen des Verwaltungsaufwandes der TAN-Listen und des sicheren Transfers dieser Listen zum User für eine webbasierte Anwendung ungünstig. Der Nutzer benötigt für jede Anmeldung eine andere TAN. Dies Szenario ist für eine Anmeldung am System

nicht sehr nutzerfreundlich, zur Bestätigung von Transaktionen jedoch sehr geeignet, da jede einzelne Aktion bestätigt wird. Der Transfer der TAN-Listen muss verschlüsselt erfolgen. Dafür sind also wiederum Zertifikate erforderlich.

Für die Nutzung von TAN-Token sind wiederum TAN-Listen und Token Cards erforderlich. Durch die Verwendung von TANs müssen TAN-Listen verwaltet und ausgegeben werden. Aus diesem Grund wird von der Verwendung abgesehen. Eine spätere Implementierung ist nicht ausgeschlossen.

Der Aufwand und die Kosten für die Verwendung biometrischer Verfahren sind in Bezug auf die Sicherheitsanforderung des Systems nicht gerechtfertigt. Ferner ist eine Realisierung als nicht sinnvoll zu betrachten, da die biometrischen Daten der Nutzer erfasst und verwaltet werden müssen. Kaum ein Nutzer würde für eine Anwendung dieser Kategorie seine Daten zur Verfügung stellen.

Eine Integration von Authentifizierungsmechanismen in das System mittels Software Zertifikaten wird aus Kostengründen, Nutzerfreundlichkeit und –akzeptanz befürwortet. Der Prototyp der Implementierung wird die Integration einer Authentifizierung über Softwarezertifikate nutzen.

3 Anwendungsuntersuchung am Beispiel des CAT

3.1 Hintergrundinformation

Das gesamte Projekt get-portal ist durch die zunehmende Liberalisierung der Versorgungsmärkte aus einer rein HTML-basierten Informationsanwendung für den Bereich Strom und Stromversorger entstanden. Hier wurde anfangs die Domäne billiger-strom.de Mitte 1998 mit Informationen, Kontaktadressen und einem Lexikon auf Basis statischer HTML-Seiten ausgestattet.

In weiteren Phasen der Entwicklung wurden Module entwickelt, die es Interessenten ermöglichen, sich in themenbezogene Mailinglisten einzutragen. Über die Site können sich Nutzer zusammenschließen, um gemeinsam höhere Abnahmemengen geltend machen zu können. Aus diesem Grund besteht für die Nutzer die Möglichkeit, sich in Listen einzutragen, sogenannten Poolinglisten[1]. Es gibt die Möglichkeit für EVUs[2], die Stammdaten, Tarife und Ansprechpartner zu administrieren. Weiterhin folgt die Integration eines Tarifrechners in die Site, welcher es Endkunden ermöglicht, nach Eingabe des jährlichen Energieverbrauches und der Postleitzahl des Energieabnahmeortes, eine Vergleichsrechnung zwischen den verschiedenen Versorgern durchzuführen. Auf Basis der berechneten Tarife kann der Endkunde nun weitere Daten über die Energieversorgungsunternehmen (EVU) abrufen oder sich die Vertragsunterlagen für diesen Tarif zusenden lassen. Dies ist das Kernmodul für die Verbindung zwischen EVU und Endkunde, da über diese Online-Vertragsanfrage ein Versorgungsvertrag und somit eine Kundenbeziehung EVU / Endkunde eingeleitet werden kann.

---

[1] Poolinglisten: Pooling (Zusammenschließung) von Personen mit gleichem Interesse, hier : Bildung eines Pools, um durch gemeinsame, größere Abnahmemengen bessere Preise zu erzielen.

[2] EVU: Energieversorgungsunternehmen, ein Unternehmen, welches Kunden mit Energie, sprich Strom, versorgt

1999 wurde auf der bisherigen Basis ein ähnliches System unter der Domäne get-gas.de für den Bereich Gasversorgung aufgebaut. 2000 wurde eine Site für Informationen rund um den Heizölmarkt entwickelt. Auch hier besteht die Möglichkeit, online Heizöl zu bestellen, wodurch wieder eine Kundenbeziehung zwischen Verkäufer und Endkunden aufgebaut werden kann. Die Versorgungsunternehmen aus den Segmenten Gas und Heizöl werden mit Gasversorgungsunternehmen (GVU) und Brennstoffversorgungsunternehmen (BVU) bezeichnet.

Mitte 2000 wurde von der get AG beschlossen, diese Domänen für Endkunden unter der Domäne get-portal.de zu kapseln. Das get-portal sollte für den Endkunden die Verwaltung seiner Stammdaten und der Verträge ermöglichen. Das bedeutet, dass die Daten aus Strom- und Heizölpreisanfragen dort für den Kunden einsehbar und teilweise administrierbar sind. Dieses Vorhaben wurde Anfang 2001 aufgegeben, gegen Ende 2001 jedoch wieder fortgeführt.

Ein elementarer Teil des get-portals betrifft die Administration der Contentinformation in dem System. Mit Content wird der Inhalt von Webseiten bezeichnet. Die einzelnen Contentmodule des get-portals wie Lexika (Gaslexikon, Stromlexikon, Wasser-lexikon,...), Aktuelles, die verschiedenen Tarifrechner für Gas- und Stromverbrauch sowie die Kalkulation von Netznutzungsentgelten sollen in dem Content Administration Tool (CAT) zusammengefasst und von dort zentral verwaltet werden. Ferner werden in diesem administrativen Teilbereich des get-portals auch die Contentkunden, Endkunden, und Verträge über Lieferungen verwaltet werden.

Diese Arbeit beschäftigt sich im Kern mit der sicheren Übertragung der Daten sowie dem Authentifizierungssystem des CAT als Teilsystem des get-portals. Im Folgenden wird abhängig vom Kontext mit get-portal sowohl die Anwendung get-portal an sich als auch die Teilsysteme wie billiger-strom.de, get-gas.de und get-oel.de bezeichnet.

3.2 Basisarchitektur

Die Basisarchitektur bezeichnet die Kombination der Softwarekomponenten Webserver, Applicationserver, Datenbankserver und der Programmierlogik, die zum Betrieb der Anwendung notwendig sind.

3.2.1 Eingesetzte Softwarekomponenten

Der Applicationserver Jakarta Tomcat ist die Referenzimplementierung der Java Server Pages (JSP) und Servlet Spezifikation von Sun Microsystems Inc.. Version 3.3 ist die aktuelle Production Quality Release (PQR) für die Servlet 2.2 und JSP 1.1 Spezifikation.

Tomcat 4 implementiert mit dem neuen Servlet Container Catalina die Servlet 2.3 and JSP 1.2 Spezifikation. Die Architektur des Servlet Containers ist von Grund auf neu entwickelt worden, um eine höhere Performance zu gewährleisten. [JT02]. Die Version 3.2.4 ist die alte PQR und wird nur noch gewartet, aber nicht weiter entwickelt.

Tomcat hat einen eigenen Webserver implementiert, kann aber auch über Module mit dem Internet Information Server von Microsoft oder dem Apache Webserver gekoppelt werden. Für die Verwendung von Tomcat mit dem Apache Webserver existiert das Webservermodul mod_jk, welche von dem Webserver geladen wird und über die Protokolle AJP12 und AJP13 mit dem Applicationserver kommuniziert.

Das Apache Jserv Protocol (AJP) kann von mod_jk in der Version 1.2 (AJP12) und 1.3 (AJP13) genutzt werden. AJP13 überträgt Daten im Gegensatz zu dem Vorgänger AJP12 komprimiert, kann SSL-Informationen des Webservers an Tomcat weiterreichen und soll laut [JT02] performanter sein. AJP setzt auf TCP/IP auf.

Als Webserver kommt der IBM HTTP Server (IHS) zum Einsatz. Dieser ist Teil der Websphere Produktfamilie von IBM. Der IHS basiert auf den Quellen des Open Source Webservers Apache.

3.2.2 Die bisherige Basisarchitektur

Die Anwendung ist webbasiert, d.h. sie läuft auf einem HTTP-Server, der für die Programmlogik Common Gateway Interface (CGI) Skripte auf Basis von Perl nutzt. Die Daten werden in einer MS Access Datenbank gehalten, die von den Perlskripten über Open Database Connectivity (ODBC) angesprochen wird. Der Internet Information Server von Microsoft als HTTP-Server und die Datenbank laufen auf einem gemeinsamen Rechner.

3.2.3 Aktuelle Basisarchitektur

2001 wurde eine neue Architektur eingeführt. Webserver und Datenbank Server werden getrennt. Aus diesem Grund dienen seitdem Rechner mit Intel PIII als separate Datenbankserver. Als Webserver kommen ebenfalls Dual Intel PIII unter IBM http Server 1.3.19 zum Einsatz. Dieser basiert auf den Quellen des Apache Webservers.

Durch die Umstellung der Anwendung von Perl auf Java Servlets und Java Server Pages (JSP) bedingt, wird die Programmlogik teilweise mit Common Gateway Interface (CGI) Skripte mit Active State Perl 5.6 und auch mit Java realisiert. Das Java Software Development Kit (SDK) wird in der Version 1.3.1 von Sun für Microsoft Windows verwendet.

Jakarta Tomcat 3.2.3 wird als Applicationserver für die Java Server Pages verwendet. Über das Webservermodul für den Apache mod_jk werden die Anfragen (requests) vom Webserver über die Protokolle AJP12 und AJP13 an den Applicationserver weitergereicht. Weitere Informationen finden sich unter [WA02] und [JA02].

Zurzeit werden die bereits vorhandenen Perlmodule teilweise in Java reimplementiert. Die älteren Module der Anwendung laufen unter Perl als CGI Skripte, die Neuentwicklungen und auch Umstellungen der alten Module erfolgen in Java. Perl und Java laufen also parallel auf dem Server. Der Datenbankzugriff erfolgt von Perl aus über

Open Database Connectivity (ODBC), von Java aus über die JDBC-ODBC[1] Bridge. Diese wird zeitnah durch den Java Database Connectivity (JDBC) Treiber von Microsoft, ersetzt werden, der seit Mai 2002 verfügbar ist. Dies empfiehlt auch SUN Microsystems.

Als Graphical User Interface (GUI) werden Webbrowser wie der Internet Explorer, Opera und Netscape Communicator eingesetzt.

Die folgende Grafik beschreibt die aktuelle Architektur. Da der IHS auf den Quellen des Apache Webservers beruht und die Architektur im Wesentlichen identisch zum Apache Webserver ist, werden hier für die Architekturbeschreibung die Begriffe Apache API, Apache Core und Apache EAPI verwendet, obwohl es sich um den IHS handelt.

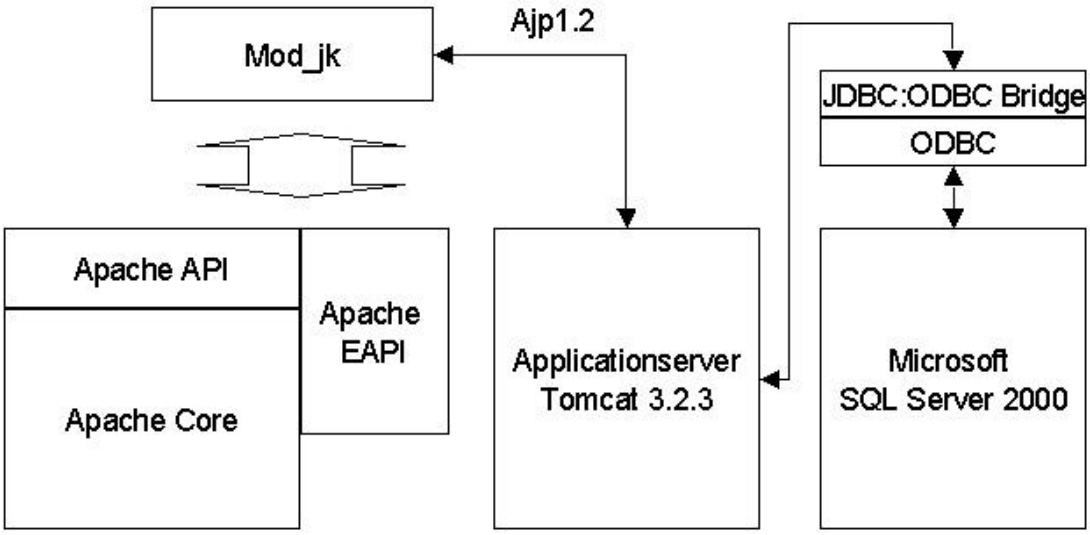

Abbildung 3-1 Die aktuelle Basisarchitektur für die CAT-Anwendung

Das Webserver Modul Mod_jk ist ein Plug-In für den Webserver und leitet die Anfragen, die eine dynamische Erzeugung von HTML Seiten erfordern, an den Applicationserver

[1] JDBC: Java Database Connectivity. Eine Datenbankzugriffsmöglichkeit von SUN. JDBC-ODBC Bridge: Ein javabasierter JDBC Treiber, der JDBC Operationen in ODBC Operationen übersetzt. Siehe http://java.sun.com/j2se/1.3/docs/guide/jdbc/getstart/bridge.doc.html

über das Apache Jserv Protocol 1.2 (Ajp12) weiter. Statische HTML Seiten werden vom IHS direkt behandelt.

Für die Entwicklung der prototypischen Implementierung gelten folgende Rahmenbedingungen:

- Das Datenbanksystem ist ein Microsoft SQL Server 2000
- Als Webserver dient IBM HTTP Server unter Microsoft Windows 2000
- Die Programmlogik basiert auf Java und Perl. Perl soll mittelfristig nicht mehr verwendet werden.
- Die Applikationen läuft auf einem Applicationserver. Zur Zeit wird die Referenzimplementierung der Java Server Pages (JSP) und Servlet Spezifikation Tomcat (Jakarta Tomcat 3.2.3) eingesetzt. Eine Umstellung auf IBM Websphere ist in Betracht zu ziehen.

Für die Anwendung wird der MS SQLServer verwendet, da sich die bereits bestehenden Access Datenbanken relativ problemlos in den Datenbankserver importieren lassen.

3.3 Analyse der Nutzergruppen und Daten in der Anwendung

3.3.1 Nutzergruppen

Das get-portal soll eine Interaktion der verschiedenen Interessengruppen fördern. An der Anwendung partizipieren neben den oben erwähnten Händlern bzw. Versorgern und Endkunden auch Redakteure, die das System administrieren und Kunden, die Content[1] aus dem System beziehen, sogenannte Contentkunden. Bis auf letztere ist für alle anderen Nutzergruppen eine Authentifizierung an der Anwendung erforderlich.

[1] Content: Inhalt in Webseiten, Daten und Informationen, die Dritten zur Verfügung gestellt werden, z.B. ein Online-Lexikon für den Bereich Strom.

3.3.2 Authentifizierung der Nutzer

Die Authentifizierung erfolgt für alle Nutzer über eine Passworteingabe. Bei erfolgreicher Authentifizierung wird dem Nutzer eine Session zur Verfügung gestellt. Hierbei treten zwei Probleme auf. Erstens erfolgt die Authentifizierung und die Autorisierung zusammen in einem Modul pro Teilanwendung. Erweiterungen lassen sich so nur schwer integrieren. Zweitens ist für die Autorisierung der Nutzer für jede Teilanwendung eine eigene Tabelle in dem System integriert. Das heißt, dass die Authentifizierung und Autorisierung zu der zugehörigen Tabelle erfolgen muss. Soll der gleiche Nutzer für zwei Teilanwendungen, zum Beispiel Strom und Gas, freigeschaltet werden, so muss er in beide zu den Teilanwendungen zugehörigen Tabellen eingetragen werden, unter Umständen mit unterschiedlichen Attributen. Diese Mehrfacheintragung schafft eine unnötige Komplexität des Systems.

Die Authentifizierung der Nutzer an dem System erfolgt bisher, wie schon erwähnt, per Nutzername/Passwort Kombination. Dies trifft auf die Redakteure und die Endkunden zu. Im Bereich der Versorger wird zusätzlich eine Kundennummer abgefragt, welche ursprünglich dazu gedacht war, mehrere Nutzer einem Unternehmen zuzuordnen. Dadurch sollte es möglich sein, dass Nutzer verschiedener Unternehmen gleiche Nutzernamen verwenden können, aber eine eindeutige Identifizierung der Nutzer noch möglich ist. Die Kundennummer hat also die Eigenschaft der Gruppierung von Nutzern im Hinblick auf Unternehmen. Der Nutzername ist also keine eindeutige Entität in dem Teilsystem der Versorgerauthentifizierung, wie dies bei den Endkunden und den Administratoren der Fall ist. Dieses Authentifizierungssystem hat sich nicht bewährt, da es aus Nutzersicht umständlich ist, Kundennummer, Nutzername und Passwort anzugeben. Für eine Authentifizierung über Nutzername/Passwort wird ein System angestrebt, welches nutzerfreundlicher ist. Dabei ist zu berücksichtigen, dass die bereits vorhandenen Daten in dem neuen System korrekt interpretiert werden.

3.3.3 Interaktion der Nutzer

Die unterschiedlichen Nutzer wie Endkunden, Versorger und Redakteure können unterschiedliche Aktionen in dem System ausführen. Die Redakteure sind für die Verwaltung und den Content des Systems verantwortlich. Während das System für Endkunden eine Informationsquelle darstellt, mit der Möglichkeit Vertragsunterlagen anzufordern, können Versorger Tarifdaten bereitstellen und über Mailinglisten, Rechner- und Anfragemodule Kundenaquise betreiben. Ein Ziel wird in Zukunft die Abwicklung von Verträgen zwischen Endkunden und Versorgern sein. Hierfür werden suffiziente Sicherheitsmechanismen benötigt.

3.3.4 Daten im System

Das System enthält im Wesentlichen Daten über Händler und Endkunden. Dabei handelt es sich vornehmlich um Adressdaten und Konteninformationen. Eine weitere Rolle spielen Verträge, die im System gehalten werden sollen. Hierbei handelt es sich um Versorgerverträge zwischen Endkunden und Versorgern. Diese sind für den Onlinevertragsabschluss und für die Historie der Kunden vorgesehen.
Weiterhin gibt es tagesaktuelle Informationen sowie aktuelle Nachrichten und fachliche Informationen wie Lexika und Berichte.

Endkunden

Endkundendaten umfassen Adressdaten, sowie Namen und Kontaktdaten wie Telefon, Email und Fax. Auf die Daten wird diskret zugegriffen. Nur der Nutzer selbst hat Zugriff auf seine Daten.

Versorgungsunternehmen

Die Daten der Versorgungsunternehmen setzen sich aus Adressdaten, Kontaktdaten und Tarifdaten zusammen. Es können weitere Informationen bereitgestellt werden. Da diese Daten sowieso für die Öffentlichkeit bestimmt sind, ist eine besonders starke Sicherung des Zugangs nicht erforderlich. Im Wesentlichen muss sichergestellt werden, dass der Zugang nicht von Mitstreitern missbraucht werden kann.

Contentkunden

Contentkunden werden die Daten der get AG wie aktuelle Nachrichten und weitere Informationen wie Tarifrechner und Lexika zur Verfügung gestellt. Sie haben die Möglichkeit, die für sie bestimmte Nachrichten zu ändern oder anzupassen und Statistikinformationen abzufragen.

Redakteure

Redakteure haben in der Applikation weitreichende Rechte. Sie können Contentkunden verwalten, Daten ergänzen, ändern und löschen. Dieser Bereich ist extrem sicherheitssensitiv. Dies betrifft das Content Administration Tool.

3.4 Sicherheitsanalyse der Anwendung

3.4.1 Authentifizierung

Die Authentifizierung der Nutzer geschieht, wie schon beschrieben, über Passworte aus einem HTML-Formular heraus. Die Übertragung der Texte erfolgt bei http im Klartext. Dies gilt auch für die übertragenen Passworte. Ein Eindringling kann sich also durch Mitlesen der übertragenen Pakete Zugang zum System verschaffen.

Die Passworte unterliegen nur der Restriktion, dass leere Passworte verboten sind. Es gibt jedoch keine Vorschriften bezüglich der Mindestlänge und der verwendeten Zeichen. Durch Verwendung von Sonderzeichen, groß- und kleingeschriebenen alphanumerischen, sowie numerischen Zeichen, lässt sich die Sicherheit der eingesetzten Passworte erhöhen. Die Nutzer erhalten bei der Anmeldung ein Passwort zugeteilt, das diesen Vorschriften entspricht. Sie können dies jedoch in ein Passwort ändern, welches diese Restriktionen nicht beachtet. Nach erfolgreicher Anmeldung erhält der Nutzer eine Session zugeteilt. Eine explizite Logoutmöglichkeit besteht nicht.

3.4.2 Autorisierung

Die Anwendung läuft auf mehreren Domänen. Abhängig von den Sicherheitseinstellungen des Browsers dürfen keine Zugriffe über Javascript auf Frames von fremden Domänen erfolgen. Da das bisherige Authentifizierungssystem eine browserseitige Aktivierung von Javascript erfordert, kommt es zu Problemen, wenn die NutzerID aus einem Frame des get-portal in ein Frame der Domäne billiger-strom.de übernommen werden sollen. Für die zukünftige Architektur könnte dieses Problem umgangen werden, wenn die Authentifizierung zentral und ohne Integration von Javascript erfolgt.

Ein weiteres Problem der Applikation ist es, dass die Rechte der Redakteure nicht gesondert spezifizierbar sind. Wird einem Redakteur Zugriff zur Anwendung erteilt, z.B. für die Eingabe der News oder zur Administration des Lexikons, so kann er automatisch sämtliche Module der Anwendung, auch die von anderen Domänen administrieren, da die Authentifizierung mit der Autorisierung identisch ist. Wie schon vorher aufgeführt, wird nur die Nutzername/Passwort Kombination geprüft. Ist diese korrekt, so wird dem Redakteur Zugang gewährt. Bei der Rechtevergabe gibt es nur die Möglichkeit, zwischen „Hat Zugang zum System“ und „Hat keinen Zugang zum System“ zu unterscheiden. Ein filigraneres Rechtesystem wäre in diesem Fall sinnvoll, um den Zugang des Redakteurs auf ein Modul oder eine Domäne zu beschränken.

Ein daraus resultierendes Problem ist die Zuteilung von Administrationsrechten zu Händlern (EVU,GVU,BVU). Es gibt den Fall, dass letztere wöchentliche Berichte über ein spezielles Themengebiet einstellen möchten. An diese Händler muss entweder ein Redakteurlogin vergeben werden, womit wiederum ein uneingeschränkter Zugriff auf alle Domänen möglich wäre, oder aber es wird ein Modul in seinen persönlichen Bereich integriert. Aus der zweiten Variante entsteht ein sehr hoher Wartungsaufwand, da dies eventuell für jeden Händler durchgeführt werden muss. Auch hier ist eine detailliertere Rechtevergabe wünschenswert.

3.4.3 Sessionmanagement

Die Autorisierung der Nutzer an der Anwendung get-portal und der untergeordneten Teilbereiche erfolgt ausschließlich über die NutzerID , welche aus Nutzernamen und Passwort zu Beginn der Session ermittelt wird und einem dieser Session zugewiesenen 32-Byte Sessionkey, dem DCI[1].

Eine Übernahme der Session ist durch das Mitlesen des DCI, z.B. bei der Übertragung, und der Formulierung einer entsprechenden Anfrage an den Server möglich, da dieser bei jeder Anfrage übertragen wird. Eine Übernahme ist durch Neuanmeldung am System nicht möglich, da hier ein neuer DCI vergeben wird. Es wird eine neue Session initialisiert. Bei Anfrage mit einem veralteten DCI wird die Autorisation verweigert.

Ein weiteres Problem ist, dass die Zuweisung des Sessionkeys nicht kollisionsfrei ist. Es ist möglich, wenn auch unwahrscheinlich, dass zwei Nutzer den gleichen Sessionkey zugewiesen bekommen. Aus diesem Grund wurde die NutzerID als Bestandteil der Sessionidentifikation in das Sessiontracking integriert.

3.4.4 Übertragung

Für die Übertragung der Daten kommen zwei Bereiche in Betracht. Der erste Bereich betrifft die Client-Serverkommunikation. Der Browser kommuniziert mit dem HTTP-Server über das HTTP-Protokoll. Dieses überträgt Daten wie Webseiten und Formulare wiederum unverschlüsselt. Diese Kommunikation findet häufig über öffentliche Netze statt.

Der zweite Bereich, der durch eine verschlüsselte Kommunikation abgesichert werden kann, ist die Server-Serverkommunikation in der DMZ[2].

---

[1] DCI: Direct Client Identifier (get AG interner Name). Der DCI ist ein Sessionkey.

[2] DMZ: Demiltitarize Zone, die entmilitarisierte Zone in einer Firewallarchitektur. Die Rechner in dieser Zone sind direkt von einem Netzwerk, z.B. dem Internet aus erreichbar.

Der HTTP-Server erhält einen Request[1], welcher beispielsweise eine JSP[2] anfordert. Da der HTTP-Server diese Anfrage nicht direkt bearbeiten kann, wird diese über ein in den Webserver integriertes Modul an den Servlet-Container weitergereicht. Dieser übernimmt die Abarbeitung der JSP und sendet das Resultat zurück an den HTTP-Server, welcher dieses an den Client weiterreicht. Wird während der Abarbeitung des JSP eine Datenbankanfrage gestartet, so wird zwischenzeitlich noch eine Verbindung zum Datenbankserver aufgebaut. Standard-mäßig laufen diese Kommunikationen unverschlüsselt ab.

3.4.5 Bewertung

Die Sicherheit bei der Authentifizierung erfolgt über Passwortauthentifizierung, welche zwar gängig ist, jedoch sicherheitstechnisch als unterste Stufe zu betrachten ist. Autorisierung und Authentifizierung sind in diesem System nicht unabhängig von einander. Die Übertragung erfolgt unverschlüsselt. Für die Übertragung ist keine Sicherheit vorhanden. Da die Übertragung der Authentifizierungsinformationen nicht verschlüsselt erfolgt, ist auch die Authentifizierung als nicht sicher zu betrachten. Wird aus Sicherheitsgründen Javascript deaktiviert, so ist das System nicht mehr nutzbar. Das Sicherheitssystem der Anwendung kann somit als ausbaufähig angesehen und auf extrem niedrigen Niveau angesiedelt werden.

[1] Request: Eine Anfrage eines Client an einen HTTP-Server

[2] JSP: Java Server Page, entwickelt von SUN Microsystems

4 Entwurf zur Integration moderner Sicherheitstechniken am Beispiel des CAT

Durch die im vorhergehenden beschriebenen Szenarien ist die Nutzungsstrategie weitgehend vorgegeben. Für diese Arbeit interessant ist jedoch die Frage der Nutzungsstrategie hinsichtlich der Authentifizierung und der Sicherheitskategorisierung der Teilsysteme der Anwendung. Es stellt sich die Frage, ob die bisherigen Sicherheitsmechanismen durch eine portalbasierte Nutzeridentifikationsstrategie zur Authentifizierung mittels Zertifikaten erweitert werden können und welche Konsequenzen dies hat.

4.1 Sicherheitsanforderungen an die Anwendung

Für den praktischen Einsatz der Anwendung sollte ein System entworfen werden, das die Privatsphäre der Daten sichert und den Nutzer innerhalb der Anwendung eindeutig identifiziert. Der Nutzer soll sich einmalig pro Sitzung (Session) an dem System authentifizieren und anschließend eindeutig zu identifizieren sein. Die Wahl der Mechanismen der Authentifizierung soll erweiterbar sein, damit auch zukünftige Technologien mit möglichst geringem Aufwand integriert werden können und dem Nutzer verschiedene Authentifizierungsmechanismen zur Verfügung stehen. Diese sind nicht parallel zu betreiben, sondern in Abhängigkeit von der Schutzstufe des beabsichtigten Zugriffes zu sehen. Die Authentifizierungsinformationen sollten verschlüsselt übertragen werden.

Die Autorisierung sollte rollenbasiert ablaufen, so dass Nutzer unterschiedliche Module der Anwendung mit den gleichen Authentifizierungsinformationen nutzen können. Nach Authentifizierung und Autorisierung wird eine Session aufgebaut, die die nötigen Informationen für die Authentifizierung hält. Das Sessionsystem muss auch bei Deaktivierung von Javascript und Cookies funktionieren.

Die Übertragung sollte mindestens über öffentliche Netze verschlüsselt ablaufen. Da Verschlüsselung immer Rechnerressourcen beansprucht, kann die Übertragung in geschützten Bereichen unverschlüsselt ablaufen. Als geschützte Bereiche werden Netzwerke bezeichnet, die durch einen direkten Zugriff von außen geschützt sind. Dies ist bei einer durch eine Firewall geschützte Demilitarized Zone (DMZ) der Fall.

4.2 Authentifizierung

Die Implementierung des Prototypen für das Authentifizierungsszenario soll sowohl eine Einzelfaktorauthentifizierung mittels Passworten als auch eine Doppelfaktorauthentifizierung mit digitalen Zertifikaten umfassen.

4.2.1 Passwortauthentifizierung

Die Authentifizierung erfolgt bisher separat für die Versorgungsunternehmen, die Contentkunden, die Redakteure und die Endkunden über eine Passwortauthentifizierung. Die Passwortauthentifizierung unterliegt dabei keinen Restriktionen bezüglich Länge, Komplexität und Gültigkeitszeitraum.

Endkunden und Versorgungsunternehmen nutzen das System direkt nach einer Onlineregistrierung. Eine Überprüfung der Identität findet nicht statt. Um den Nutzern den Zugang zum System nicht unnötig zu erschweren, muss eine Authentifizierung über bekannte und intuitiv verständliche Szenarien erfolgen. Aus praktischen Gründen ist es zu komplex, für jeden Kunden die Identität zu überprüfen und ein Zertifikat auszustellen. Der Zugriff dieser Nutzer ist aus Sicht der Applikation auf nicht sensible Daten beschränkt. Aus diesem Grund wird eine Einzelfaktorauthentifizierung über Passwörter mit folgenden Restriktionen eingeführt:

1. Ein Passwort muss mindestens 6 Zeichen lang sein.

2. Ein Passwort muss numerische Zeichen, alphanumerische Zeichen und Sonderzeichen enthalten.
3. Bei mehrfacher Fehleingabe erfolgt die Sperrung des Logins.
4. Ein Passwort ist maximal 3 Monate gültig. Anschließend muss es geändert werden oder der Zugang zu diesem Account wird gesperrt.
5. Wird ein Zugang 6 Monate nicht genutzt, wird er inaktiv geschaltet. Der Zugang kann vom Administrator wieder freigeschaltet werden.
6. Der Zugang soll verschlüsselt erfolgen.

Punkt 1 und 2 erhöhen die Komplexität der verwendeten Passwörter und vermindern dadurch die Wahrscheinlichkeit des Erratens der Passwörter. Die Generierung der Authentifizierungs-information mittels eines Brute Force Angriff wird durch die größere Komplexität ebenfalls erschwert.
Punkt 3 unterbindet das Erraten der Authentifizierungsinformation durch Probieren beliebig vieler Passwörter, da nur eine begrenzte Anzahl von Kombinationen probiert werden kann.
Punkt 4 und 5 sind organisatorischer Natur. Dadurch wird die Wahrscheinlichkeit der Übernahme unbenutzter Accounts verringert.
Punkt 6 und 7 unterbinden die Publikation von Authentifizierungsinformation auf dem Übertragungsweg und der Speicherung.

Die Grundlagen für eine passwortbasierte Authentifizierung sind durch Präsentation einer Passwort–Nutzername Kombination und Vergleich derselben mit den Informationen im Datenbanksystem gelegt. Das Szenario ist bereits in das bestehende System integriert. Die Neuerung ist nicht die Methode an sich sondern die Einführung und die Integration der vorher besprochenen Restriktionen.

4.2.2 Zertifikatbasierte Authentifizierung

Bei der zertifikatbasierten Authentifizierung handelt es sich um eine starke Authentifizierungsmethode. Im Folgenden wird untersucht, wie eine zertifikatbasierte Authentifizierung in das System integriert werden kann.

4.2.2.1 Lösungsmöglichkeiten für eine zertifikatbasierte Authentifizierung

Für die Realisierung der zertifikatbasierten Authentifizierung wurden 3 Möglichkeiten in Betracht gezogen.

1. Authentifizierung über Zertifikate in einem Browserapplet
2. Authentifizierung über Zertifikate aus einem Browserplugin
3. Authentifizierung über Softwarezertifikate

Authentifizierung über Zertifikate in einem Browserapplet

Ein Java Applet ist Javacode, der in einer Java Virtual Mashine (JVM) in dem Webbrowser des Nutzers läuft. Der Code unterliegt Restriktionen bezüglich der Ressourcen, auf welche er Zugriff hat. Es entspricht einem Sandkastenmodell. Der Browser gewährt dem Applet abhängig von den Sicherheitseinstellungen Zugriff auf die zur Verfügung stehenden Ressourcen.

Damit ein Applet Authentifizierungsinformationen an einen Server übertragen kann, benötigt es Zugriff auf die Netzwerkübertragungskomponente des Systems. Ebenso muss das Applet Zugriff auf das Zertifikat haben. Bei Softwarezertifikaten liegt dies in Format einer Datei vor. Das Applet benötigt also Lesezugriff auf den Datenträger.

Sind diese Aktionen durch die Sicherheitseinstellungen des Browsers verweigert, funktioniert das Authentifizierungszenario nicht. Da dieses möglichst universell einsetzbar sein und nicht in Abhängigkeit von den Sicherheitseinstellungen der Webbrowser funktionieren soll, wurde dieser Lösungsansatz verworfen.

Authentifizierung über Zertifikate aus einem Browserplugin

Ein Plug-In ist ein Modul, welches die Funktionalität eines Browsers erweitert. Der Browser stellt eine Schnittstelle zur Verfügung, über die das Plug-In mit dem Browser kommuniziert. Im Gegensatz zu einem Applet unterliegt es keinen Restriktionen durch

den Browser bezüglich des Zugriffs auf Ressourcen des Systems. Ein Plug-In läuft mit den Rechten des Browsers.

Netscape stellt ein Software Development Kit zur Entwicklung von Plug-Ins zur Verfügung. Weitere Informationen hierzu befinden sich unter http://developer.netscape.com/docs/manuals/communicator/plugin/index.htm .

Die Entwicklung eines eigenen Plug-Ins wurde in Erwägung gezogen. Da jedoch schon ähnliche Lösungen, wie z.B. Keon Web PassPort, existieren und die Eigenentwicklung eines Plug-Ins für diesen Anwendungsfall keine sichtbaren Vorteile gegenüber einer webserverbasierten Authentifizierung aufwies, wurde der Ansatz nicht realisiert.

RSA Keon Web Passport

Die RSA Keon Web Passport Architektur besteht im Wesentlichen aus zwei Komponenten, dem RSA Keon Web PassPort Plug-In und dem RSA Keon Web PassPort Server. Authentifizierungsinformationen werden von RSA als Credentials bezeichnet.

Die Architektur ist um einen LDAP Verzeichnisdienst designed, in dem die Nutzerdaten, ergänzt um zusätzliche Attribute, verwaltet werden. Weitere Bestandteile der Architektur sind ein Authentication Broker zum Validieren von Nutzer Login Informationen und ein Credential Server zum Speichern der Nutzer Credentials.

Der Web PassPort Server läuft innerhalb des Webservers und dient dazu, Authentifizierungskriterien für den Zugriff auf Webressourcen zu verwalten. Verschiedenen Ressourcen werden über URL Filter identifiziert und können unterschiedlichen Authentifizierungskriterien zugeordnet werden. Der Server evaluiert, welche Credentials für den Zugriff auf die Ressource benötigt werden.

Das Plug-In präsentiert diese Informationen, falls vorhanden. Beim ersten Zugriff auf den Server muss sich der Nutzer an einem sogenannten Authentication Broker

anmelden, um anschließend seine Credentials in Form von Virtual Cards vom Credential Server zu beziehen.

Wird auf den Server zugegriffen und das Keon Web PassPort Plug-In ist noch nicht installiert, wird es automatisch heruntergeladen und installiert.

Das folgende Beispielszenario verdeutlicht den Ablauf.

- Browser: Nutzer lädt Web Site
- URL Filter: Prüft die Authentifizierungs Politik. Wird Authentifizierung benötigt und kein Authentication Cookie gefunden wird die Anforderung an den Authentication Broker weitergeleitet.
- Authentication Broker sendet Login Seite an den Browser
- Browser: Nutzer reagiert auf die geforderte Authentifizierungsmethode
- Authentication Broker validiert die Nutzerinformation
- Ist die Information ok, wird ein Authentication Cookie zurückgegeben.
- Browser: Lädt die Web Site aus 1. erneut.
- URL Filter: Erkennt Authentication Cookie, da aber kein Virtual Card Cookie vorhanden ist, wird die Anfrage an den Credential Server weitergeleitet.
- Credential Server: Sendet Virtual Card Form an den Browser. Die Form enthält ein Objekt eines bestimmten MIME Types speziell für Virtual Cards definiert.
- Ist das Keon Web PassPort Plugin nicht installiert, so wird es installiert.
- Credential Server: Sendet Virtual Card Data aus dem LDAP Verzeichnis an den Browser und setzt ein Virtual Card Cookie.
- Browser: Lädt die Web Site aus 1. erneut.
- URL Filter Erkennt Authentication Cookie und Virtual Card Cookie und erlaubt den Zugriff auf die Ressource.

Abbildung 4-1 Ablauf eines Authentifizierungsszenarios in RSA Keon Web PassPort. (Gekürzt übernommen aus [RS01])

Über dieses Produkt kann also eine Authentifizierung in webbasierten Systemen verwirklicht werden. Es erscheint laut der Dokumentation [RS01] sehr leistungsfähig. Ob das Produkt der Anforderung der untersuchten Anwendung genügt, konnte nicht evaluiert werden, da RSA Security Inc. für dieses Produkt keine Testversion zur Verfügung stellte.

Authentifizierung über Softwarezertifikate

Wie in 2.4.1 beschrieben, kann ein Webserver ein Clientzertifikat anfordern. Der IHS kann für die Clientauthentifizierung unter anderem folgende Parameter für die Einstellung *ClientAuth* verwenden:

1. none
2. required
3. optional

Der Webserver akzeptiert nur Clientzertifikate, die die gleiche Trusted Root CA haben wie eines der Rootzertifikate aus dem Zertifikatspeicher des Webservers. Dadurch kann der Zugriff auf Zertifikate bestimmter CAs beschränkt werden.
Bei der Einstellung *none* wird vom Server kein Zertifikat vom Browser verlangt. Bei *ClientAuth required* muss der Browser ein Zertifikat präsentieren, das von der gleichen Trustet Root CA signiert wurde, um Zugriff zu erlangen. Durch die Einstellung *optional* kann ein Zertifikat vom Browser präsentiert werden. Dies ist jedoch nicht zwingend erforderlich. Der Client kann dann immer noch auf den Server zugreifen.

Eine weitere Möglichkeit der Beschränkung des Zugriffs auf Verzeichnisse des Webservers ist die Auswertung von X.509 Attributen des Clientzertifikates. So kann z.B. der Zugriff auf Zertifikate mit der Organizational Unit (OU) „Redakteur“ beschränkt werden. In diesem Fall dürfen nur Nutzer auf das Verzeichnis zugreifen, deren Zertifikate von der gleichen Zertifizierungsstelle signiert wurden und deren OU Redakteur ist.

Im SSL/TLS Betrieb des Webservers kann durch CGI-Skripte auf diverse Parameter zugegriffen werden, die SSL Informationen enthalten (vgl. 7.5). Diese können nun zur Authentifizierung gegen eine Applikation verwendet werden. Es ist zu beachten, dass es sich in diesem Fall um eine zweistufige Authentifizierung handelt.

Das präsentierte Nutzerzertifikat wird von dem Webserver gegen die Unterzeichnerzertifikate evaluiert. Ist dies erfolgreich, so erhält der Nutzer Zugriff auf den Server und die entsprechenden Variablen werden gesetzt. Wird von dem Webserver nur optional ein Zertifikat verlangt und von dem Nutzer keines präsentiert, so werden die SSL Variablen für die Clientauthentifizierung nicht gesetzt. (vgl. 7.5)

Diese Möglichkeit wird von dem Authentifizierungsmodul benutzt, um eine Authentifizierung über Zertifikate zu realisieren. Hierbei findet die Authentifizierung gegenüber dem Server über das SSL-Handshake des Webservers mit dem Browser statt. Präsentiert der Browser ein gültiges Zertifikat, so werden die SSL-Umgebungsvariablen gesetzt. Diese können nun von serverseitigen Skripten ausgewertet werden. Ein Skript kann so zum Beispiel anhand des vom Client präsentierten DN oder Zertifikates einen Nutzer in der Datenbank identifizieren, um ihm eine Nutzer ID zuordnen. Damit ist der Nutzer per Zertifikat erst gegen den Webserver authentifiziert worden. Zum anderen wird anhand der Umgebungsvariablen ausgewertet, wie die Authentifizierung für die Anwendung erfolgt. Werden die SSL-Variablen für die Clientauthentifizierung gesetzt, so kann gegen die Nutzerdatenbank der Anwendung authentifiziert werden. Ansonsten ist immer noch SSL-Kommunikation mit dem Server möglich, der Nutzer hat sich jedoch gegenüber der Anwendung nicht erfolgreich authentifiziert, kann somit nicht autorisiert werden und erhält keinen Zugriff auf geschützte Bereiche der Anwendung.

Die Authentifizierung geht somit zweistufig vonstatten. Die erste Stufe (der Webserver) prüft die Gültigkeit des Zertifikates. Die zweite Stufe (die Anwendung) prüft, ob der Nutzer berechtigt ist, auf die Anwendung zuzugreifen.

4.2.3 Auswahl der Authentifizierungsmethode

Für die Implementierung des Prototypen wird eine Authentifizierung über Passworte mit den Restriktionen nach 4.2.1 und eine Authentifizierung über Softwarezertifikate angestrebt.

Da die Authentifizierungsmethode möglichst unabhängig von Sicherheitseinstellungen der Browser sein soll, ist die Authentifizierung über Zertifikate durch ein Java Applet

nicht geeignet. Die Realisierung über ein Browser Plug-In wurde wegen der Existenz äquivalenter Produkte und der Möglichkeit der Authentifizierung durch Zertifikate an einem Webserver nicht aufgegriffen. Die Integration von Keon Web PassPort scheiterte an der Möglichkeit der testweisen Realisierung.

4.3 Verwaltung der Zertifikate

Für die Realisierung der Authentifizierung über den Webserver werden Clientzertifikate benötigt. Die verschlüsselte Übertragung per SSL und TLS erfordert Zertifikate für die Webserver. Das Verschlüsseln und Signieren von Emails erfordert ebenfalls Zertifikate.

Für die Anwendung wird eine Zertifizierungsstelle aufgebaut, die Zertifikate für den Betrieb von Webservern mit SSL und TLS, Zertifikate zur Signierung und Verschlüsselung von Emails und Zertifikate zur Authentifizierung gegenüber Webservern unterzeichnen kann. Außerdem kann die Zertifizierungsstelle Zertifikate zum Codesigning signieren.

X.509 konforme Zertifikate lassen sich mit der Open Source Implementierung OpenSSL erzeugen. OpenSSL stellt alle Funktionen zur Verfügung, die zur Erzeugung von Zertifikaten und dem Betrieb einer Zertifizierungsstelle benötigt werden.

4.4 Autorisierung

Die Autorisierung erfolgt durch ein Autorisierungsmodul. Dieses unterstützt sowohl eine rollenbasierte Autorisierung als auch eine „Vererbung“ von Rechten. Das System enthält sowohl Elemente des DAC als auch des Zuständikeitsmodells ([KH95] S.106 ff.).

Architektur

Das Autorisierungsmodul besteht aus Rollen, Rechten und deren Relation zueinander. Einer Rolle sind bestimmte Rechte zugeordnet. Die Nutzer werden wiederum einer oder mehreren Rollen zugeordnet. Außerdem ist es möglich, Nutzern explizite Rechte einzuräumen. Die Rollen, Rechte und Nutzer sowie die Regeln für die Autorisierung werden in einer relationalen Datenbank, dem Microsoft SQL Server, gehalten. Die Rollen können hierarchisch angeordnet werden.

Funktionsweise

Die Autorisierung eines Nutzers für ein Modul der Anwendung erfolgt anhand der NutzerID und des Rechtes selbst. Das Recht wird durch eine Zeichenkette dargestellt. Die NutzerID und das Recht werden einer gespeicherten Prozedur als Parameter mitgegeben.

Die Rollen in dem System können hierarchisch angeordnet werden. Die Besonderheit des System ist die Möglichkeit einer rollenhierarchischen Sperrung von Rechten. Wird einer übergeordneten Rolle ein Recht explizit über Sperrung entzogen, so ist dieses Recht auch für untergeordnete Rollen gesperrt, unabhängig davon, ob das Recht hier gesetzt ist. Dies betrifft nicht die explizit gesetzten Nutzerrechte. Da die Evaluierung dieser vorher erfolgt, hat ein explizit gesetztes bzw. gesperrtes Nutzerrecht Vorrang vor der rollenbasierten Autorisierung.

Die Autorisierung in der Prozedur erfolgt je nach Relationen ein- oder zweistufig. Zuerst wird geprüft, ob eine Relation (Nutzer, Recht) in der Tabelle `User_Rel_Rolle_Recht` existiert. Ist dies der Fall, so wird das Ergebnis aus der Spalte `Recht` zurückgegeben. Die Rückgabewerte sind 1 für „Nutzer hat das Recht“ und 0 für „Nutzer hat das Recht nicht“. Existiert eine Relation, so ist die Autorisierung beendet. Ist keiner der beiden zuvor genannten Fälle eingetreten, existiert das Recht nicht, d.h. es existiert keine Relation (Nutzer, Recht) und die zweite Stufe der Autorisierung wird ausgeführt.

Nun werden die Rollen, in denen sich der Nutzer befindet, auf das entsprechende Recht überprüft. Für die Überprüfung der Rollen müssen alle möglichen Pfade von dem Recht zu der obersten Rolle durchwandert werden, falls kein Pfad gefunden wird, bei dem das Recht auf dem Pfad in keiner der übergeordneten Rollen oder der aktuellen Rolle selbst nicht gesperrt ist und in mindestens einer der übergeordneten Rollen gesetzt ist.

Wird ein Pfad gefunden, bei dem in den übergeordneten Rollen und der Rolle selbst das Recht nicht gesperrt ist, so ist das Recht freigeschaltet und es wird 1 zurückgegeben.

Ist das Recht in einer übergeordneten Rolle gesperrt, so ist das Recht auch in der untergeordneten, aktuellen Rolle gesperrt. Es handelt sich hierbei um eine Negativvererbung. Nur gesperrte Rechte werden vererbt.

4.5 Übertragung

Für die gesicherte Übertragung in einem webbasierten System sind sowohl das TLS V1.0 als auch das SSL V3.0 Protokoll am besten geeignet, da sie erstens speziell für diese Aufgabe entwickelt wurden und zweitens in vielen Webbrowsern integriert sind. Die Verwendung der entsprechenden Protokolle ist abhängig von den Einstellungen der Webbrowser und von der Unterstützung durch den Webserver.

Die protokollspezifische Einstellungen und Parameter für die verschlüsselte Übertragung in Webbrowsern wie Netscape Communicator 4.78, 6.2, Opera 5.12, 6.0 und Internet Explorer 5.5 können konfiguriert werden. So kann angegeben werden, dass für die symmetrische Verschlüsselung unter SSL 3.0 und TLS V1.0 nur 3-DES mit dem asymmetrischen Verfahren RSA und der Hashfunktion SHA für den Schlüsseltausch verwendet werden darf.

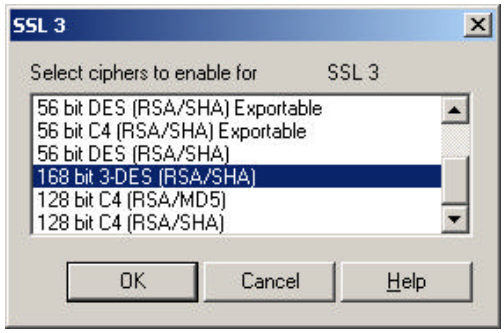

Abbildung 4-2 Einstellung der Verschlüsselungsparameter für SSL3 und TLS V1.0 in Webbrowsern (Opera 5.12)

Protokolle zur Verschlüsselung lassen sich gezielt deaktivieren. Diese werden dann für die Kommunikation nicht verwendet.

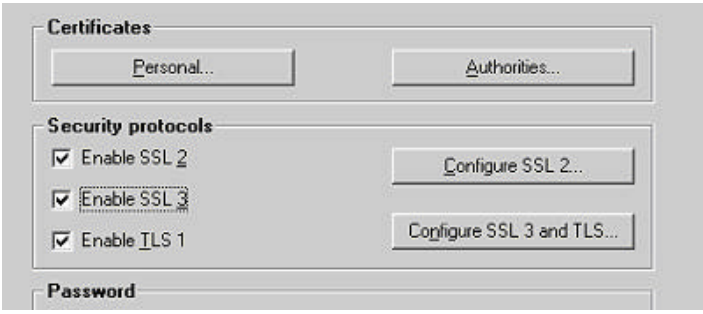

Abbildung 4-3 Einstellung der Übertragungsprotokolle in Opera 6.0

Sinnvoller ist jedoch eine Einstellung der Verschlüsselungsparameter für den Webserver selbst, da sich in diesem Fall nur Clients mit dem Server über HTTPS verbinden können, die diese Parameter unterstützen. SSL V2 kann so zum Beispiel gezielt für die Kommunikation deaktiviert sein, unabhängig davon, ob Clients dies unterstützen oder nicht. Ist SSL V2.0 deaktiviert, so können sich Clients, die nur dieses Protokoll unterstützen, nicht mit dem Server verbinden. Weiterhin können die eingesetzten Verfahren zur Verschlüsselung eingestellt werden. Somit kann die Schlüssellänge von symmetrischen Schlüsseln auf mindestens 128 Bit RC4 festgelegt werden. Die

Verwendung von DES mit 56 Bit kann ausgeschlossen werden. Über die Konfiguration des Webservers lässt sich also eine „Mindestsicherheit“ einstellen.

4.6 Sessionmanagement in dem Prototypen des CAT

Cookies und URL Encoding

Die Authentifizierungsanwendung läuft unter Jakarta Tomcat, und der Applicationserver stellt ein Sessiontracking zur Verfügung. Dafür kommen je nach Browserkonfiguration Cookies oder URL Encoding in Betracht. Die Verwendung von Cookies kann in der Konfiguration des Applicationservers explizit abgeschaltet werden. Dies ist für die Anwendung empfehlenswert, da nach Möglichkeit keine Cookies und kein Javascript verwendet werden sollen.

Für die Anwendung des URL Encoding muss in diesem Fall in jeden Link in den JSPs die SessionID kodiert werden. Wird ein Link nicht kodiert, so wird die Session beim Verfolgen des Links verlassen und die in der Session gespeicherten Daten gehen verloren. URL Encoding muss ebenfalls integriert werden, wenn applicationserverseitig Cookies unterstützt werden, der Browser diese jedoch deaktiviert hat. Eine Möglichkeit, die Session in der Antwort des Servlets (Servlet Response) in die URL zu kodieren beschreibt das folgende Beispiel:

```
<A href='<%=response.encodeURL ("NNE_TarifeingabeNNENNKOL.jsp?Aktion=Anzeigen ")%>' >
bearbeiten </A>
```

Der Wert des HTML-Tags wird durch die Bearbeitung der JSP durch den Servlet Container gesetzt. Bei Ausführung wird die SessionID mit in die URL kodiert. Das Ergebnis sieht beispielsweise folgendermaßen aus:

```
<A    href='NNE_TarifeingabeNNENNKOL.jsp;jsessionid=palj0xq6y1?Aktion=Anzeigen'    >
bearbeiten </A>
```

SSL Sessions

SSL Sessioncaching mindert die Kosten für den SSL Handshake zwischen Client und Server und erlaubt es beiden, mit einem verkürzten Handshake zu kommunizieren. Dies wird von dem IBM HTTP Server unter Windows jedoch nicht unterstützt. Aus diesem Grund kann das Sessioncaching nicht benutzt werden. Jeder Zugriff des Client auf den Server erzeugt somit einen neuen Sessionkey. Ein SSL Sessiontracking nach 2.5.3 ist durch die Verwendung des Webservers damit unter Windows nicht möglich.

5 Designentwurf für den Prototypen des CAT

Für die Authentifizierung über Zertifikate und die Absicherung des SSL/TLS Betriebs der Webserver werden Zertifikate benötigt. Entweder können Zertifikate bei einem Trustcenter beantragt werden oder es wird eine eigene Zertifizierungsstelle aufgebaut, die als vertrauenswürdige dritte Instanz fungiert.

Selbstsignierte Zertifikate werden von dem Besitzer unterzeichnet. Diese Art von Zertifikaten sagt nichts über den Besitzer aus, da in diesem Fall die vertrauenswürdige dritte Instanz fehlt. Ein selbstsigniertes Zertifikat kann zur Verschlüsselung verwendet werden, jedoch ist es für eine Authentifizierung nicht geeignet, da es problemlos möglich ist, selbstsignierte Zertifikate mit beliebigen Identitäten zu erzeugen.

Für die untersuchte Anwendung wurde unter anderem wegen des vorher erwähnten Grundes eine eigene Zertifizierungsstelle aufgebaut. Dies hat seinen Ursprung auch darin, dass mehrere SSL/TLS Serverzertifikate ausgestellt werden müssen. Jeder Nutzer der Anwendung benötigt ebenfalls ein Zertifikat zur Authentifizierung. Die Erweiterungen der Zertifikate lassen sich bei selbständiger Erzeugung der Zertifikate an den Verwendungszweck anpassen. Beim Kauf von Zertifikaten sind deren Erweiterungen weitgehend vorgegeben.

5.1 Die Zertifizierungsstelle

5.1.1 Software

OpenSSL ist eine kryptographische Bibliothek, die eine Menge von kryptographischen Algorithmen kapselt, die in verschiedenen Internetstandards Anwendung finden. Die OpenSSL Bibliothek wird sowohl von SSL, TLS und S/MIME Implementierungen als auch von SSH und OpenPGP genutzt (vgl. http://www.openssl.org).

Für den Aufbau der Zertifizierungsstelle wurde die Version 0.9.6.c von OpenSSL verwendet. Es ist möglich, nur mit OpenSSL eine Zertifizierungsstelle aufzubauen. Dies erwies sich jedoch als nicht sehr komfortabel.

Komfortablere Lösungen bieten die auf OpenSSL aufsetzenden frei verfügbaren Softwarepakete wie OpenCA und pyCA. Diese haben schon Skripte zum Verwalten der Zertifikate, der Zertifizierungsstellen und das Frontend zum Betreiben einer Zertifizierungsstelle integriert.

Für die CA wurde das Paket pyCA verwendet, da OpenCA zu Beginn der Arbeit erst in der Version 0.2 zur Verfügung stand und noch nicht sehr ausgereift war. Die Entwicklung von OpenCA erfolgt jedoch sehr schnell, so dass mittlerweile die Version 0.8.1 verfügbar ist. OpenCA wurde nicht evaluiert, sollte jedoch für künftige Projekte zum Aufbau einer CA in Betracht gezogen werden.

PyCA ist eine auf der Programmiersprache Python basierende CA. PyCA besteht aus einer Sammlung von Python Skripten und benötigt als RA einen Webserver mit CGI Unterstützung. Es lässt sich problemlos installieren. Die Skripte müssen nur an die passende Stelle im System kopiert werden.

Das grundlegende Design der Zertifizierungsstelle erfolgt in der Konfigurationsdatei von OpenSSL, standardmäßig openssl.cnf. Für die Konfiguration der CA erweitert pyCA dabei die Konfigurationsdatei von OpenSSL um eigene Einträge. Diese beinhalten verschiedene Einträge wie die Emailadresse des Administrators oder die URL, an die Registrierungsanträge geschickt werden (vgl. 7.7.2).

5.1.2 Die Struktur der Zertifizierungsstelle

Die Zertifizierungsstelle der get AG, welche die Zertifikate für Authentifizierung und den SSL- Serverbetrieb unterzeichnet, ist mehrstufig. Es existiert eine sogenannte Root CA, die Wurzel der Zertifizierungsstelle. Die Root CA unterschreibt nur die Zertifikate der ihr untergliederten Zertifizierungsstellen. Der Root CA sind in der aktuellen Architektur vier

Zertifizierungsstellen untergeordnet. Diese unterzeichnen alle Zertifikate, die unterschiedlichen Zwecken dienen. Sie sind entweder für die Zertifizierung von Emailadressen, Clientauthentifizierung, Serverzertifizierung oder Codesigning ausgelegt. Die Funktion der jeweiligen Zertifizierungsstelle ergibt sich aus den eingestellten Parametern in der Konfigurationsdateilung der Zertifikate der CA zur Clientauthentifizierung.

Es ergibt sich folgende Struktur für die eingesetzte Zertifizierungsstelle:

Zertifizierungsstelle

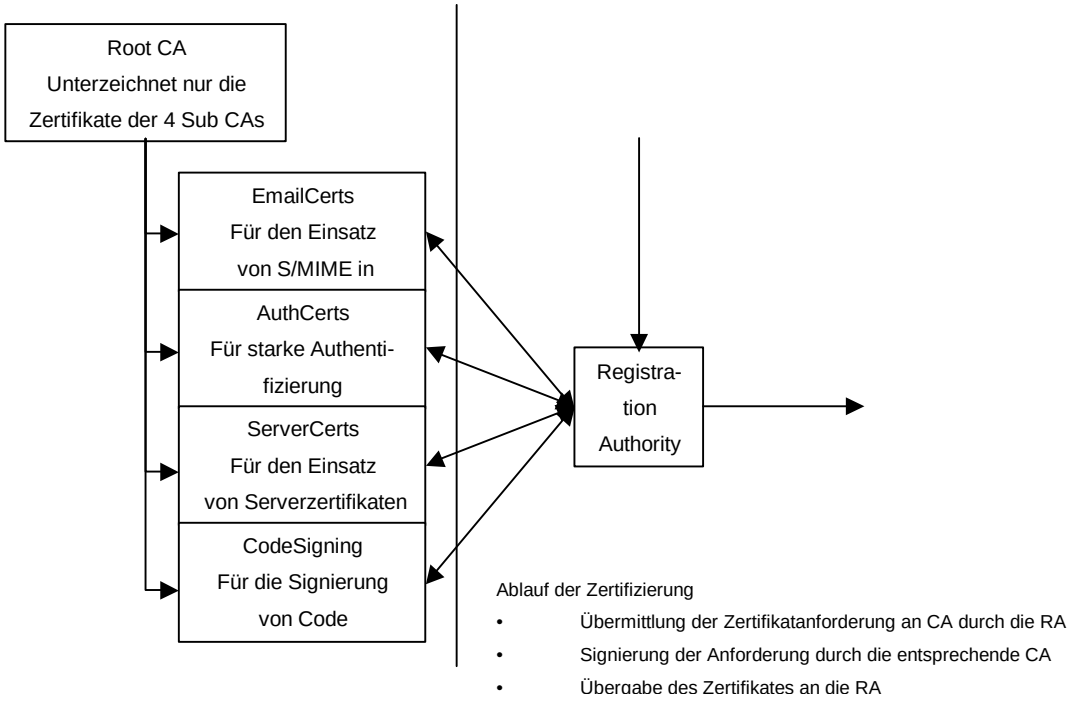

Abbildung 5-1 Struktur der CA der get AG

Die CA EmailCerts signiert Zertifikate für die Verwendung in Emailprogrammen zur verschlüsselten Übertragung und zur Unterzeichnung von Emails. Signierte Zertifikate dieser CA werden von den Redakteuren in Netscape Messenger zur sicheren Kommunikation genutzt.

Zertifikate für die Clienthauthentifizierung von Browsern gegenüber Webservern werden von der CA AuthCerts unterschrieben. Diese werden in dem Prototypen für die Authentifizierung eingesetzt.
Um Webserver mit TLS/SSL Unterstützung zu betreiben, sind Zertifikate nötig, die von der CA ServerCerts unterzeichnet sind. Diese werden in dem Webserver des CAT zur Authentifizierung der Clients eingesetzt.
Die CA für das Codesigning hat zur Zeit für die Anwendung keine Bedeutung. Sie dient zum Signieren von Zertifikaten für die Unterzeichnung von Code.

Bei allen CAs ist die Zertifikaterweiterung `basicConstraint = critical,CA:true` gesetzt, damit Anwendungen, die diese Erweiterung nicht kennen, das Zertifikat zurückweisen. Die Schlüssellängen betragen 1024 Bit. Diese Länge wird heutzutage als ausreichend betrachtet (vgl. [KF98] S. 86).

Die Laufzeit des Zertifikates der Root CA beträgt vier Jahre. Die Laufzeiten von dieser CA signierten Zertifikate müssen alle innerhalb des Laufzeitraumes der CA liegen. Aus diesem Grund signiert die Root CA Zertifikate für 730 Tage. Die Zertifikate der vier CAs sind also 2 Jahre gültig. Deren Signierungen sind jeweils wieder nur für 200 Tage gültig. Danach müssen Zertifikate, die von ihnen signiert wurden, verlängert werden oder verfallen. Nur die Webserverzertifikate sind für ein Jahr gültig.

Die privaten Schlüssel der CA sind nur auf dem CA-Rechner vorhanden und als Sicherung auf Diskette. Der Rechner hat keinen Netzanschluß und steht in einem abgeschlossenen Raum mit Zutrittsbeschränkung. Alle Zertifikatsanforderungen müssen per Diskette auf den CA-Rechner übertragen werden. Die signierten Zertifikate werden ebenfalls per Diskette an die Nutzer rücküberführt. Die Nutzer des CAT stehen mit den Mitarbeitern der get AG in ständigem Kontakt und sind diesen persönlich bekannt. Aus diesem Grund können die Zertifikate nach der Signierung persönlich übergeben werden.

Zertifikate können unterzeichnet und zurückgerufen werden. Die CRLs werden über den Webserver veröffentlicht. Es steht noch kein LDAP Verzeichnis für die CRLs und die Zertifikate zur Verfügung.

Die Basisarchitektur der RA und der CA ist identisch (s. Abbildung 5-2 Basisarchitektur der CA und RA). Das System der RA enthält keine Schlüssel. Beide Systeme basieren auf Linux und verwenden Python und Apache für das Frontend pyCA. OpenSSL wird von der CA zum Signieren der Zertifikate benötigt. Die RA nutzt OpenSSL und das Webserver Modul mod_ssl für den Zugriff auf die RA über SSL/TLS.

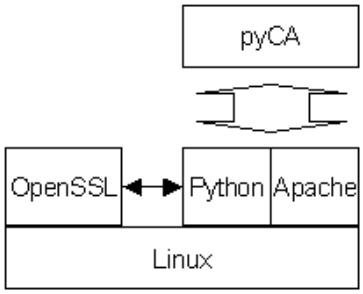

Abbildung 5-2 Basisarchitektur der CA und RA

5.1.3 Einschätzung des Aufbaus der Zertifizierungsstelle

Ein elementarer Teil der Arbeit liegt in dem Design und dem Aufbau der CA, deren Aufwand sich folgendermaßen charakterisiert: Einerseits ist der Aufwand für den Aufbau der Architektur hoch, ferner muss der sichere Betrieb gewährleistet sein, was Personal mit einer relativ hohen Qualifizierung erfordert. Es sollte die Alternative der Nutzung einer externen CA, wie z.B. Verisign oder TC Trustcenter, geprüft werden.

Für die untersuchte Applikation wurde eine eigene CA aufgebaut, da einerseits die Attribute der Zertifikate für die Integration der Authentifizierung in den Prototypen konfigurierbar sein sollten und zum anderen aus Kostengründen, da die Anzahl der Nutzer in Zukunft stark steigen kann, falls dazu übergegangen wird, eine Doppelfakorauthentifizierung auf Basis von Zertifikaten für die Endkunden zu realisieren.

Die Verwendung von Zertifikaten bekannter CAs wie TC Trustcenter und Verisign ist ein Vorteil in Bezug auf die Nutzerfreundlichkeit, da die Rootzertifikate dieser CAs häufig schon in die gängigen Webbrowser wie MS Internet Explorer, Opera und Netscape Communicator integriert sind. Das erspart dem Nutzer die Installation der Zertifikate der Unterzeichner CA. Dies ist sicherlich kein Sicherheitsgewinn, da dem Nutzer dadurch häufig nicht bewusst ist, wem er eigentlich vertraut. Die Einschätzung des Vertrauensverhältnis wird dem Nutzer dadurch abgenommen.

Bei Verbindung mit einem Webserver, der z.B. ein von Verisign unterzeichnetes Zertifikat besitzt, wird der Trust Dialog in einem Browser, der ein Zertifikat der CA von Verisign installiert hat, nicht ausgeführt. Das bedeutet, dass der Nutzer in diesem Fall nicht wirklich realisiert, wem er hier vertraut.

5.2 Die Basisarchitektur der Anwendung

5.2.1 Komponenten der Basisarchitektur

Für die Implementierung des Prototypen ergeben sich einige Änderungen in der Basisarchitektur. Als Webserver wird der IBM HTTP Server in der Version 1.3.19 eingesetzt. Für den Betrieb von SSL V3.0 und TLS V1.0 wird das ibm_ssl_module IBMModuleSSL128.dll, basierend auf der Kryptobibliothek gsk5 von IBM, verwendet. Die Verwaltung der Zertifikate und der Erstellung der Zertifikatanforderungen für den Webserver erfolgt mit dem „IBM Key Management utility“, welches zusammen mit dem ibm_ssl_module und dem Webserver von IBM ausgeliefert wird.

Als Applicationserver wird ein Upgrade auf Jakarta Tomcat auf die Version 4.0.4 aus den weiter unten beschriebenen Gründen durchgeführt. Für die Verbindung zwischen Applicationserver und IHS wird weiterhin das Modul mod_jk verwendet.

Damit ergibt sich für die geänderten Komponenten folgendes Bild:

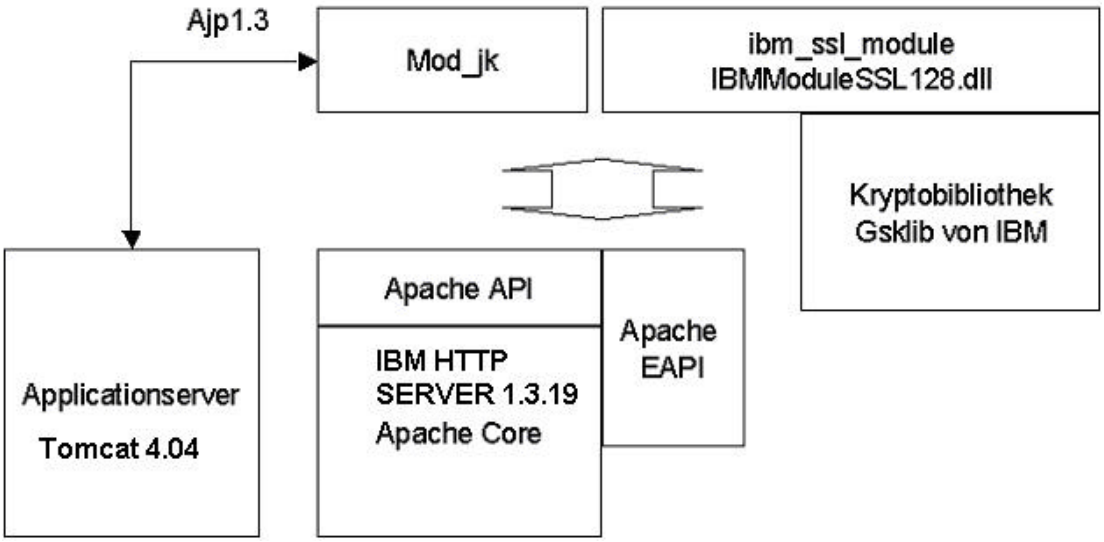

Abbildung 5-3 Neue Basisarchitektur für den Prototypen (vgl. Abbildung 3-1)

5.2.2 Aufgaben des Webservers in der Anwendung

In der neuen Architektur ergeben sich für den Webserver zwei Aufgaben. Zum einen dient er zur Überprüfung der Zertifikate für die zertifikatbasierte Authentifizierung der Nutzer. Zum anderen ist er für die verschlüsselte Übertragung der Daten über nicht öffentliche Netzwerke verantwortlich. Dies ist die erste Stufe in dem zweistufigen Authentifizierungsszenario der Anwendung. Für den Einsatzfall werden die von der aufgebauten Zertifizierungsstelle unterzeichneten Zertifikate eingesetzt.

5.3 Aufbau des Authentifizierungssystems

5.3.1 Authentifizierung im Prototypen

Für das Authentifizierungssystem wird auf die Java Packages `javax.servlet.*` zurückgegriffen. Diese stellen Klassen vom Typ `HttpServletRequest` zur Verfügung. Über diese kann auf die Attribute des Requests von einem Java Servlet zugegriffen werden. Die Pakete stellen auch Klassen vom Typ `HttpSession` bereit, die für das Sessionhandling (vgl. Kapitel 4.6) zuständig sind.

5.3.2 Passwortbasierte Authentifizierung

Die passwortbasierte Authentifizierung erfolgt über ein HTML-Formular, welches den Nutzernamen und das Passwort erfragt, und anschließendes Versenden des Requests an den Webserver. Dieser leitet den Request an die Servlet Engine weiter. Das aufgerufene Servlet reicht die Daten an eine gespeicherte Prozedur in der Datenbank weiter, welche die Daten mit der Nutzerinformation in der Datenbank vergleicht (Tabelle User_Login, vgl. Anhang 7.9). Kann ein Nutzer anhand der Daten authentifiziert werden, wird eine Session kreiert und die Daten aus der Ergebnismenge der gespeicherten Prozedur werden über das Servlet in die Session übernommen und die Antwortseite ausgeliefert. Folgende Codesequenz verdeutlicht den Vorgang der Sessioninitiierung.

```
protected HttpSession session;
session = request.getSession(true);              // Kreieren der Session
session.setAttribute("UserName", getName());     // Übernehmen von Daten
                                                 //  in die Session
```

Damit ist der Nutzer authentifiziert. Die Autorisierung erfolgt nun wie in Kapitel 5.4 beschrieben.

5.3.3 Zertifikatbasierte Authentifizierung

Die zertifikatbasierte Authentifizierung läuft, wie schon beschrieben, zweistufig ab. Erst erfolgt die Authentifizierung für den Zugriff auf den Webserver, anschließend erfolgt die Authentifizierung gegen die Anwendung.

Die SSL Authentifizierungsinformationen werden durch den Webserver nach der Authentifizierung zur Verfügung gestellt (s. Anhang 7.5). Die Applikation benötigt diese Informationen zur Authentifizierung der Nutzer. Aus CGI-Skripten kann auf die Information unter Windows 2000 über Umgebungsvariablen zugegriffen werden. Da Java als eine portierbare Laufzeitumgebung angelegt ist, sollte diese Möglichkeit nicht bestehen. (Es ist auch unter Java in einigen Javaversionen möglich, auf Umgebungsvariablen zuzugreifen, dies wird aber in der Literatur explizit nicht empfohlen ([JK02] S.41).

Da die Applikation auf einem Applicationserver läuft und die SSL-Informationen von dem HTTP-Server bereitgestellt werden, müssen diese an den Applicationserver weitergereicht werden. Dies wird über das Webservermodul mod_jk realisiert. Die SSL-Informationen werden vom IHS mittels mod_jk an den Applicationserver Tomcat übermittelt. Die Kommunikation muss über ajp13 erfolgen. Ajp12 reicht ssl-spezifische Informationen nicht weiter (vgl.).

Die Version Tomcat 3.2.3, die ursprünglich verwendet wurde, unterstützt laut Dokumentation AJP13. AJP12 sollte nicht deaktiviert werden, da es für den shutdown des Application Servers verwendet wird. Es ist nicht gelungen, diese Version von Tomcat mit AJP13 so zu konfigurieren, dass auf die SSL-spezifischen Daten zugegriffen werden konnte. Die Kommunikation zwischen Webserver und Applicationserver funktionierte über AJP12 problemlos, bei Verwendung von AJP13 konnte keine Verbindung zum Applikationserver hergestellt werden.

Aus den in Kapitel 3.2.1 genannten Gründen wird die Version Tomcat 4.0.4 eingesetzt. Diese unterstützt AJP13 und genügt damit den Anforderungen. Nach Anpassung der

Konfigurationsdatei von mod_jk reicht dieses die SSL-Parameter über AJP13 an den Applicationserver weiter.

In Tomcat 4.0.4 wird über ein Objekt vom Typ `HttpServletRequest` auf die Informationen des Request zugegriffen. Der folgende Aufruf ergibt trotz korrekter Konfiguration des Applicationservers null:

```
HttpServletRequest request;
request.getAttribute("javax.servlet.request.cipher-suite");
```

Die Ursache hierfür liegt an der Änderung der Spezifikation selber. Der erste öffentliche Entwurf spezifizierte die Namen als "key-size" und "cipher-suite". Nach der Version Tomcat 4.0-b1 werden "key_size" und "cipher_suite" als Namen verwendet.
Der korrekte Aufruf für die Version 4.04 lautet

```
HttpServletRequest request;
request.getAttribute("javax.servlet.request.cipher_suite");
```

und liefert als Ergebnis für die ausgehandelten Parameter beispielsweise

```
SSL_RSA_WITH_RC4_128_SHA
```

Dies Problem hat Zeit erfordert, da anfänglich nicht feststand, ob es sich um einen Konfigurationsfehler oder Programmierfehler handelte. Es lohnt sich auf alle Fälle, die Spezifikationen zu lesen.

Die folgenden Attribute werden von mod_jk an Tomcat weitergereicht, so dass ein Zugriff über das Servlet möglich ist:

1. `javax.servlet.request.cipher_suite`
2. `javax.servlet.request.ssl_session`
3. `javax.servlet.request.X509Certificate`

Attribut 1 enthält die ausgehandelten Parameter der Verschlüsselung. Attribut 2 gibt Auskunft über die Parameter der SSL Session und über Attribut 3 kann auf das Zertifikat des Clients zugegriffen werden. Seit Tomcat 4 enthält `javax.servlet.request.X509Certificate` kein einzelnes Zertifikat mehr, sondern die Zertifikatkette inklusive des Clientzertifikates, die in einem Array abgelegt ist. Das folgende Codebeispiel greift auf den DN des ersten Zertifikates in der Kette zu:

```
...
X509Certificate myCerts[] = (X509Certificate [])
request.getAttribute("javax.servlet.request.X509Certificate");
if ( myCerts!= null) {
    X509Certificate cert = myCerts[0];
    String DN = cert.getSubjectDN());
}
...
```

Durch den Zugriff auf den DN des Clientzertifikates kann nun über den Vergleich mit dem in der Datenbank gespeicherten DN der Nutzer ermittelt werden. Die so erhaltenen Nutzerdaten werden wie bei der Authentifizierung über Passworte als Attribut in der Session gespeichert.[1]

Auffällig ist bei diesem Authentifizierungsszenario, dass die Zuordnung des Zertifikatinhabers gegenüber der Anwendung über den DN erfolgt. Der Vorgang des Einstellens des DN in die Datenbank ist noch nicht automatisiert. Durch menschliches Versagen können durch eine fehlerhafte Zuordnung Nutzerdaten vertauscht werden.

Das Problem lässt sich durch die Haltung von Nutzerdaten und zugehörigen Zertifikaten in einem Verzeichnisdienst lösen. Werden die Nutzer in einem LDAP Verzeichnis verwaltet, so kann ein Authentifizierungsmodul implementiert werden, welches das Clientzertifikat direkt gegen das Verzeichnis prüft und so den Nutzer ermittelt und authentifiziert.

---

[1] Die kompletten Quelltexte der Klassen befinden sich auf der beigefügten CD.

Eine andere Möglichkeit, dieses Dilemma zu lösen, ist es, neue Nutzer über ihr Zertifikat in der Anwendung zu registrieren. Das Zertifikat enthält bereits detaillierte Informationen über den Nutzer, die verwendet werden können.

5.3.4 Integration der Authentifizierung

Für die Authentifizierung der Nutzer sind Java Klassen entwickelt worden, über die die Authentifizierung über Passworte und softwarebasierte Zertifikate realisiert wird. Die Architektur ist so gehalten, das andere Authentifizierungsmethoden leicht integriert werden können. Die Klasse `ModLogin.java` implementiert die Anmeldung. Alle Authentifizierungsmethoden wie die Passwortauthentifizierung und die zertifikatbasierte Authentifizierung werden hiervon abgeleitet und implementieren den vom Authentifizierungsszenario abhängigen Teil in einer eigenen Klasse. Es ergibt sich die folgende Klassenhierachie:

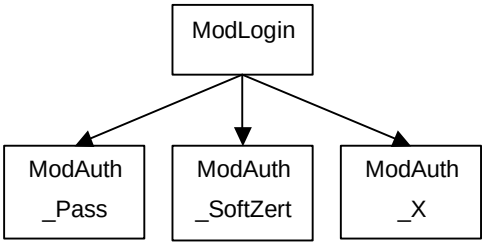

Abbildung 5-4 Klassenhierachie der Authentifizierung[1]

ModLogin beinhaltet Methoden, die von allen Authentifizierungsmodulen, wie ModAuth_SoftZert, benötigt werden oder überschrieben werden müssen.

Das Authentifizierungsmodul hat die Aufgabe, aus den vom Nutzer angegebenen Daten, wie dem Zertifikat, dem Passwort oder anderer Authentifizierungsinformation, den Nutzer in der Datenbank zu identifizieren. Als Eingabe erhält die Authentifizierungs-methode die Daten und als Rückgabe liefert sie die ID des Nutzers aus der Datenbank.

[1] ModAuth_X soll in diesem Zusammenhang die Erweiterbarkeit der Architektur für andere Authentifizierungsmethoden darstellen. Die kompletten Quelltexte der Klassen befinden sich auf der beigefügten CD.

Anhand dieser Nutzer ID werden durch die in der Klasse ModLogin implementierten Methoden benötigte Daten des Nutzers aus der Datenbank ermittelt, eine Session wird kreiert und die ID des Nutzers und die Daten werden als Attribute in die Session geschrieben. Somit ist der Nutzer authentifiziert und hat eine Session erhalten.

Überblick über den Ablauf des Authentifizierungsszenarios:

1. Stufe der Authentifizierung gegen den Webserver.
    a. Der Nutzer fordert die Startseite (JSP) vom Server an.
    b. Dieser verlangt für den Zugriff ein Zertifikat.
    c. Präsentiert der Client ein Zertifikat, wird er gegen über dem Webserver authentifiziert, ansonsten bricht die Kommunikation ab.
2. Stufe der Authentifizierung gegen die Anwendung
    a. Da es sich um eine JSP handelt, wird die Anfrage über mod_jk an den Applicationserver weitergeleitet.
    b. Das Servlet prüft die SSL Parameter gegen die in der Datenbank vorhandenen Informationen und ordnet dem Nutzer die entsprechende NutzerID zu.
    c. Die Session wird kreiert, die Antwortseite wird URLencoded und an den Browser gesendet.
3. Über die in der Session gehaltene NutzerID erfolgt die Autorisierung.

Abbildung 5-5 Ablauf der Authentifizierung

Der Kritikpunkt an diesem Szenario ergibt sich daraus, dass der Server bei der Einstellung *optional* für die Clientauthentifizierung nicht unbedingt ein Clientzertifikat vom Browser fordert. In diesem Fall wird die Übertragung durch SSL/TLS gesichert. Es können jedoch alle nicht anderweitig zugriffsbeschränkten Seiten durch nicht authentifizierte Nutzer abgerufen werden. Dies betrifft sämtliche HTML Seiten und die JSPs, die nicht vor Ausführung die Autorisierung durchführen.

Für die Integration der Verfahren in das Produktionssystem wird die Einstellung eines optional gefordten Clientzertifikates mindestens übergangsweise gelten, bis alle Nutzer mit Zertifikaten ausgestattet sind. Eine weitere Möglichkeit ist es, für die passwortgesteuerte Authentifizierung einen eigenen virtuellen Webserver mit SSL Unterstützung aufzusetzen.

5.4 Integration des Autorisierungsmoduls

Das Autorisierungsmodul ist im Gegensatz zu der Authentifizierung anwendungsabhängig. Aus diesem Grund wurde das Modul datenbankbasiert aufgebaut und in die Anwendung integriert.

Die Autorisierung erfolgt wie in Kapitel 4.4 beschrieben anhand der Übergabe eines Rechtes und der NutzerID aus der Session. Ein Aufruf in einem JSP für die Autorisierung eines Redakteurs zur Eingabe von neuen Nachrichten sieht beispielsweise folgendermaßen aus:

```
<jsp:useBean id="Perm" class="autorisierung.ModAutorize " scope="request">
<jsp:setProperty name="Perm" property="*" />
</jsp:useBean>
<%  if( ! (Perm.isLoggedIn ("Aktuelles_News_Eingabe"))) {%><jsp:forward
page='<%="/LoginFehler.jsp" %>' /><% } %>
```

Das Recht ist `Aktuelles_News_Eingabe` und die NutzerID wird in der Methode `isLoggedIn` aus der aktuellen Session ermittelt. Diese Daten werden im Servlet der gespeicherten Prozedur `checkPermissionUser` übergeben, die nach dem beschriebenen Algorithmus aus Kapitel 4.4 ermittelt, ob der Nutzer das Recht besitzt (Rückgabe 1) oder nicht (Rückgabe 0).

Für die Integration des Autorisierungsmoduls sind die einzelnen Rechte definiert und die Rollen entworfen worden. Die Zuordnung der Rollen zu den Nutzern erfolgt in der Tabelle `User_Rel_User_Rolle.` Die Zuordnung Rolle zu Recht erfolgt in der Tabelle `User_Rel_Rolle_Recht.` Die direkte Zuordnung eines Rechtes zu einem Nutzer geschieht in der Tabelle `User_Rel_User_Recht`. Hiervon sollte jedoch nur in Sonderfällen Gebrauch gemacht werden, da dadurch die rollenbasierte Autorisierung wie in Kapitel 4.4 nicht mehr greift. [1]

[1]Anmerkung: Soll ein Bereich geschützt werden, so ist die Autorisierung für jede JSP in diesem Bereich durchzuführen. HTML Seiten können mit diesem System nicht geschützt werden.

Die folgende Abbildung zeigt einen Teil des Frontends zur Verwaltung der Rollen und Rechte.

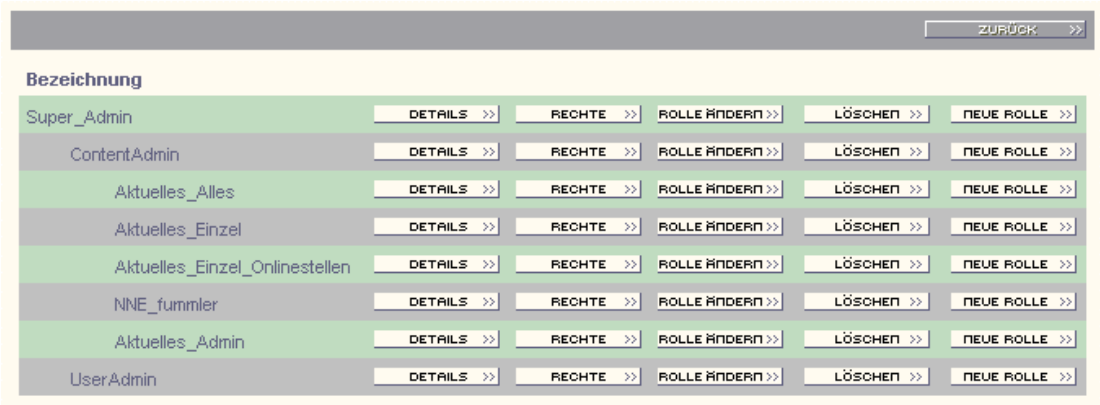

Abbildung 5-6 Frontend für die Rollen- und Rechteverwaltung des CAT

Unterhalb der „Bezeichnung“ sind die Rollen und deren untergeordnete Rollen zu erkennen. Unterhalb von Rollen können wieder neue Rollen angelegt werden.

5.5 Sicherung der Übertragung

Die Übertragung der Daten über die öffentlichen Netze wird mit SSL/TLS gesichert. Der Webserver benötigt dazu, wie schon erwähnt, ein Zertifikat. Die Zertifikatanforderung wird mit dem IBM Keymanagement Tool IKeyman erzeugt, exportiert und von der ServerCerts CA signiert. Das so erstellte Zertifikat wird in die mit Ikeyman verwaltetet Schlüsseldatenbank importiert. Der Webserver hat nun Zugriff auf das Zertifikat, sowie auf den privaten Schlüssel. Damit ist der Betrieb des Servers abhängig von dem verwendeten Browser mit SSL und TLS 1.0 möglich. Somit ist die Übertragung gesichert.

Die Konfiguration der Verschlüsselungsparameter wird an dem Webserver vorgenommen. Die Verwendung von SSL V2 zur Kommunikation wird nicht unterstützt.

Die Authentifizierung gegen den Webserver erfolgt über Softwarezertifikate mittels Clientauthentifizierung. Damit ein Webserver die Authentifizierung der Clients vornehmen kann, müssen die folgenden Zertifikate der get AG CA in der Zertifikatdatenbank des Webservers vorhanden sein:

1. das Root Zertifikat der CA,
2. das Zertifikat für die Server CA,
3. das Zertifikat für die Authentifizierungs CA,

Für den SSL/TLS Betrieb genügen die ersten beiden Zertifikate, da die Zertifikatkette vollständig ist und hiermit eine sichere Verbindung aufgebaut werden kann. Auf Browserseite muss das Rootzertifikat der CA integriert sein, damit er dem Server ohne Browserdialog vertraut. Da die Nutzerzertifikate von der AuthCerts CA unterschrieben werden, muss deren Zertifikat ebenfalls in der Zertifikatdatenbank des Webservers vorhanden sein, damit diese Zertifikatkette vollständig ist und dieser das Zertifikat des Nutzers prüfen kann.

Die Authentifizierung erfolgt nur gegen die in der Schlüsseldatenbank gespeicherten Zertifikate. Werden ausschließlich die Zertifikate der get AG Zertifizierungsstelle integriert und ist die Präsentation eines Clientzertifikates erforderlich, so ist es nur Nutzern gestattet, sich mit dem Server zu verbinden, die über ein von der get AG signiertes Zertifikat verfügen, da der Server andernfalls die Herkunft des Zertifikates nicht überprüfen kann.

Für den Prototypen wurden nur die get AG Zertifikate in die Zertifikatdatenbank des Webservers integriert. Dadurch können sich nur Nutzer, die im Besitz eines von der get AG unterzeichneten Zertifikates sind, anmelden. Der Webserver fordert ein Clientzertifikat. Dadurch ist es anderen Nutzern, die kein entsprechendes Zertifikat

besitzen, nicht möglich, über HTTPS auf den Webserver zuzugreifen und Daten verschlüsselt zu übertragen. Das Clientzertifikat muss zusätzlich von der get AG herausgegeben worden sein, damit eine Verbindung zu dem Server möglich ist. Dies ist in der Konfiguration des Webservers einstellbar. (vgl. Anhang 7.7.1)

Für die Sicherung der Übertragung werden serverseitig die symmetrischen Algorithmen 3DES mit 168 Bit und RC4 mit 128 Bit Schlüsseln verwendet. Kürzere Schlüssel werden nicht zugelassen, da diese als nicht sicher betrachtet werden müssen. Clients, die sich mit kürzeren Schlüsseln verbinden wollen, werden zurückgewiesen. Dies ist beispielsweise bei Browsern der Fall, die den Exportbeschränkungen von Sicherheitstechnologie der USA unterlegen haben. Deren Schlüssellängen für die Symmetrische Verschlüsselung sind auf 40 Bit bzw. 56 Bit beschränkt gewesen.

Als Hashalgorithmen werden MD5 und SHA eingesetzt. Die Länge der asymmetrischen Schlüssel der Serverzertifikate beträgt 1024 Bit. Das IBM Keymanagement Tool, welches für die Verwaltung der Schlüsseldatenbank verwendet wird, kann Zertifikatanforderungen nur mit 512 Bit und 1024 Bit stellen. Es können aber Zertifikate mit größeren Schlüssellängen im PKCS#12-Format importiert werden.

Jede einzelne Domäne, für die eine Übertragung mit SSL/TLS betrieben werden soll, muss wie beschrieben über eine eigene IP-Adresse verfügen. In dem Prototypen ist dies für die CAT-Anwendung realisiert worden. Weitere virtuelle Webserver, wie beispielsweise das get-portal selbst, benötigen für das Login der Endkunden aufgrund der Passwortauthentifizierung ebenfalls eine verschlüsselte Übertragung. Dafür muss wiederum eine eigene IP-Adresse und eine eigene Schlüsseldatenbank für die Verwaltung der Zertifikate und der Schlüssel des Servers eingerichtet werden.

6 Zusammenfassung

Die Arbeit beschäftigt sich mit der Integration von Methoden und Verfahren zur sicheren Übertragung in einem webbasierten System. Es wurden Verfahren zur sicheren Übertragung von Daten über öffentliche Netze und Methoden zur Authentifizierung von Nutzern diskutiert.

6.1 Untersuchung der Implementierung

Die Implementierung erfolgte in den neuen Prototypen des Content Administration Tool der get AG. Die Integration umfasst die sichere Übertragung der Daten über das Secure Socket Layer Protokoll und das Transport Layer Security Protokoll. Dabei wurde besonderer Wert auf die Verwendung der vorher diskutierten Verfahren und Algorithmen gelegt. Für die Verschlüsselung der Übertragung werden nur 3DES (168 Bit) und RC4 mit 128 Bit eingesetzt. Als Hashalgorithmen dienen MD5 und SHA. Die Serverzertifikate und die asymmetrischen Verfahren nutzen Schlüssellängen von 1024 Bit.

In den Prototypen wurde ein rollenbasiertes Autorisierungssystem integriert. Damit ist die Zuteilung von Berechtigungen zu Nutzern und Gruppen möglich. In der vorherigen Architektur durfte ein authentifizierter Nutzer auf den gesamten für diese „Logingruppe“ gültigen Bereich uneingeschränkt zugreifen. Eine Unterteilung der Berechtigungen gab es nur durch die drei Gruppen von Nutzern Endkunden, Redakteur und Händler, die sich in disjunkte Bereiche des Systems einloggten. Diese wurden jeweils gegen eine andere Tabelle in der Datenbank authentifiziert und gleichzeitig autorisiert. Die Authentifizierung und die Autorisierung sind in der neuen Architektur voneinander getrennt.

Für die Authentifizierung an dem System wurde eine erweiterbare Architektur geschaffen, welche bisher eine Authentifizierung über Passworte und softwarebasierte

Zertifikate nach X.509 Standard erlaubt. Die Passwortauthentifizierung wurde um die verschlüsselte Übertragung per SSL/TLS erweitert, so das ein Schutz gegen das Mitlesen des Passwortes während der Übertragung gegeben ist. Außerdem wurden Restriktionen eingeführt, die die Passwörter sicherer machen.

Für eine höhere Sicherheit bei der Authentifizierung wurde die Möglichkeit geschaffen, Nutzer über X.509 konforme Zertifikate an dem System anzumelden. Die Basisarchitektur der Anwendung wurde dahin gehend erweitert, das für die zertifikatbasierte Authentifizierung einige SSL-Parameter der Clientauthentifizierung des Webserver über das Webservermodul mod_jk an den Applicationserver weitergereicht werden und dort im HTTPServletRequest Objekt zur weiteren Verarbeitung zur Verfügung stehen. Die implementierten Klassen nutzen diese Informationen für die Authentifizierung der Nutzer und den Aufbau der Sessions. Die Verfolgung von Session wurde durch URL-encoding realisiert. Dies ist somit unabhängig von Javaskript und Cookies und funktioniert so auch mit starken Sicherheitseinstellungen der Browser, in denen Javaskript und Cookies deaktiviert sind. Dieser Vorteil wurde mit dem Nachteil erkauft, dass in jeden Link der Anwendung die SessionID dynamisch eingebunden werden muss.

Ein Sessionverfolgung über SSL-Sessioncaching konnte nicht realisiert werden, da der verwendete Webserver dieses für Windows nicht unterstützt, sondern nur für Systeme wie Unix oder Linux. Dies ist ein erheblicher Nachteil bei Verwendung von SSL, da so beim SSL-Handshake mehr Overhead entsteht und der Rechner zusätzlich belastet wird.

Die Integration der zertifikatbasierten Authentifizierung und der Autorisierung in den Prototypen des CAT konnte erfolgreich abgeschlossen werden. Somit kann die Integration der Architektur in das Produktionssystem erfolgen.

Für die Verwendung und Erzeugung von Zertifikaten für die Clientauthentifizierung, Emailsignierung, Codesigning und für die Serverzertifizierung wurde eine komplette

Zertifizierungsstelle aufgebaut. Diese Aufgabe ist nicht trivial und erfordert viel Zeit sowohl in der Aufbauphase als auch im Betrieb. Es ist jedoch möglich, mit OpenSSL und der darauf aufsetzenden Software pyCA eine voll funktionsfähige und kostengünstige Zertifizierungsstelle aufzubauen.

Die Zertifizierungsstelle gibt durch die Signierung der Zertifikate als Serverzertifikat oder Authentifizierungszertifikat den jeweiligen Verwendungszweck vor.

Ein wesentlicher Vorteil einer eigenen Zertifizierungsstelle ist die freie Konfigurierbarkeit der Zertifikaterweiterungen. Somit kann der Verwendungszweck der Zertifikate bestimmt werden. Ferner kann die Struktur der Zertifizierungsstelle frei definiert werden, wodurch zum Beispiel eine hierachische Klassifizierung von Zertifikaten erreicht werden kann.

Ein Nachteil einer eigenen Zertifizierungsstelle ist die Tatsache, das sich die Nutzer die Rootzertifikate der Zertifizierungsstelle erst besorgen und installieren müssen. Ansonsten wird der Trust Dialog des Browsers bei jedem Zugriff auf Webserver deren Zertifikate von der eigenen Zertifizierungsstelle signiert wurden, aufgerufen. Dies ist bei Zertifizierungsstellen, deren Zertifikate schon in den gängigen Browsern vorinstalliert sind, nicht der Fall.

6.2 Bewertung

Die Authentifizierung wurde durch die neue Architektur von der Autorisierung entkoppelt. Damit ist die Integration weiterer Authentifizierungsmethoden in die Anwendung wesentlich vereinfacht worden. Das Autorisierungssystem ist ein integraler Bestandteil der Anwendung und benötigt voraussichtlich in naher Zukunft keine Erweiterung.

Durch die Verwendung einer Doppelfaktorauthentifizierung mittels Softwarezertifikaten wurde die Sicherheit des Authentifizierungsverfahrens erhöht, da sich nur Nutzer anmelden können, welche über ein Zertifikat verfügen. Das Anmeldeszenario ist für den

Nutzer unkompliziert, da er nur das Passwort für den Schutz des privaten Schlüssels bzw. der Zertifikatdatenbank eingeben muss. Ist das Zertifikat im Browser installiert, ist dies auch für nichtversierte Nutzer nachvollziehbar und anwendbar. Die zertifikatbasierte Authentifizierung ist somit aus Nutzersicht nicht komplizierter als eine Authentifizierung über Passworte.

Die zertifikatbasierte Authentifizierung wird in Zukunft auf Grund der gesunkenen Kosten für Public Key Infrastrukturen (PKI) und dem damit verbundenen Sicherheitsgewinn voraussichtlich häufiger in Anwendungen eingesetzt werden. Die Anwendung wird damit den zukünftigen Anforderungen an Systeme zur Authentifizierung gerecht.

Die Sicherung der Übertragung entspricht durch den Einsatz von TLS V1.0 und SSL 3.0 und durch die Restriktion der Verwendung von symmetrischen Schlüssel mit Längen von mindestens 128 Bit bei RC 4 und 168 Bit bei 3DES dem heutigen Stand der Technik bei webbasierten Anwendungen. Die Privatsphäre der Daten in einem verteilten Dienstleistungssystem ist damit bei Übertragung ausreichend gesichert.

Die Sicherung der Übertragung innerhalb der DMZ ist untersucht, aber nicht realisiert worden. Eine Möglichkeit der Sicherung der übertragenen Daten ist die Tunnelung der verwendeten Übertragungsprotokolle.

Die Verschlüsselung mit asymmetrischen Schlüsseln von 1024 Bit der Zertifizierungsstelle wird zur Zeit als ausreichend betrachtet, da man nach dem heutigen Kenntnisstand davon ausgehen kann, dass die Berechnung des diskreten Logarithmus innerhalb der nächsten hundert Jahre nicht möglich sein sollte ([KH98] S. 86). Dieser Wert kann aber im Produktionsbetrieb auf 2048 Bit gesetzt werden. OpenSSL unterstützt größere Schlüssellängen.

Eine domänenübergreifende Authentifizierung der Nutzer über Zertifikate ist durch dieses Konzept insoweit möglich, als dass die Authentifizierung in diesem Fall auf dem Applicationserver läuft und der Context für jeden virtuellen Webserver konfiguriert wird.

Die Webserver müssen die SSL-Parameter wie beschrieben an den Applicationserver weiterleiten. Da auf dieselbe Datenbank zugegriffen wird und der Nutzer gegenüber den Webservern das gleiche Zertifikat präsentiert, wird er verständlicherweise auch als der selbe Nutzer identifiziert. Abhängig von den Browsereinstellungen des Nutzers wird er jedoch bei jedem Wechsel von einer Domäne zur anderen Domäne in einem Browserdialog nach dem Passwort für den Schutz seines privaten Schlüssels gefragt. Die Browser können häufig so konfiguriert werden, das der Dialog nur beim ersten Zugriff auf den Schlüssel abgefragt wird.

Für die passwortbasierte Authentifizierung ist dies nicht möglich, da beim Wechsel des Servers die dem Nutzer zugeordnete Session verloren geht. Damit besteht hier nicht die Möglichkeit, den Nutzer an dem anderen Server zu authentifizieren, da die Authentifizierungsinformationen nicht automatisch präsentiert werden, wie dies durch die Anforderung der Clientzertifikate bei der zertifikatbasierten Methode der Fall ist. Der Nutzer bemerkt in diesem Fall nicht zwingend, das ihm eine neue Session zugeordnet worden ist.

6.3 Ausblick

Die Anzahl der Anwendungen, die eine Authentifizierung über Zertifikate unterstützen, wird voraussichtlich in Zukunft zunehmen. Die Anwendung unterstützt zur Zeit nur die Verwendung von Zertifikaten auf Softwarebasis. Die Integration von Zertifikaten zur Authentifizierung auf Basis von Smartcards ist in diesem Zusammenhang ein interessantes Gebiet, das weiter durchleuchtet werden sollte. Es existieren bereits Produkte wie z.B. RSA Keon Web PassPort oder Gemplus eSigner, die unter Umständen für eine Verwendung dieser Art in Betracht gezogen werden können.

Eine weitere Möglichkeit zur Authentifizierung über Zertifikate auf Basis von Smartcards in dem CAT ist die Entwicklung eines Browser Plug-Ins, welches auf den Certificate Request des Browsers nicht das Softwarezertifikat präsentiert, sondern über das Plug-In

auf die Smartcard zugreift, welche Zertifikat und privaten Schlüssel enthält. Da in diesem Fall die Parameter am Webserver unverändert bleiben, kann diese Implementierung voraussichtlich problemlos mit dem Authentifizierungsmodul ModAuth_SoftZert betrieben werden.

Die Authentifizierung der Nutzer erfolgt bei der zertifikatbasierten Variante gegen den DN in der Datenbank. Aus diesem Grund muss der DN der Nutzer immer in der Datenbank eingetragen sein, um für den Vergleich zur Verfügung zu stehen. Für die Zukunft kann die Nutzerverwaltung für die zertifikatbasierte Authentifizierung in ein LDAP Verzeichnis ausgelagert werden.

Da die Zertifikate der Nutzer für den Betrieb einer SigG-konformen Zertifizierungsstelle publiziert werden müssen, kann die Datenhaltung der Nutzer ebenfalls in dem Verzeichnis erfolgen.

Das Terminal Access Controller Access Control System (TACACS, RFC 1492) oder auch der Remote Authentication Dial-In User Service (RADIUS, RFC 2138) bieten die Möglichkeit zur Authentifizierung, Autorisierung und dem Accounting von Nutzern. Die Verwendung von Zertifikaten im Zusammenhang mit diesen Diensten könnte eine interessante Kombination der Verfahren sein.

Da es sich bei dem Authentifizierungssystem des Prototypen um eine erweiterbare Architektur handelt, sollte die Integration weiterer Module für die Authentifizierung gegen Server wie LDAP oder Radius kein Problem darstellen. Die Entwicklung und Integration eines Moduls zur Authentifizierung über smartcardgespeicherte Zertifikate ist somit ebenfalls möglich.

Durch die kryptographischen Berechnungen werden die Webserver zusätzlich belastet. Gerade der IHS unter Windows unterstützt kein SSL-Sessioncaching. Da so bei jeder Verbindung der komplette SSL-Handshake ausgeführt wird, entsteht hier ein hoher Berechnungsaufwand durch die Verschlüsselung. Diese verbraucht wiederum

Systemressourcen. Eine Verminderung der Belastung kann durch Hardwareunterstützung mit Produkten wie „Cryptoserver 2000“ oder der „Crypton 2 SSL Encryption Card“ von Utimaco Safeware erfolgen. Die Karte übernimmt laut Produktbeschreibung alle kryptographischen Berechnungen, die beim SSL-Verfahren notwendig sind, sowohl für die Authentifizierung als auch für die symmetrischen Verschlüsselungsprozesse. Inwieweit dies den Webserver entlastet, ist noch zu untersuchen.

Die Zertifizierungsstelle ist bisher nicht SigG konform. Für den Betrieb einer SigG konformen Zertifizierungsstelle ist ein Verzeichnisdienst für die Verwaltung der ausgegebenen und zurückgerufenen Zertifikate notwendig und die Veröffentlichung der Policy der Zertifizierungsstelle.

Das get-portal bietet die Möglichkeit, Vertragsanfragen an Versorger zu richten. Durch den Einsatz der Zertifizierungsstelle für die Endkunden und Versorger ist die technische Grundlage für das digitale Signieren von Verträgen geschaffen. Die rechtliche Basis zur Eignung der Zertifizierungsstelle für den Abschluss digital signierter Verträge muss noch durchleuchtet werden. Beispielsweise schreibt das SigG für digital signierte Daten vor, dass der private Schlüssel auf Chipkarte gespeichert sein muss, der öffentliche Schlüssel muss in öffentlichem Verzeichnis publiziert werden ([BB99] S. 19.). Die Anwendung muss somit noch für den Einsatz von Chipkarten erweitert werden.

Für die Kommunikation in den geschützten Bereichen (DMZ) zwischen Webserver und Applicationserver sollte untersucht werden, inwieweit das Open Source Modul mod_jk für eine verschlüsselte Übertragung zwischen den Servern erweitert werden kann.

Der Java Authentication and Authorization Service ist ein Dienst, der Authentifizierung und Autorisierung unter Java ermöglicht. JAAS benötigt eine Java2 Umgebung.
Über die Authentifizierung von JAAS kann mittlerweile bestimmt werden, wer zu einem Zeitpunkt den Code laufen lässt. Vorher war es nur möglich zu entscheiden, woher der Code kommt, der laufen soll (Codesigning). JAAS verfügt über Klassen für LoginContexte und Plugable Authentication Modules (PAM). Dadurch wäre JAAS

eventuell für die gestellten Aufgaben zu nutzen gewesen. Die Version von JAAS, die zu Beginn der Arbeit erhältlich war, unterstützte nur Codesigning, d.h. es konnte nur ermittelt werden, wer den Code verteilt oder geschrieben hat, nicht wer ihn laufen lässt. Aus diesem Grund wurde die Möglichkeit einer Authentifizierung und Autorisierung über JAAS nicht weiter untersucht. In Zukunft ist die Entwicklung von JAAS mit Spannung zu verfolgen.

7 Anhang

7.1 Abkürzungsverzeichnis

3DES	Data Encryption Standard
AAA	Authentication Authorization Accounting
AES	Advanced Encryption Standard
AJP12	Apache Jserv Protocol 1.2
AJP13	Apache Jserv Protocol 1.3
BVU	Brennstoffversorgungsunternehmen
C	Country name
CA	Certification Authority
CAT	Content Administration Tool
CGI	Common Gateway Interface
CN	Common Name
CRL	Certificat revocation list
DAC	Discrete Access Control
DES	Data Encryption Standard
DN	Distinguish Name
DSA	Digital Signature Algorithm
DSS	Digital Signature Standard
E	Email address
EVU	Energieversorgungsunternehmen
GUI	Graphical User Interface
GVU	Gasversorgungsunternehmen
HMAC	Hash Message Authentication Code
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IHS	IBM HTTP Server
IDEA	International Data Encryption Algorithm
JDBC	Java Database Connectivity

JSP	Java Server Pages
L	Locality
LDAP	Lightweight Directory Access Protokoll
NIST	National Institute of Standards and Technology
MAC	Message Authentification Code
MD2	Message Digest 2
MD4	Message Digest 4
MD5	Message Digest 5
O	Organization
ODBC	Open Database Connectivity
OU	Organizational unit
PAM	Plugable Authenticatin Module
PIN	Personal Identification Number
PKI	Public Key Infrastructure
PQR	Production Quality Realese
RADIUS	Remote Authentication Dial-In User Service (RFC 2138)
RC2	“Ron's Code" or “Rivest's Cipher" Version 2
RC4	“Ron's Code" or “Rivest's Cipher" Version 4
RC5	“Ron's Code" or “Rivest's Cipher" Version 5
RIPEMD	RACE Integrity Primitives Evaluation Message Digest
RSA	Rivest Shamir Adleman
SDK	Software Development Kit
SHA	Secure Hash Algorithm
SHTTP	Secure Hypertext Transfer Protocol
SQL	Structure Query Language
SSL	Secure Socet Layer
ST	state or province
TACACS	Terminal Access Controller Access Control System (RFC 1492)
TCP	Transport Control Protocol
TAN	Transaction number
TLS	Transport Layer Security

7.2 Abbildungsverzeichnis

7.3 Tabellenverzeichnis

7.4 Literaturverzeichnis

[AS00] DFN-PCA: „Das SSL Apache Handbuch“, Version 1.07, Hamburg 2000

[BB99] F.Bitzer, K. M. Brisch: „Digitale Signatur“, Springer Verlag, Köln 2000

[BK 00] Brokat: http://www.brokat.com/de/netsecurity/authentifizierung/index.htmS, Mai 2000

[CCP1] Common Criteria: “ Part 1: Introduction and general Model”, Version 2.1, August 1999

[CCP2] Common Criteria: “ Part 2:Security functional requirements”, Version 2.1, August 1999

[CCP3] Common Criteria: “ Part 3: Security assurance requirements”, Version 2.1, August 1999

[CE10] Common Criteria: ”Common Methodology for Information Technology Security Evaluation”, Version 1, August 1999

[CT11] Lisa Thalheim, Jan Krissler, Peter-Michael Ziegler: „Körperkontrolle“ , S.114 – S.123, C’T Magazin für Computertechnik, Heft 11, 21.05.2002

[DA99] T. Dierks, C. Allen: „The TLS Protocol Version 1.0“, RFC 2246, Januar 1999

[DS94] George Coulouris, Jean Dollimore, Tim Kindberg: “Distributed Systems – Concepts and Design”, Addison Wesley, 1994

[DU 22] Duden, „Die deutsche Rechtschreibung“, 22. Auflage, Mannheim, Leipzig, Wien, Zürich

[EA01] Arno Erpenbeck: ”Asymmetrische Verschlüsselung”, Universität Osnabrück, 2001

[HC98] D. Harkins, D. Carrel: “The Internet Key Exchange (IKE)”, RFC 2409, November 1998

[HFP99] R. Housley, W. Ford, W. Polk: “Internet X.509 Public Key Infrastructure”, Certificate and CRL Profile , RFC 2459, January 1999

[HW93] Welter, Heinrich: „Heterogene Netze“. Addison Wessley 1993

[HM97] H. Krawczyk, M. Bellare, R. Canetti:”HMAC: Keyed-Hashing for Message Authentication”, RFC2104, February 1997

[KM97] D. Kristol, L. Montulli: ”HTTP State Management Mechanism”, RFC 2109, Bell Laboratories, Lucent Technologies, February 1997

[JA02] Jakarta Apache Website, http://jakarta.apache.org, Juni 2002

[JK 02] Ian F. Darwin,: ”Java Kochbuch”, O’Reilly, deutsche Übersetzung, Beijing, Cambridge, Farnham, Köln, Paris, Sebastopol, Taipei, Tokyo 2002

[JVH95] Josef von Helden: "Verbesserung der Authentifizierung in IT-Sytemen durch spezielle Dienste des Betriebssystems", Verlag Shaker, Aachen, 1995

[JT02] Jakarta Tomcat Website, http://jakarta.apache.org/tomcat/index.html, Juni 2002

[KH95] Kersten, Heinrich: “Sicherheit in der Informationstechnik”. Oldenbourg, München, Wien 1995

[KF98] Kai Fuhrberg: ”Internetsicherheit“, München, Wien, Hanser 1998

[KL00] R. Khare, S. Lawrence: “Upgrading to TLS Within HTTP/1.1”, RFC 2817, Mai 2000

[OC02] OpenCA Dokumentation, http://www.openca.org, 2001

[OS00] DFN-PCA: „Das OpenSSL Handbuch“, Version 2.03, Hamburg 2000

[MA02] Atreya, Mohan: ”Introduction to the PKCS Standards”, http://www.rsa.com

[ML00] Mark Lobel: "The Case for Strong User Authentication", TRS PricewaterhouseCoopers LLP, 2000

[NE98] Netscape Communications Cooperations, http://developer.netscape.com/docs/manuals/security/sslin/index.htm, Introduction to SSL, 1998

[PY02] pyCA-Dokumentation, http://www.pyca.org, 2001

[R00] E. Rescorla: “HTTP Over TLS”, RFC2818, Mai 2000

[RS01] RSA Security Inc.:”RSA Keon Web Passport, Technical Overview”, http://www.rsasecurity.com 2001

[TH97] Thomas Herrmann: "Vergleichende Bewertung von Verfahren zur Benutzerauthentifikation", Shaker Verlag, Aachen,1998

[SX00] Symeon Xenitellis: “The Open-source PKI Book, A guide to PKIs and Open-source implementations”, http://ospkibook.sourceforge.net, 2000

[WA02] Apache Website, http://www.apache.org, Juni 2002

7.5 Umgebungsvariablen des Webservers

Die Umgebungsvariablen des Webservers bei SSL Betrieb können ausgelesen werden und für die Anwendung ausgewertet werden. `SSL_CLIENT_CERTBODY` enthält die Zertifikatdaten und `SSL_CLIENT_SESSIONID` die ID der SSL Session.

Ausgabe der Umgebungsvariablen des Servers bei Verwendung von https und einer Clientauthentifizierung:

```
COMSPEC = C:\WINNT\system32\cmd.exe
DOCUMENT_ROOT = d:/testdomains/content.test.bogus
GATEWAY_INTERFACE = CGI/1.1
HTTPS = ON
HTTPS_CIPHER = SSL_RSA_WITH_RC4_128_SHA
HTTPS_KEYSIZE = 128
HTTPS_SECRETKEYSIZE = 128
HTTP_ACCEPT = text/html, image/png, image/jpeg, image/gif, image/x -xbitmap,
*/*
HTTP_ACCEPT_CHARSET = windows-1252;q=1.0, utf-8;q=1.0, utf-16;q=1.0, iso-8859-
1;q=0.6, *;q=0.1
HTTP_ACCEPT_ENCODING = deflate, gzip, x-gzip, identity, *;q=0
HTTP_ACCEPT_LANGUAGE = en
HTTP_CONNECTION = Keep-Alive, TE
HTTP_HOST = cat.test.bogus
HTTP_IF_MODIFIED_SINCE = Fri, 05 Jul 2002 10:43:36 GMT
HTTP_TE = deflate, gzip, chunked, identity, trailers
HTTP_USER_AGENT = Opera/6.0 (Windows 2000; U) [en]
PATH = C:\Perl\bin\;C:\Program
Files\ibm\gsk5\lib;C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem;C:\Progra
mme\Microsoft SQL Server\80\Tools\BINN
QUERY_STRING = Aktion=printenv
REMOTE_ADDR = 10.0.15.103
REMOTE_PORT = 2943
REQUEST_METHOD = GET
REQUEST_URI = /cgi-bin/domaintools.pl?Aktion=printenv
SCRIPT_FILENAME = d:/testdomains/content.test.bogus/cgi-bin/domaintools.pl
SCRIPT_NAME = /cgi-bin/domaintools.pl
```

```
SERVER_ADDR = 10.0.15.99
SERVER_ADMIN = webmaster@get-ag.com
SERVER_NAME = cat.test.bogus
SERVER_PORT = 443
SERVER_PROTOCOL = HTTP/1.1
SERVER_SIGNATURE =

IBM_HTTP_SERVER/1.3.19 Apache/1.3.20 Server at cat.test.bogus Port 443
SERVER_SOFTWARE = IBM_HTTP_SERVER/1.3.19 Apache/1.3.20 (Win32) mod_jk/1.1.0
SSL_CIPHER = SSL_RSA_WITH_RC4_128_SHA
SSL_CLIENT_C = DE
SSL_CLIENT_CERTBODY =
MIIEUjCCA7ugAwIBAgIBATANBgkqhkiG9w0BAQUFADCBgjELMAkGA1UEBhMCREUxEDAOBgNVBAgTB1
NhY2hzZW4xEDAOBgNVBAcTB0xlaXB6aWcxDzANBgNVBAoTBmdldCBBRzEbMBkGA1UE CxMSZ2V0IEFH
IFRydXN0Y2VudGVyMSEwHwYDVQQDExhnZXQgQUcgQXV0aGVudGljYXRpb24gQ0EwHhcNMDIwNzAzMT
g1MTAzWhcNMDMwMTE5MTg1MTAzWjCBmTELMAkGA1UEBhMCREUxEDAOBgNVBAgTB1NhY2hzZW4xEDAO
BgNVBAcTB0xlaXB6aWcxDzANBgNVBAoTBmdldCBBRzESMBAGA1UECxMJUmVkYWt0ZXVyMRcwFQYDVQ
QDEw5qYW4gZmFsa2VucmVjazEoMCYGCSqGSIb3DQEJARYZamFuLmZhbGtlbnJlY2tAZ2V0LWFnLmNv
bTBcMA0GCSqGSIb3DQEBAQUAA0sAMEgCQQC8xFS9Xk2q1214kk9tj8sZpgv5OB2E2s5P80axCqeFKK
YBylvmiaApmqWOi4zGjvf5lUN9Z+4ouRcg5b3FHhtvAgMBAAGjggIBMIIB/
TALBgNVHQ8EBAMCBsAwEwYDVR0lBAwwCgYIKwYBBQUHAwIwRQY DVR0SBD4wPIY6aHR0cHM6Ly90cnV
zdC5nZXQtYWcuY29tL3B5Y2EvZ2V0LWNlcnQucHkvQXV0aENlcnRzL2NhLmNydDBMBgNVHR8ERTBDM
EGgP6A9hjtodHRwczovL3RydXN0LmdldC1hZy5jb20vcHljYS9nZXQtY2VydC5weS9BdXRoQ2VydHM
vY3JsLmNybDBCBglghkgBhvhCAQ0ENRYzVGhpcyBjZXJ0aWZpY2F0ZSBpcyB1c2VkIGZvciB zdHJvb
mcgYXV0aGVudGljYXRpb24uMCgGCWCGSAGG+EIBAgQbFhlodHRwczovL3RydXN0LmdldC1hZy5jb20
vMDEGCWCGSAGG+EIBBAQkFiJweWNhL2dldC1jZXJ0LnB5L0F1dGhDZXJ0cy9jcmwuY3JsMC4GCWCGS
AGG+EIBAwQhFh9weWNhL25zLWNoZWNrLXJldi5weS9BdXRoQ2VydHM/
MCwGCWCGSAGG+EIBBwQfFh1weWNhL25zLXJlbm V3YWwucHkvQXV0aENlcnRzPzAyBglghkgBhvhCAQ
gEJRYjVGVzdENBL3BvbGljeS9BdXRoQ2VydHMtcG9saWN5Lmh0bWwwEQYJYIZIAYb4QgEBBAQDAgeA
MA0GCSqGSIb3DQEBBQUAA4GBAAxBsVskQAkjcJK1U6e9KBqeILc14PT8VP7HuHze4PjqVwGs0PnmAa
fzxtK72mmD8LcvlDQeEJrSHy4+CYhBWZfvJ2z98+iURtUrLd8rPcEKvjBoxz 90xdYXsyHLs0B9S7vo
FlvZ53Qr+Q3CqsTPzlX6iwyqoMBvd32xkK2ME8Y8AwIwRQYD
SSL_CLIENT_CERTBODYLEN = 1480
SSL_CLIENT_CN = jan falkenreck
SSL_CLIENT_DN = MAIL=jan.falkenreck@get-ag.com,CN=jan
falkenreck,OU=Redakteur,O=get AG,L=Leipzig,ST=Sachsen,C=DE
SSL_CLIENT_EMAIL = jan.falkenreck@get-ag.com
SSL_CLIENT_IC = DE
```

```
SSL_CLIENT_ICN = get AG Authentication CA
SSL_CLIENT_IDN = CN=get AG Authentication CA,OU=get AG Trustcenter,O=get
AG,L=Leipzig,ST=Sachsen,C=DE
SSL_CLIENT_IEMAIL =
SSL_CLIENT_IL = Leipzig
SSL_CLIENT_IO = get AG
SSL_CLIENT_IOU = get AG Trustcenter
SSL_CLIENT_IPC =
SSL_CLIENT_IST = Sachsen
SSL_CLIENT_L = Leipzig
SSL_CLIENT_NEWSESSIONID = TRUE
SSL_CLIENT_O = get AG
SSL_CLIENT_OU = Redakteur
SSL_CLIENT_PC =
SSL_CLIENT_SERIALNUM = 01
SSL_CLIENT_SESSIONID = YAcAAAo AD2dYWFhYWFhYWFhYWFhYWFhYoXglPQsAAAA=
SSL_CLIENT_ST = Sachsen
SSL_PROTOCOL_VERSION = SSLV3
SSL_SERVER_C = DE
SSL_SERVER_CN = cat.test.bogus
SSL_SERVER_DN = CN=cat.test.bogus,OU=contentadministration,O=get
AG,L=Leipzig,ST=Sachsen,C=DE
SSL_SERVER_L = Leipzig
SSL_SERVER_O = get AG
SSL_SERVER_OU = contentadministration
SSL_SERVER_ST = Sachsen
SYSTEMROOT = C:\WINNT
WINDIR = C:\WINNT
```

Listing 7-1 Ausgabe der Umgebungsvariablen des Webservers mit Clientauthentifizierung

Die Ausgabe der Umgebungsvariablen des Servers bei Verwendung von https und ohne Clientauthentifizierung ergibt folgende Liste:

```
COMSPEC = C:\WINNT\system32\cmd.exe
DOCUMENT_ROOT = d:/testdomains/content.test.bogus
GATEWAY_INTERFACE = CGI/1.1
```

```
HTTPS = ON
HTTPS_CIPHER = SSL_RSA_WITH_RC4_128_SHA
HTTPS_KEYSIZE = 128
HTTPS_SECRETKEYSIZE = 128
HTTP_ACCEPT = text/html, image/png, image/jpeg, image/gif, image/x -xbitmap,
*/*
HTTP_ACCEPT_CHARSET = windows-1252;q=1.0, utf-8;q=1.0, utf-16;q=1.0, iso-8859-
1;q=0.6, *;q=0.1
HTTP_ACCEPT_ENCODING = deflate, gzip, x-gzip, identity, *;q=0
HTTP_ACCEPT_LANGUAGE = en
HTTP_CONNECTION = Keep-Alive, TE
HTTP_HOST = cat.test.bogus
HTTP_TE = deflate, gzip, chunked, identity, trailers
HTTP_USER_AGENT = Opera/6.0 (Windows 2000; U ) [en]
PATH = C:\Perl\bin\;C:\Program
Files\ibm\gsk5\lib;C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem;C:\Progra
mme\Microsoft SQL Server\80\Tools\BINN
QUERY_STRING = Aktion=printenv
REMOTE_ADDR = 10.0.15.103
REMOTE_PORT = 2930
REQUEST_METHOD = GET
REQUEST_URI = /cgi-bin/domaintools.pl?Aktion=printenv
SCRIPT_FILENAME = d:/testdomains/content.test.bogus/cgi-bin/domaintools.pl
SCRIPT_NAME = /cgi-bin/domaintools.pl
SERVER_ADDR = 10.0.15.99
SERVER_ADMIN = webmaster@get-ag.com
SERVER_NAME = cat.test.bogus
SERVER_PORT = 443
SERVER_PROTOCOL = HTTP/1.1
SERVER_SIGNATURE =

IBM_HTTP_SERVER/1.3.19 Apache/1.3.20 Server at cat.test.bogus Port 443
SERVER_SOFTWARE = IBM_HTTP_SERVER/1.3.19 Apache/1.3.20 (Win32) mod_jk/1.1.0
SSL_CIPHER = SSL_RSA_WITH_RC4_128_SHA
SSL_CLIENT_NEWSESSIONID = TRUE
SSL_CLIENT_SESSIONID = YAcAAAoAD2dYWFhYWFhYWFhYWFhYVXglPQoAAAA=
SSL_PROTOCOL_VERSION = SSLV3
SSL_SERVER_C = DE
```

```
SSL_SERVER_CN = cat.test.bogus
SSL_SERVER_DN = CN=cat.test.bogus,OU=contentadministration,O=get
AG,L=Leipzig,ST=Sachsen,C=DE
SSL_SERVER_L = Leipzig
SSL_SERVER_O = get AG
SSL_SERVER_OU = contentadministration
SSL_SERVER_ST = Sachsen
SYSTEMROOT = C:\WINNT
WINDIR = C:\WINNT
```

Listing 7-2 Ausgabe der Umgebungsvariablen des Webservers ohne Clientauthentifizierung

Die Ausgabe der Umgebungsvariablen des Servers bei Verwendung von HTTP ohne SSL:

```
COMSPEC = C:\WINNT\system32\cmd.exe
DOCUMENT_ROOT = d:/testdomains/content.test.bogus
GATEWAY_INTERFACE = CGI/1.1
HTTPS = OFF
HTTP_ACCEPT = text/html, image/png, image/jpeg, image/gif, image/x-xbitmap,
*/*
HTTP_ACCEPT_CHARSET = windows-1252;q=1.0, utf-8;q=1.0, utf-16;q=1.0, iso-8859-
1;q=0.6, *;q=0.1
HTTP_ACCEPT_ENCODING = deflate, gzip, x-gzip, identity, *;q=0
HTTP_ACCEPT_LANGUAGE = en
HTTP_CONNECTION = Keep-Alive, TE
HTTP_HOST = cat.test.bogus
HTTP_REFERER = http://cat.test.bogus/cgi-bin/domaintools.pl
HTTP_TE = deflate, gzip, chunked, identity, trailers
HTTP_USER_AGENT = Opera/6.0 (Windows 2000; U) [en]
PATH = C:\Perl\bin\;C:\Program
Files\ibm\gsk5\lib;C:\WINNT\system32;C:\WINNT;C:\WINNT\System32\Wbem;C:\Progra
mme\Microsoft SQL Server\80\Tools\BINN
QUERY_STRING = Aktion=printenv
REMOTE_ADDR = 10.0.15.103
REMOTE_PORT = 2866
REQUEST_METHOD = GET
```

```
REQUEST_URI = /cgi-bin/domaintools.pl?Aktion=printenv
SCRIPT_FILENAME = d:/testdomains/content.test.bogus/cgi-bin/domaintools.pl
SCRIPT_NAME = /cgi-bin/domaintools.pl
SERVER_ADDR = 10.0.15.99
SERVER_ADMIN = webmaster@get-ag.com
SERVER_NAME = cat.test.bogus
SERVER_PORT = 80
SERVER_PROTOCOL = HTTP/1.1
SERVER_SIGNATURE =

IBM_HTTP_SERVER/1.3.19 Apache/1.3.20 Server at cat.test.bogus Port 80
SERVER_SOFTWARE = IBM_HTTP_SERVER/1.3.19 Apache/1.3.20 (Win32) mod_jk/1.1.0
SYSTEMROOT = C:\WINNT
WINDIR = C:\WINNT
```

Listing 7-3 Ausgabe der Umgebungsvariablen bei Verwendung von http

7.6 Installation des IBM HTTP Servers und Konfiguration für SSL/TLS Betrieb

Der IBM HTTP Server (IHS) ist ein Webserver auf Basis des Apache Webservers (http://www.apache.org), einem Projekt der Apache Software Foundation. Die Quellen des Apache Webservers sind Open Source und können geändert oder weiterentwickelt werden. Hier setzt IBM mit dem IHS an. Die Quellen des Apache Webservers dienen IBM als Grundlage für den IBM HTTP Server, der auch um verschiedene Module erweitert wird. Der IHS hat im Gegensatz zum Apache Webserver die folgenden Vorteile:

- Standardmäßig support für SSL secure connections
- Fast Response Cache Accelerator
- IBM support als Teil der WebSphere Produktlinie
- Unterstützung für Kryptohardware
- Administration Server für die Konfiguration des IHS
- IBM bietet verschiedene zusätzliche Module für den IHS an.

Installation

Die benötigten Dateien und Bibliotheken werden automatisch kopiert. Es wird nach einem Dienstkonto gefragt, unter welchem der IHS gestartet werden soll.

Unter Linux ist die Reihenfolge des Einspielens der Pakete zu beachten. Folgende Paktete müssen für den SSL Server eingespielt werden:

1. IBM_turbo_libstdc++_29-1.0-0.i386.rpm
2. IBM_HTTP_Server-1.3.19-0.i386.rpm
3. gsk5bas-5.0-3.61.i386.rpm
4. IBM_SSL_Base-1.3.19-0.i386.rpm
5. IBM_SSL_DE-1.3.19-0.i386.rpm
6. IBM_SSL_128-1.3.19-0.i386.rpm

7. IBM_ADMIN_DE-1.3.19-0.i386.rpm (nur falls der Adminserver betrieben werden soll)

Das Einspielen kann mittels des Befehls rpm –ivh <Paketname> erfolgen, z.B.
rpm –ivh IBM_HTTP_Server-1.3.19-0.i386.rpm

Für die Konfiguration des IHS kann wahlweise die Datei conf/httpd.conf manuell editiert werden oder aber die Konfiguration über ein webbasiertes Interface mittels des Adminserver des IHS angepasst werden. Über diesen kann auch die Konfiguration des SSL/TLS Betrieb gesteuert werden.

Für die Verwaltung der Zertifikate für den SSL Betrieb ist das Tool Ikeyman Bestandteil der Distribution. Dies ist ein Java Programm zur Erstellung einer Zertifikatdatenbank und für die Verwaltung der Zertifikate für den IHS. Es können Zertifikatanforderungen von 512 und 1024 Bit RSA Verschlüsselung gestellt werden. Die unterzeichneten Zertifikate können bei Vorhandensein des Zertifikates der Unterzeichner CA in die Datenbank importiert werden. Das Tool bietet die Möglichkeit, das Passwort für die Zertifikatdatenbank in einer Datei zu verstecken. Das hat den Vorteil, das der SSL-Webserver nach einem shutdown automatisch gestartet werden kann. Andernfalls ist das Passwort durch den Administrator anzugeben.

Die Konfiguration für den Betrieb mit SSL/TLS und zur Clientauthentifizierung kann Anhang 7.7.1 entnommen werden.

Damit ein Client auf die Anwendung zugreifen kann, wird ein Zertifikat zur Clientauthentifizierung benötigt. Der Browser muss die vorgegebenen Verfahren und Schlüssellängen unterstützen. Ansonsten kann nicht auf die Anwendung zugegriffen werden.

7.7 Konfigurationsdateien

Für das reibungslose Zusammenspiel der Basisarchitektur sind einige Konfigurationen notwendig. Im Weiteren werden die wesentlichen Ausschnitte aufgeführt, die während der Arbeit konfiguriert wurden.

7.7.1 Konfiguration des Webservers und des Applicationservers

Der Folgende Konfigurationsdateiabschnitt zeigt die Einstellungen des Webservers bezüglich der Authentifizierung und der Übertragungsparameter. Von dem Webserver wird ein Zertifikat angefordert und dieses muss von der get AG ausgestellt sein. Die Übertragung kann nur über 3DES bzw. RC4 mit 128 bit verschlüsselt ablaufen. Die verwendeten Hashalgorithmen sind MD5 und SHA.

```
# Ausschnitt aus der Webserverkonfigurationsdatei httpd.conf des IBM HTTP
Servers
...
# Laden des SSL Modules
LoadModule ibm_ssl_module modules/IBMModu leSSL128.dll

# ############# Konfiguration #################
# ################## CAT ######################
<VirtualHost cat.test.bogus>
      ServerAdmin    webmaster@get-ag.com
      ServerSignature Off
      ServerName     cat.test.bogus
      ServerAlias    content.test.bogus
      DocumentRoot   d:/testdomains/content.test.bogus
      ServerPath     d:/testdomains/content.test.bogus
      ScriptAlias    /cgi-bin/ "d:/testdomains/content.test.bogus/cgi-bin/"
</VirtualHost>

<VirtualHost cat.test.bogus:443>
        ServerSignature On
        SSLEnable
```

```
        loglevel info

        # Authentifizierung (required, none, optional #2 crl)
        SSLClientAuth require

        # Zertifikat, das für den Server verwendet wird.
        SSLServerCert cat.test.bogus

        # Es dürfen sich nur Clients verbinden, die ein von der
        # get AG ausgestelltes Zertifikat haben.
        SSLClientAuthRequire IssuerOrg = "get AG"

        #SSLClientAuthGroup Org = "get AG"

        # Cipher die unterstützt werden. Unterstützt ein Client diese
        # Ciphern nicht, wird er zurückgewiesen.
        SSLCipherSpec SSL_RSA_WITH_3DES_EDE_CBC_SHA
        SSLCipherSpec SSL_RSA_WITH_RC4_128_MD5
        SSLCipherSpec SSL_RSA_WITH_RC4_128_SHA
      # Andere cipher, die von diesem Server nicht unterstützt werden
      #      SSLCipherRequire SSL_RC2_CBC_128_CBC_EXPORT40_WITH_MD5
      #        SSLCipherSpec SSL_RC2_CBC_128_CBC_EXPORT40_WITH_MD5
      #        SSLCipherSpec SSL_NULL_WITH_NULL_NULL
      #        SSLCipherSpec TLS_RSA_EXPORT1024_WITH_RC4_56_SHA
      #        SSLCipherSpec TLS_RSA_EXPORT1024_WITH_DES_CBC_SHA
      #        SSLCipherSpec SSL_RSA_WITH_DES_CBC_SHA
      #        SSLCipherSpec SSL_RSA_EXPORT_WITH_RC4_40_MD5
      #        SSLCipherSpec SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
      #        SSLCipherSpec SSL_RSA_WITH_NULL_SHA
      #        SSLCipherSpec SSL_RSA_WITH_NULL_MD5

        Keyfile "c:/ddk/cat.test.bogus/cat.kdb"
           ServerAdmin    webmaster@get-ag.com
        DocumentRoot   d:/testdomains/content.test.bogus
        ServerName     cat.test.bogus
        ServerPath     d:/testdomains/content.test.bogus
        ScriptAlias    /cgi-bin/ "d:/testdomains/content.test.bogus/cgi-bin/"
</VirtualHost>
```

```
# ########################## CAT Ende ######################
# ################## SSL/TLS Einstellungen ################
# SSL Einstellungen generell
SSLDisable
SSLV2Timeout 100
SSLV3Timeout 10000

# Festlegen der Ports, auf denen gelauscht werden soll.
# HTTP 80 und HTTPS 443
Listen 443
Listen 80


# ############## Tomcat Verbindung mod_jk #################
include "C:/tomcat404/conf/mod_jk.conf"
...
```

Listing 7-4 Auszug aus der Konfigurationsdatei des Webservers (Abschnitt zur Konfiguration von Authentifizierung und Übertragung)

Wie im hinteren Teil der obigen Webserverkonfigurationsdatei zu sehen, wird die Datei für die Einstellungen des Applicationservers bezüglich der Kommunikation mit dem Webserver geladen. Diese Datei ist folgendermaßen aufgebaut:

```
...
# Section 1
LoadModule jk_module modules/mod_jk.dll

<IfModule mod_jk.c>

# Section 2
JkWorkersFile "c:/tomcat404/conf/workers.properties"
JkLogFile  "c:/tomcat404/logs/jk.log"
JkLogLevel debug

# Verbindung über ajp13
JkMount /*.jsp ajp13

# Section 3
```

```
# Weiterreichung der SSL-Variablen an den Applicationserver
JkExtractSSL On
JkHTTPSIndicator HTTPS
JkSESSIONIndicator SSL_CLIENT_SESSIONID
JkCIPHERIndicator SSL_CIPHER
# What is the indicator for the client SSL certificated (default is =
SSL_CLIENT_CERT)
 JkCERTSIndicator SSL_CLIENT_CERTBODY
</IfModule>
...
```

Listing 7-5 Auszug aus der Konfiguration von mod_jk

Section 1 in Abbildung 7-5 läd das Modul mod_jk. In Section 2 werden mod_jk die Einstellungen für die Workerprozesse bekannt gemacht, und dem Webserver mitgeteilt, das er alle Anforderungen, die JSPs betreffen, an den Applicationserver zur Bearbeitung weiterleitet. In Section 3 werden die Werte für die Übergabe der SSL Parameter eingestellt. Dies ist nötig, da die Variablen von SSL Implementierung zu SSL Implementierung. Variieren. Mod_jk wurde ursprünglich für den Apache Webserver geschrieben, auf dessen Quellen der IHS beruht. Für den Apache Webserver gibt es unter anderen mod_ssl und apache_ssl als SSL Implementierungen, welche unterschiedliche Bezeichner für das Clientzertifikat haben. Auch der IHS weicht mit der Bezeichnung `SSL_CLIENT_CERTBODY` von der Bezeichnung `SSL_CLIENT_CERT` bei `mod_SSL` ab.

In der Konfigurationsdatei des Applicationservers Tomcat muss abschließend noch der Kontext für die Anwendung eingerichtet werden.

7.7.2 Konfiguration und Struktur der Zertifizierungsstelle

Die CA besteht aus der Root CA, welche ausschliesslich Zertifikate der ihr untergeordneten CAs unterzeichnet. Nutzer- und Serverzertifikate werden von der Root

CA nicht unterzeichnet. Diese werden von den diesem Zweck zugeordneten CAs unterzeichnet. Die Datei zeigt den Aufbau der CA anhand der Konfigurationsdatei von OpenSSL. Da diese Datei sehr groß ist, wurde sie nicht abgedruckt und ist auf der beigefügte CD enthalten.

7.8 X509 Zertifikate

Im Folgenden ist ein X.509 Zertifikat, das zur Verwendung von Clientauthentifizierung gegenüber Webservern dient, zu sehen. Die beiden Darstellungen zeigen einerseits das Zertifikat im PEM-Format und andererseits in Textausgabe.

```
-----BEGIN CERTIFICATE-----
MIIE2zCCBESgAwIBAgIBAzANBgkqhkiG9w0BAQUFADCBgjELMAkGA1UEBhMCREUx
EDAOBgNVBAgTB1NhY2hzZW4xEDAOBgNVBAcTB0xlaXB6aWcxDzANBgNVBAoTBmdl
dCBBRzEbMBkGA1UECxMSZ2V0IEFHIFRydXN0Y2VudGVyMSEwHwYDVQQDExhnZXQg
QUcgQXV0aGVudGljYXRpb24gQ0EwHhcNMDIwNjIxMTQ1MzE3WhcNMDMwMTA3MTQ1
MzE3WjCBljELMAkGA1UEBhMCREUxEDAOBgNVBAgTB1NhY2hzZW4xEDAOBgNVBAcT
B0xlaXB6aWcxDzANBgNVBAoTBmdldCBBRzEPMA0GA1UECxMGRGV2dGVjMRcwFQYD
VQQDEw5qYW4gZmFsa2VucmVjazEoMCYGCSqGSIb3DQEJARYZamFuLmZhbGtlbnJl
Y2tAZ2V0LWFnLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANDQ
p/fTH5UiKPe2iOEW39r3zsQl3qBSj1oz7dsJo5CEM1c/jhw6lIZ2knWpWxEu4Ipk
0+I2txTtCVndoTNpmedHVoHoz+ptEzX1bzb2dJiJnxNn+NWWMTHNX5+BkYIl3zSf
uXdLIKUBIACDqx0QdGc8ISKU9VDWGdzzDPZlltmkEB2cQ+VGsNs4bEl3AgqL5Hk5
pMTglmWNAWb8qqkRpZjcfnszHHQ5VHCJR+Yr/8oiz3XEgudQYn7Iv95Grz8TQoVN
sK3C4kkkU++Km6lzHleM+k9sFx6VLpZLpXBCVDQtaK7iqOe7bjchWM+jEaelWfp4
LH3mRxFh+ipY3CugG9MCAwEAAaOCAcUwggHBMAkGA1UdEwQCMAAwCwYDVR0PBAQD
AgXgMBMGA1UdJQQMMAoGCCsGAQUFBwMCMEwGA1UdHwRFMEMwQaA/oD2GO2h0dHBz
Oi8vdHJ1c3QuZ2V0LWFnLmNvbS9weWNhL2dldC1jZXJ0LnB5L0F1dGhDZXJ0cy9j
cmwuY3JsMEIGCWCGSAGG+EIBDQQ1FjNUaGlzIGNlcnRpZmljYXRlIGlzIHVzZWQg
Zm9yIHN0cm9uZyBhdXRoZW50aWNhdGlvbi4wKAYJYIZIAYb4QgECBBsWGWh0dHBz
Oi8vdHJ1c3QuZ2V0LWFnLmNvbS8wMQYJYIZIAYb4QgEEBCQWInB5Y2EvZ2V0LWNl
cnQucHkvQXV0aENlcnRzL2NybC5jcmwwLgYJYIZIAYb4QgEDBCEWH3B5Y2EvbnMt
Y2hlY2stcmV2LnB5L0F1dGhDZXJ0cz8wLAYJYIZIAYb4QgEHBB8WHXB5Y2EvbnMt
cmVuZXdhbC5weS9BdXRoQ2VydHM/MDIGCWCGSAGG+EIBCAQlFiNUZXN0Q0EvcG9s
aWN5L0F1dGhDZXJ0cy1wb2xpY3kuaHRtbDARBglghkgBhvhCAQEEBAMCB4AwDQYJ
KoZIhvcNAQEFBQADgYEASvjdyE44q3AnnUB+HR03mXg2Rc7z0Iz+qUpi/MTBuvgD
gVBEErxX3gB1biooQz5UDKVYVCmcsEyWIYSCzKh2+PRr1US8DLC8P69sE/mLwT+t
f/niv1AFoQslzUW035x+EQsKAqCSzpBC1XL/zabKi2PHI0ZeaAK3jsu54KRCvJg=
-----END CERTIFICATE-----
```

Abbildung 7-6 Ein X.509 Zertifikat im PEM-Format

Die nächste Abbildung zeigt das Zertifikat in Klartextdarstellung.

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 3 (0x3)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=DE, ST=Sachsen, L=Leipzig, O=get AG, OU=get  AG Trustcenter, CN=get AG
Authentication CA
        Validity
            Not Before: Jun 21 14:53:17 2002 GMT
            Not After : Jan  7 14:53:17 2003 GMT
        Subject: C=DE, ST=Sachsen, L=Leipzig, O=get AG, OU=Devtec, CN=jan
falkenreck/Email=jan.falkenreck@get-ag.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:d0:d0:a7:f7:d3:1f:95:22:28:f7:b6:88:e1:16:
                    df:da:f7:ce:c4:25:de:a0:52:8f:5a:33:ed:db:09:
                    a3:90:84:33:57:3f:8e:1c:3a:94:86:76:92:75:a9:
                    5b:11:2e:e0:8a:64:d3:e2:36:b7:14:ed:09:59:dd:
                    a1:33:69:99:e7:47:56:81:e8:cf:ea:6d:13:35:f5:
                    6f:36:f6:74:98:89:9f:13:67:f8:d5:96:31:31:cd:
                    5f:9f:81:91:82:25:df:34:9f:b9:77:4b:20:a5:01:
                    20:00:83:ab:1d:10:74:67:3c:21:22:94:f5:50:d6:
                    19:dc:f3:0c:f6:65:96:d9:a4:10:1d:9c:43:e5:46:
                    b0:db:38:6c:49:77:02:0a:8b:e4:79:39:a4:c4:e0:
                    96:65:8d:01:66:fc:aa:a9:11:a5:98:dc:7e:7b:33:
                    1c:74:39:54:70:89:47:e6:2b:ff:ca:22:cf:75:c4:
                    82:e7:50:62:7e:c8:bf:de:46:af:3f:13:42:85:4d:
                    b0:ad:c2:e2:49:24:53:ef:8a:9b:a9:73:1e:57:8c:
                    fa:4f:6c:17:1e:95:2e:96:4b:a5:70:42:54:34:2d:
                    68:ae:e2:a8:e7:bb:6e:37:21:58:cf:a3:11:a7:a5:
                    59:fa:78:2c:7d:e6:47:11:61:fa:2a:58:dc:2b:a0:
                    1b:d3
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            X509v3 Key Usage:
                Digital Signature, Non Repudiation, Key Encipherment
            X509v3 Extended Key Usage:
                TLS Web Client Authentication
            X509v3 CRL Distribution Points:
                URI:https://trust.get-ag.com/pyca/get-cert.py/AuthCerts/crl.crl
```

```
            Netscape Comment:
                This certificate is used for strong authentication.
            Netscape Base Url:
                https://trust.get-ag.com/
            Netscape CA Revocation Url:
                pyca/get-cert.py/AuthCerts/crl.crl
            Netscape Revocation Url:
                pyca/ns-check-rev.py/AuthCerts?
            Netscape Renewal Url:
                pyca/ns-renewal.py/AuthCerts?
            Netscape CA Policy Url:
                TestCA/policy/AuthCerts-policy.html
            Netscape Cert Type:
                SSL Client
    Signature Algorithm: sha1WithRSAEncryption
        4a:f8:dd:c8:4e:38:ab:70:27:9d:40:7e:1d:1d:37:99:78:36:
        45:ce:f3:d0:8c:fe:a9:4a:62:fc:c4:c1:ba:f8:03:81:50:44:
        12:bc:57:de:00:75:6e:2a:28:43:3e:54:0c:a5:58:54:29:9c:
        b0:4c:96:21:84:82:cc:a8:76:f8:f4:6b:d5:44:bc:0c:b0:bc:
        3f:af:6c:13:f9:8b:c1:3f:ad:7f:f9:e2:bf:50:05:a1:0b:25:
        cd:45:b4:df:9c:7e:11:0b:0a:02:a0:92:ce:90:42:d5:72:ff:
        cd:a6:ca:8b:63:c7:23:46:5e:68:02:b7:8e:cb:b9:e0:a4:42:
        bc:98
-----BEGIN CERTIFICATE-----
MIIE2zCCBESgAwIBAgIBAzANBgkqhkiG9w0BAQUFADCBgjELMAkGA1UEBhMCREUx
EDAOBgNVBAgTB1NhY2hzZW4xEDAOBgNVBAcTB0xlaXB6aWcxDzANBgNVBAoTBmdl
dCBBRzEbMBkGA1UECxMSZ2V0IEFHIFRydXN0Y2VudGVyMSEwHwYDVQQDExhnZX Qg
QUcgQXV0aGVudGljYXRpb24gQ0EwHhcNMDIwNjIxMTQ1MzE3WhcNMDMwMTA3MTQ1
MzE3WjCBljELMAkGA1UEBhMCREUxEDAOBgNVBAgTB1NhY2hzZW4xEDAOBgNVBAcT
B0xlaXB6aWcxDzANBgNVBAoTBmdldCBBRzEPMA0GA1UECxMGRGV2dGVjMRcwFQYD
VQQDEw5qYW4gZmFsa2VucmVjazEoMCYGCSqGSIb3DQEJARYZamFuLmZhbG tlbnJl
Y2tAZ2V0LWFnLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANDQ
p/fTH5UiKPe2iOEW39r3zsQl3qBSj1oz7dsJo5CEM1c/jhw6lIZ2knWpWxEu4Ipk
0+I2txTtCVndoTNpmedHVoHoz+ptEzX1bzb2dJiJnxNn+NWWMTHNX5+BkYIl3zSf
uXdLIKUBIACDqx0QdGc8ISKU9VDWGdzzDPZlltmkEB2cQ+VGsNs4bE l3AgqL5Hk5
pMTglmWNAWb8qqkRpZjcfnszHHQ5VHCJR+Yr/8oiz3XEgudQYn7Iv95Grz8TQoVN
sK3C4kkkU++Km6lzHleM+k9sFx6VLpZLpXBCVDQtaK7iqOe7bjchWM+jEaelWfp4
LH3mRxFh+ipY3CugG9MCAwEAAaOCAcUwggHBMAkGA1UdEwQCMAAwCwYDVR0PBAQD
AgXgMBMGA1UdJQQMMAoGCCsGAQUFBwMCMEwGA1UdHwRFMEMwQa A/oD2GO2h0dHBz
Oi8vdHJ1c3QuZ2V0LWFnLmNvbS9weWNhL2ldC1jZXJ0LnB5L0F1dGhDZXJ0cy9j
cmwuY3JsMEIGCWCGSAGG+EIBDQQ1FjNUaGlzIGNlcnRpZmljYXRlIGlzIHVzZWQg
Zm9yIHN0cm9uZyBhdXRoZW50aWNhdGlvbi4wKAYJYIZIAYb4QgECBBsWGWh0dHBz
Oi8vdHJ1c3QuZ2V0LWFnLmNvbS8wMQYJYIZIAYb4QgEEBC QWInB5Y2EvZ2V0LWNl
cnQucHkvQXV0aENlcnRzL2NybC5jcmwwLgYJYIZIAYb4QgEDBCEWH3B5Y2EvbnMt
Y2hlY2stcmV2LnB5L0F1dGhDZXJ0cz8wLAYJYIZIAYb4QgEHBB8WHXB5Y2EvbnMt
cmVuZXdhbC5weS9BdXRoQ2VydHM/MDIGCWCGSAGG+EIBCAQlFiNUZXN0Q0EvcG9s
aWN5L0F1dGhDZXJ0cy1wb2xpY3kuaHRtbDARBglghk gBhvhCAQEEBAMCB4AwDQYJ
KoZIhvcNAQEFBQADgYEASvjdyE44q3AnnUB+HR03mXg2Rc7z0Iz+qUpi/MTBuvgD
```

```
gVBEErxX3gB1biooQz5UDKVYVCmcsEyWIYSCzKh2+PRr1US8DLC8P69sE/mLwT+t
f/niv1AFoQslzUW035x+EQsKAqCSzpBC1XL/zabKi2PHI0ZeaAK3jsu54KRCvJg=
-----END CERTIFICATE-----
```

Abbildung 7-7 Ein X.509 Zertifikat in Klartextausgabe

Der Block von “----BEGIN CERTIFICATE-----“ bis „-----END CERTIFICATE-----“ beinhaltet das komplette Zertifikat im PEM-Format. (vgl. Tabelle 2-3 Formate von Zertifikaten und typische Endungen).

7.9 Datenstruktur für die Authentifizierung und Autorisierung

7.9.1 ER-Diagramm

Das folgende ER-Diagramm verdeutlicht die Zusammenhänge zwischen Authentifizierung und Autorisierung. Der linke Teil des Diagramms stellt die Authentifizierung, der Rechte die Autorisierung dar. Die Verbindung erfolgt über die Entitäten User_Stammdaten.

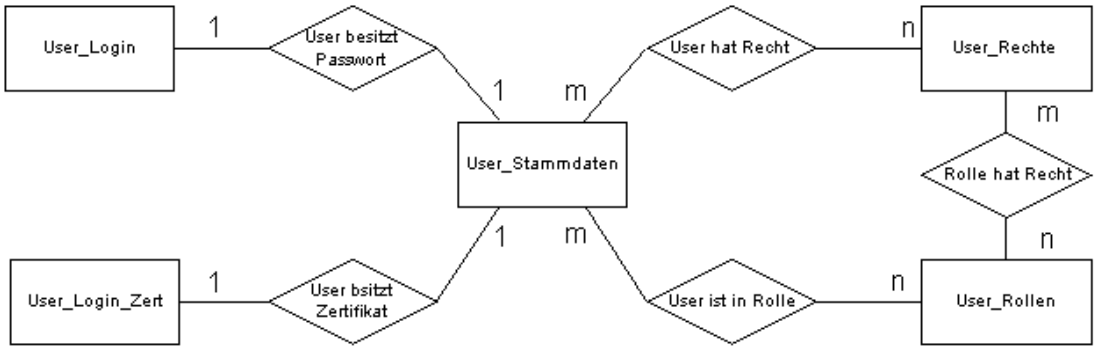

Abbildung 7-8 ER-Diagramm für die Authentifizierung und Autorisierung

7.9.2 Implementierung

Die Implementierung des ER-Diagramms wird durch die Diagrammansicht der Tabellenstruktur des MS SQL Servers deutlich.

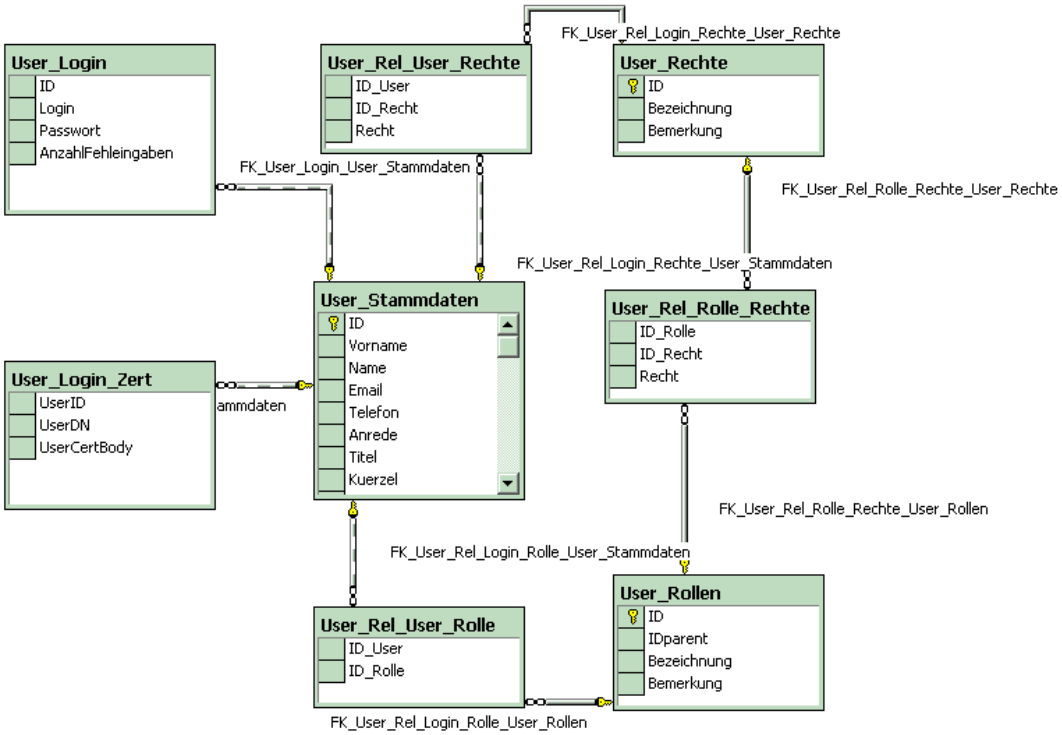

Abbildung 7-9 Implementierung der Tabellenstruktur für die Authentifizierungs- und Autorisierungsverfahren

7.10 Übersicht über die PKCS Standards

Standard	Description
PKCS # 1	The RSA encryption standard. This standard defines mechanisms for encrypting and signing data using the RSA public key system.
PKCS # 3	The Diffie-Hellman key-agreement standard. This defines the Diffie-Hellman keyagreement protocol.
PKCS # 5	The password-based encryption standard (PBE). This describes a method to generate a Secret Key based on a password.
PKCS # 6	The extended-certificate syntax standard. This is currently being phased out in favor of X509 v3.
PKCS # 7	The cryptographic message syntax standard. This defines a generic syntax for messages which have cryptography applied to it.
PKCS # 8	The private-key-information syntax standard. This defines a method to store Private Key Information.
PKCS # 9	This defines selected attribute types for use in other PKCS standards.
PKCS # 10	The certification request syntax standard. This describes a syntax for certification requests.
PKCS # 11	The cryptographic token interface standard. This defines a technology independent programming interface for cryptographic devices such as smartcards.
PKCS # 12	The personal information exchange syntax standard. This describes a portable format for storage and transportation of user private keys, certificates, etc.
PKCS # 13	The elliptic curve cryptographic standard. This describes mechanisms to encrypt and sign data using elliptic curve cryptography.
PKCS # 14	This covers pseudo random number generation (PRNG). This is currently under active development.
PKCS # 15	The cryptographic token information format standard. This describes a standard for the format of cryptographic credentials stored on cryptographic tokens.

NOTE: PKCS # 2 and # 4 do not exist anymore because they have been incorporated into PKCS # 1

Tabelle 7-1 Übersicht über die PKCS (aus [MA02])

7.11 Bemerkungen

Browser/Software

Aufgrund der früher herrschenden Exportbeschränkungen der USA für Kryptoprodukte mit Schlüssellängen über 40 Bit RSA konnte manche Software keine größeren Schlüssellängen erzeugen. In diesem Fall muss der Schlüssel extern erzeugt werden z.B. mit OpenSSL und anschließend in die Software integriert werden, wenn diese das unterstützt.

IBM IKeyman

Wird eine Zertifikatanforderung durch die CA signiert, so ist darauf zu achten, dass für die Verwendung des Zertifikates der Zeitraum des Signierens nicht vor dem Gültigkeitszeitraum des Zertifikates liegt. Signiert die CA die Anforderung nach Greenwich Mean Time (GMT) und der Server, der die Anforderung in Mid European Time (MET) stellt, so kann das Zertifikat nach der Signierung nicht sofort verwendet werden. Das Zertifikat ist zwar signiert, aber nach MET ist das Signaturdatum um eine Stunde nach hinten verschoben, liegt also vor dem Erzeugungsdatum. Beim Importieren des Zertifikates reagiert Ikeyman dann mit einem Fehler „ASN.1 Fehler bei der Ver/Entschlüsselung."

Installation des Root Zertifikates in Browser

Da die CA der get AG nicht über ein Zertifikat aus der in den Browsern vorinstallierten CAs verfügt, ist es notwendig, dass der Nutzer das Root Zertifikat der get AG CA in seinen Browser importiert. Dazu wird ihm eine URL zur Verfügung gestellt, von der aus er das Zertifikat installieren kann. Wird das Root Zertifikat nicht in den Zertifikatsspeicher geladen, so muss der Nutzer explizit bei jedem SSL-Handshake bestätigen, dass er diesem Zertifikat traut.

Zertifikatexport bei Opera 5.12 Win

Opera 5 hat scheinbar keine Nutzerschnittstelle zum Importieren und Exportieren persönlicher Zertifikate. Es lassen sich jedoch Schlüssel und Zertifikatanforderungen

generieren. Zertifikate lassen sich über S/MIME in den Browser integrieren. Eine Authentifizierung gegenüber Webservern ist somit möglich. Allerdings kann dieses Zertifikat durch die fehlende Exportfunktion dann nicht in anderen Anwendungen verwendet werden.

7.12 Anlagen

1 CD

7.13 Inhalt der CD

Die beigelegte CD enthält die folgenden Programme und Dateien:

1. Die implementierten Java Klassen für die Authentifizierung
2. Die Gespeicherten Prozeduren für die Authentifizierung
3. Die implementierten Java Klassen für die Autorisierung
4. Die Gespeicherten Prozeduren für die Autorisierung
5. Den Auszug aus der Konfigurationsdatei des IBM HTTP Servers für die Konfiguration des SSL Servers für Autorisierung und Übertragung
6. Die Konfigurationsdatei von mod_jk
7. Die Basiskonfigurationsdatei der Zertifizierungsstelle
8. Die Konfigurationsdateien der einzelnen Zertifizierungsstellen
9. Die programmierten Shellskripte für den Betrieb der CA mit OpenSSL
10. Diese Arbeit im PDF-Format
11. Diese Arbeit im HTML-Format

Ich versichere, dass ich die Diplomarbeit selbstständig verfasst und keine anderen Quellen und Hilfsmittel als die angegebenen benutzt habe. Alle Stellen der Arbeit, die dem Wortlaut und dem Sinn nach anderen Werken entnommen sind, habe ich in jedem einzelnen Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht. Das gleiche gilt auch für die beigegebenen Zeichnungen, Skizzen und Darstellungen.

Leipzig, den 14.7.2002