G. Wolf, A. Pfitzmann, A. Schill, G. Wicke, J. Zöllner: Sicherheitsarchitekturen: Überblick und Optionen; erschienen in: G. Müller, A. Pfitzmann (Hrsg.): Mehrseitige Sicherheit in der Kommunikationstechnik; Addison-Wesley, 1997.

Sicherheitsarchitekturen: Überblick und Optionen♣

G.Wolf, A.Pfitzmann**, A.Schill*, G.Wicke**, J.Zöllner**

Technische Universität Dresden, 01062 Dresden

**Institut für Betriebssysteme, Datenbanken und Rechnernetze*

***Institut für Theoretische Informatik*

Zusammenfassung

Der Begriff „Sicherheitsarchitektur" wird in der Wissenschaft unterschiedlich verwendet. Im vorliegenden Artikel wird eine Klasseneinteilung existierender Ansätze für Sicherheitsarchitekturen vorgenommen. Einige dieser Ansätze werden exemplarisch bezüglich allgemeiner Anforderungskriterien untersucht. Nahezu durchgängig vorhandene Defizite liegen besonders in der Unterstützung der Gleichberechtigung aller Kommunikationspartner und der dazu notwendigen Aushandlung der Sicherheitsanforderungen zwischen diesen. Die Ausgestaltung der Verhandlung in heterogenen Systemen ist somit ein dringendes Problem, das es zu lösen gilt. Es wird diskutiert, wie Aushandlungsprozesse auf verschiedenen Abstraktionsniveaus realisiert werden können.

1 Motivation und Anforderungen

1.1 Motivation

Untersuchungen in den Bereichen elektronischer Zahlungsverkehr, Workflow Management und Medizin zeigen, daß bei konkreten Anwendungen verschiedene Interessen und Anforderungen der Nutzer bestehen (siehe dazu [WWZ_96]). Das führt zu der Erkenntnis, daß mehrseitige Sicherheit ein wesentlicher Bestandteil einer Sicherheitsarchitektur sein muß. Nur so kann sie bei der Vielfalt der möglichen Anwendungen, die auf Sicherheit und Schutz in offenen Datennetzen angewiesen sind, auf Akzeptanz ihrer Nutzer stoßen.

♣ Diese Arbeit wurde finanziell unterstützt vom Bundesministerium für Bildung, Wissenschaft, Forschung und Technologie (BMBF).

Mehrseitige Sicherheit in der Telekommunikation bedeutet, daß Anwendern die Möglichkeit gegeben wird, ihre Sicherheitsinteressen und -anforderungen zu formulieren und durch einen Abgleich ihrer Forderungen die Basis für eine Kommunikation miteinander zu schaffen. Will man mehrseitige Sicherheit technisch realisieren, so müssen Funktionen bereitgestellt werden, die den Anwender bei der Formulierung seiner Interessen unterstützen. Diese Funktionen müssen jedoch auf verschiedenen Abstraktionsebenen wirksam sein, um Nutzer bei ihren Aktionen auf dem Weg zur mehrseitigen Sicherheit in effizienter Weise zu begleiten.

Nach einer Beschreibung der wesentlichen allgemeinen Anforderungen an Sicherheitsarchitekturen im folgenden Abschnitt wird im zweiten Kapitel eine Auswahl von Architekturen und verwandten Ansätzen zur Realisierung von Sicherheit zwischen Kommunikationspartnern untersucht. Auf der Grundlage der existierenden Schwächen werden Schlußfolgerungen für die Konzeption einer Architektur für mehrseitige Sicherheit gezogen. Erste Konzepte für die Realisierung von Konfigurationsmöglichkeiten und Aushandlungsprozessen auf unterschiedlichen Ebenen werden im Kapitel 3 vorgestellt.

1.2 Allgemeine Anforderungen an Sicherheitsarchitekturen

Im folgenden werden Anforderungen genannt, die zum Vergleich verschiedener existierender Architekturen bzw. Architekturansätze herangezogen werden. Dabei steht die Zweckdienlichkeit aus Nutzersicht im Vordergrund. Die Kriterien lassen sich grob den organisatorischen Rahmenbedingungen, der angebotenen Funktionalität und der Implementierung der Architektur zuordnen (siehe Tabelle 1.1).

Rahmenbedingungen	*angebotene Funktionalität*	*Implementierung*
• Validierbarkeit • Referenzimplementierung • Sicherheitspolitik • überprüfbare Rechteentstehung • Standardisierung der Architektur • Zertifizierung der Architektur • Marktverfügbarkeit • Voraussetzung vertrauenswürdiger Instanzen	• Unterstützung spezieller Hardware • Anonymität von Instanzen • Sicherheitsmechanismen • Skalierbarkeit	• Schnittstellenspezifikation • Erweiterbarkeit • Anforderungen an Hardware • Verwendung von Standards • Verwendung zertifizierter Bausteine

Tabelle 1.1: Vergleichskriterien für Sicherheitsarchitekturen

Die einzelnen Kriterien werden nun näher erläutert:

Rahmenbedingungen (insbesondere organisatorischer, technischer und juristischer Art)**:**

- Validierbarkeit: Wird eine Bewertung der Architektur hinsichtlich ihrer Sicherheitseigenschaften ermöglicht? Kann sich der Endanwender davon überzeugen, d.h. liegen Spezifikationen bzw. Quelltexte vor?
- Sicherheitspolitik: Setzt die Architektur eine fest vorgegebene Sicherheitspolitik um oder ist sie politikneutral, d.h. erlaubt sie verschiedene? Ist die Architektur offen für neue Schutzziele der Nutzer? (Dies hat z.B. Auswirkungen darauf, ob Instanzen gleichrangig bzw. gleich stark sein können oder nicht.)
- überprüfbare Rechteentstehung: Wenn Nutzer Rechte nicht nur verliehen bekommen, sondern auch selbst erzeugen und weitergeben können, ist solch ein Transfer dann überprüfbar und zum Ursprung zurückführbar? Welche Instanzen müssen kooperieren, um eine Zurückführung zu ermöglichen (z.B. keine, alle)?
- Voraussetzung vertrauenswürdiger Instanzen: Muß eine bestimmte Infrastruktur vorhanden sein; werden Instanzen zur Schlüsselzertifizierung, -verteilung oder -generierung vorausgesetzt?
- Zertifizierung: Ist die Sicherheit der Architektur bezüglich bestimmter Kriterien zertifiziert worden?
- Standardisierung: Ist das Konzept der Architektur ein (internationaler) Standard?
- Referenzimplementierung: Existiert eine beispielhafte Implementierung der Architektur?
- Marktverfügbarkeit: Handelt es sich - sofern eine Implementierung existiert - um ein frei verfügbares (public domain) oder ein kommerzielles Produkt?

angebotene Funktionalität:

- Unterstützung spezieller Hardware: Kann durch die Architektur der Einsatz spezieller Hardware unterstützt werden?
- Anonymität von Instanzen: Unterstützt die Architektur Anonymitätskonzepte; verhält sie sich diesbezüglich möglicherweise neutral?
- Sicherheitsmechanismen: Bietet die Architektur Sicherheitsmechanismen für
  - symmetrische Verschlüsselung,
  - asymmetrische Verschlüsselung,
  - symmetrische Authentisierung,
  - asymmetrische Authentisierung, digitale Signatur,
  - Schlüsselaustausch,
  - Schlüsselzertifizierung,
  - Zugriffskontrolle u.a.
- Anpaßbarkeit: Ist die angebotene Funktionalität für eine Feinabstimmung auf spezielle Bedürfnisse verschiedener Nutzer geeignet?

Implementierung:

- Schnittstellenspezifikation: Sind die Export- bzw. Importschnittstellen der Architektur erweiterbar? Herrscht z.B. Offenheit gegenüber neuen Kryptoverfahren (Import)? Werden zur Diensterbringung ausreichende Exportschnittstellen angeboten?
- Erweiterbarkeit: Können Module mit neuer Funktionalität aufgenommen werden?
- Anforderungen an Hardware: Welche technischen Voraussetzungen, die über gängige Standardkonfigurationen hinausgehen, müssen erfüllt sein (z.B. manipulationssichere (tamper resistant) Hardware, Chipkartenleser)?
- Verwendung von Standards: Sind standardisierte Kryptoverfahren und Datenaustauschformate in der Implementierung enthalten?
- Verwendung zertifizierter Bausteine: Wird in der untersuchten Architektur zertifizierte Hard-/ Software eingesetzt?

2 Existierende Ansätze

Im folgenden Kapitel erfolgt die Beschreibung und ein bewertender Vergleich von Sicherheitsarchitekturen und verwandten Ansätzen. Diese Ansätze können in die nachstehenden Kategorien eingeordnet werden:

- Betriebssysteme: besonders solche, die Konzepte einer Sicherheitsarchitektur integriert haben (z.B. BirliX [HäKK_92], Windows NT [Cust_95]).
- Krypto-Bibliotheken, Schnittstellen, APIs: Sammlungen von kryptographischen Verfahren mit Anwendungsschnittstelle (z.B. LiSA [BKMS_96], Kryptomanager [Kann_94]), alleinstehende Schnittstellen zu kryptographischen Verfahren oder Programmierschnittstellen mit oder ohne Krypto-Bibliothek, (z.B. das Microsoft CryptoAPI [Wiew_96], SecuDE [Secu_96] und das GSS-API [Linn_93]).
- Anwendungsspezifische „Architekturen“: Lösungen zur Erhöhung der Sicherheit in Anwendungen, z.B. Ansätze für Zahlungssysteme (EU-Projekt SEMPER [Waid_96] und der SET-Standard [SET_96]) oder für Datenbanken (IRO-DB [EKPT_96]).
- Verteilte Systeme: verteilte Betriebssysteme bzw. verteilte Betriebssystemaufsätze (Middleware), z.B. Kerberos [Hugh_96], DCE [Schi_97] oder CORBA [OMG_95].
- Sicherheitsarchitekturen: nicht nur Sammlungen von kryptographischen Verfahren mit verallgemeinerter Schnittstelle, sondern Integration besonderer Konzepte, wie z.B. die Integration von Sicherheitspolitiken oder Aushandlungen (CISS [MPSC_93], PLASMA [Kran_96]).

Einige Ansätze aus den oben angeführten Kategorien werden zunächst exemplarisch vorgestellt, um einen Überblick der entsprechenden Aufgabenstellung in diesen Bereichen zu vermitteln. Anschliessend, in Abschnitt 2.5 werden diese Beispiele weiteren vergleichbaren Ansätzen gegenübergestellt.

2.1 Microsoft CryptoAPI

Das CryptoAPI ist der Versuch von Microsoft, Sicherheit für Anwendungen einfacher verfügbar zu machen. Es wurde erkannt, daß die Entwicklung moderner Applikationen aufgrund fortgeschrittener Kommunikationsmöglichkeiten zunehmend die Integration von Sicherheitsmechanismen erfordert. Dennoch wird diese Forderung aus Gründen des erhöhten Aufwandes oft nicht umgesetzt.

Das CryptoAPI liegt in der Version 1.1 vor und wird mit dem Betriebssystem Windows NT 4.0 ausgeliefert; die erweiterte Version 2.0 ist in Arbeit. Entwicklungsziel war es laut Microsoft, „die Entwicklung krypto-fähiger Anwendungen funktionell, flexibel, benutzbar und exportierbar zu machen." [Wiew_96]

Architektur

Die Architektur des CryptoAPI ist demzufolge modular, alle kryptographischen Operationen werden durch austauschbare Komponenten realisiert, die als CSP (cryptographic service provider) bezeichnet werden. Die CSPs werden unabhängig von der Applikation implementiert und über eine definierte Schnittstelle (das CAPI) angesprochen. Dadurch sollen die CSPs austauschbar sein und eine Anwendung kann mit unterschiedlichen CSPs zusammenarbeiten, was z.B. verschiedene Sicherheitsstufen ermöglichen könnte.

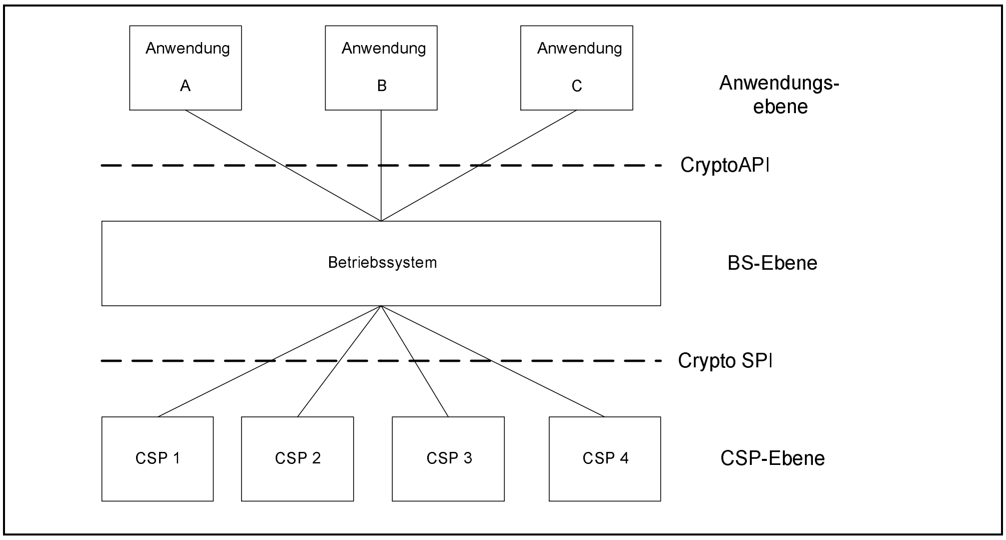

Abbildung 2.1: CryptoAPI-Architektur

Die Funktionalität des CryptoAPI läßt sich laut [Wiew_96] in 5 Gruppen teilen:

- „context management" zur Auswahl von verschiedenen CSPs durch eine Anwendung. Diese Auswahl kann sowohl nach Name als auch nach Funktionalität des CSPs getroffen werden.
- „key generation" zur Erstellung und Parametrisierung von asymmetrischen und symmetrischen Schlüsseln.
- „key exchange" für Übertragung und Austausch von Schlüsseln.
- „data encryption/decryption" zur Ver- und Entschlüsselung von Daten nach verschiedenen Algorithmen.
- „hashing/signature" zur Sicherung von Integrität und Authentizität von Nachrichten, z.B. durch digitale Signaturen.

Hinzukommen werden in Version 2.0 beispielsweise Funktionen zur Verwaltung von Zertifikaten.

Implementierung des Systems

CSPs sind als digital signierte DLLs realisiert. Sie werden nicht direkt von der Anwendung angesprochen, die entsprechenden Aufrufe werden vielmehr an das Betriebssystem gerichtet und von diesem weitergeleitet. Die Schnittstelle zwischen Betriebssystem und CSPs wird auch als Crypto SPI (Service Provider Interface) bezeichnet (vgl. Abbildung 2.1).

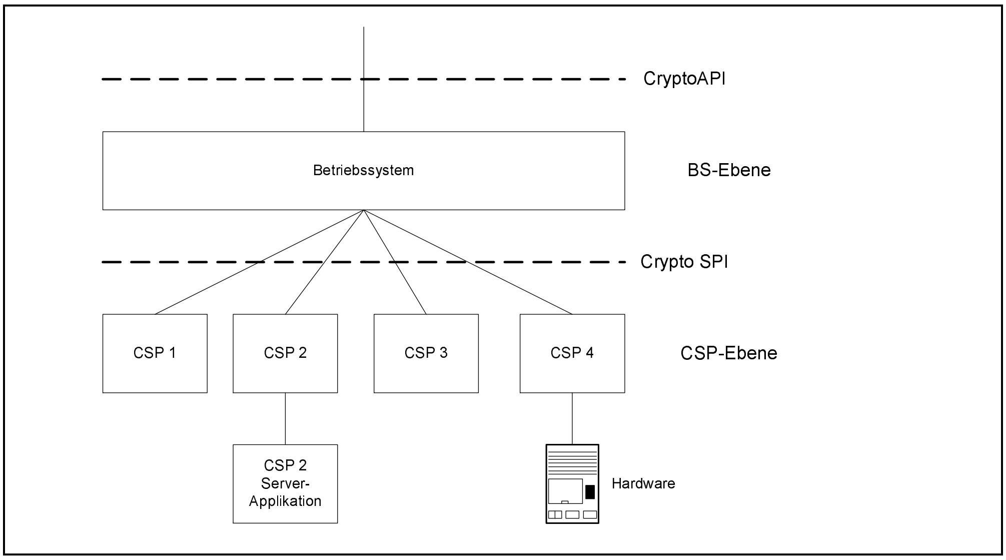

Abbildung 2.2: Komplexe CSPs

Eine Anwendung kann gleichzeitig mit mehreren CSPs in Verbindung stehen und einen entsprechend geeigneten CSP für die Ausführung von bestimmten Operationen auswählen.

CSPs können durchaus aus mehreren Komponenten bestehen, wie in Abbildung 2.2 dargestellt. Hervorzuheben ist hier der CSP 4, dessen eine Komponente auf spezialisierter Hardware abläuft. Ein Beispiel für solche Hardware ist eine Smartcard. Solche Geräte können insbesondere die persönliche Sicherheit des Nutzers und sein Vertrauen in das System viel besser unterstützen als Software allein.

Basisfunktionalität

Das CryptoAPI wird von Microsoft mit einem Basis-CSP ausgeliefert. Dieser ist eine Implementierung des Typs PROV_RSA_FULL. Einige dieser Typen und die von ihnen verwendeten Verfahren sind in Tabelle 2.1 aufgeführt.

Es ist vorgesehen, die Palette je nach Notwendigkeit zu erweitern, in die Tabelle sind bereits implementierte bzw. geplante CSPs aufgenommen worden.

CSP-Typ	Schlüssel-austausch	Signatur	Verschlüsselung	Hashing
PROV_RSA_FULL	RSA	RSA	RC2, RC4	MD5, SHA
PROV_RSA_SIG	nein	RSA	nein	MD5, SHA
PROV_DSS	nein	DSS	nein	SHA
PROV_FORTEZZA	KEA	DSS	Skipjack	SHA
PROV_MS_MAIL	RSA	RSA	CAST	MD5
PROV_SSL	RSA	RSA	verschiedene	verschiedene

Tabelle 2.1: CSP-Typen und ihre Mechanismen

Wie man Tabelle 2.1 entnehmen kann, bietet schon die gebotene Grundfunktionalität von PROV_RSA_FULL eine ausreichende Palette verbreiteter kryptographischer Mechanismen an. Die verwendbaren Schlüssellängen sind allerdings begrenzt - maximal 512 Bit für RSA und 40 Bit für die symmetrischen Verschlüsselungsverfahren.

2.2 SET

Der Standard SET (Secure Electronic Transaction) spezifiziert die Abwicklung von Zahlungen mit Kreditkarte über Kommunikationsnetze sowie die Verwaltung des Kreditkartenkontos. Veröffentlicht wurde er von den Kreditkarteninstituten VISA und MasterCard und ist von den Firmen IBM, Microsoft, Netscape, GTE, SAIC, Terisa und Verisign mitentwickelt worden, vgl. [SET_96]. Vorrangig ist er auf den US-amerikanischen Markt zugeschnitten und nicht unmittelbar auf die deutsche Kreditwirtschaft übertragbar.

SET-Nachrichten (vgl. Abbildung 2.3) werden mit dem DES (Data Encryption Standard) verschlüsselt. Der dafür notwendige Sitzungsschlüssel wird mit RSA verschlüsselt ausgetauscht. Mit SHA (Secure Hash Algorithm) erzeugte Hashwerte der Nachrichten werden unter Verwendung von RSA signiert. Vorgesehen ist, diese Nachrichten per E-mail zu versenden, wobei dies nach dem MIME-Standard (Multi-purpose Internet Mail Extensions) erfolgt. MIME dient dabei zur Beschreibung der Kapselung der SET-Nachrichten für den Transport. Damit ist eine plattformunabhängige Verarbeitbarkeit der SET-Nachrichten gewährleistet. Als Transportprotokoll wird exemplarisch das Internetprotokoll TCP/IP beschrieben.

Herauszustellen sind daneben die kryptographischen Kapselungsmöglichkeiten, wie sie für eine Sicherheitsarchitektur in Frage kommen. Die Datenformate für Verschlüsselung und digitale Signatur folgen dabei dem sog. Public-Key Cryptography Standard (PKCS). SET beschreibt weiterhin eine Zertifizierungshierarchie für X.509-Zertifikate, die einen extrem vertrauenswürdigen Wurzelknoten für die Urzertifikat-Erstellung erfordert.

SET sieht zum Teil nur unidirektionale Verschlüsselung vor, so vom Kreditkarteninhaber zum Händler, aber nicht umgekehrt. Diese Einschränkung der Protokolle ist nicht technischer Natur, sondern liegt in der Einschätzung begründet, daß nicht alle potentiellen SET-Nachrichten schützenswert sind.

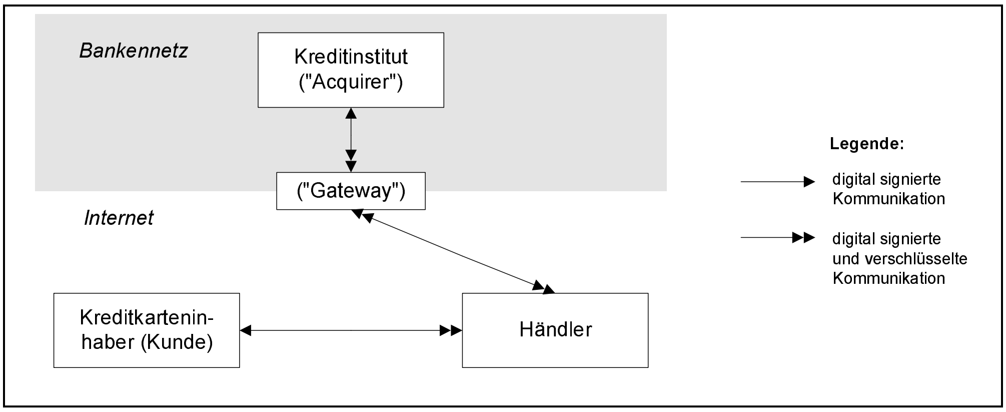

Abbildung 2.3: SET-Kommunikationsbeziehungen

Hierarchische Zertifizierung

Die in Abbildung 2.4 dargestellte SET-Zertifizierungshierarchie zeigt, wie eine (noch zu schaffende) globale Struktur um SET-spezifische Instanzen erweitert wird. Es sind Instanzen zur Zertifizierung von Kreditkarteninhabern, kartenakzeptierenden Händlern und entsprechenden Kreditinstituten (genauer sog. „Acquirer" und die ihnen zugeordneten sog. „Acquirer Payment Gateways"). Die erzeugten Zertifikate dienen zum Nachweis der Authentizität der in den Transaktionen verwendeten Schlüssel. Im einzelnen sind das bei SET Zertifikate für

a) öffentliche Testschlüssel zur digitalen Signatur bei Transaktionsdaten,

b) öffentliche Schlüssel zur Verschlüsselung symmetrischer Sitzungsschlüssel,

c) öffentliche Testschlüssel zu Zertifikaten,

d) öffentliche Testschlüssel zu Zertifikat-Rückruflisten.

Alle Zertifizierungsinstanzen müssen Zertifikate c) und d) vorweisen können. (SET sieht eine Hierarchie vor, bei der sich die Wurzelinstanz selbst zertifiziert.) Alle weiteren Zertifikate werden von der jeweils übergeordneten Instanz ausgegeben.

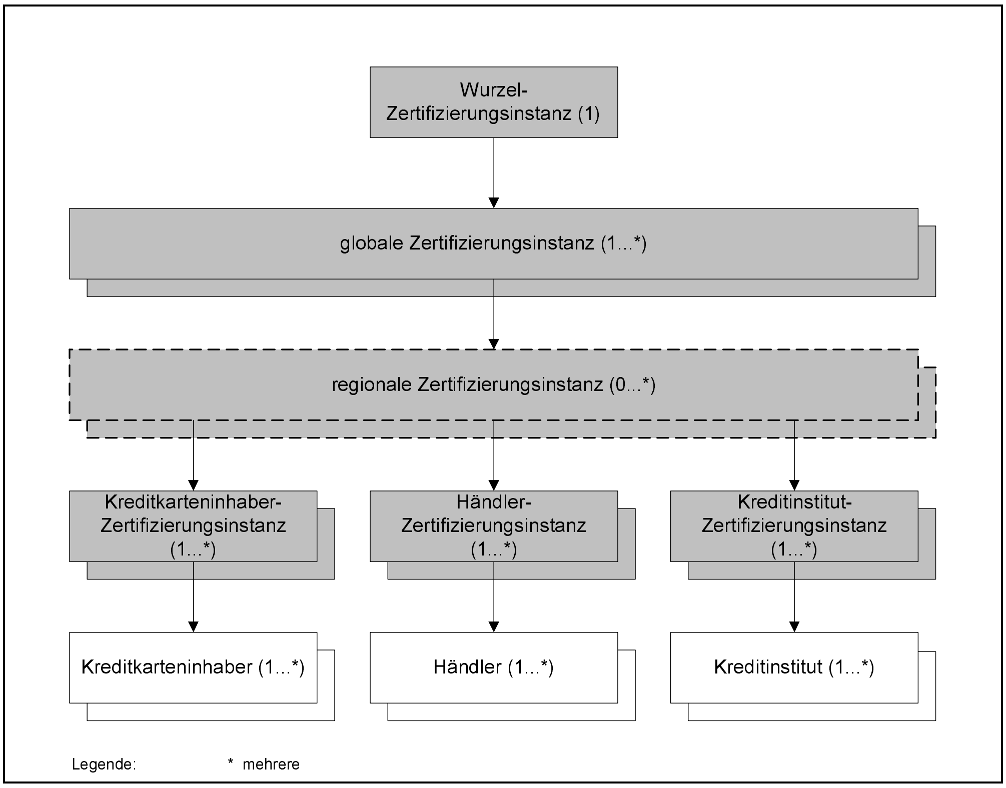

Abbildung 2.4: SET-Zertifizierungshierarchie [SET_96]

Spezifisch bei SET verwendete Zertifikate sind solche der Kategorie a) für digitale Signaturen und b) für Verschlüsselung. Zertifikate a) benötigen die einzelnen Kreditkarteninhaber, Händler, Kreditkarteninstitute („Acquirer“ und „Gateways“) sowie die Zertifizierungsinstanzen, die ihnen Zertifikate ausstellen. Zertifikate b) benötigen nur die Instanzen, die verschlüsselte Nachrichten erhalten sollen. Das sind bei SET Händler, Kreditinstitute („Gateways“) und die Zertifizierungsinstanzen für Kunden, Händler und Kreditinstitute. Um angemessene und langanhaltende Sicherheit zu erreichen, wird als Länge der RSA-Schlüssel 1024 Bit vorgeschlagen. Eine Ausnahme bildet die Wurzelzertifizierungsinstanz mit 2048 Bit.

Die Zertifikate c) und d) können bei SET kombiniert werden. Für die digitale Signatur von Zertifikaten sowie von Zertifikat-Rückruflisten sind demzufolge nicht notwendigerweise verschiedene Schlüssel erforderlich. Eine Zertifizierungsinstanz der untersten Ebene (siehe Abbildung 2.4) darf außerdem Rollen für Zertifikate kombinieren, d.h. z.B. sowohl

Kreditkarteninhaber als auch Kreditinstitute und Händler unter Nutzung ein und desselben Schlüssels zertifizieren.

Die Etablierung einer solchen internationalen Zertifizierungshierarchie im Sinne einer Sicherheitsinfrastruktur ist jedoch nicht unabhängig von den jeweiligen politischen Rahmenbedingungen, unter denen Zertifizierungsinstanzen arbeiten. So stellen eventuelle Einschränkungen der Schlüssellänge aufgrund von nationalen Exportbeschränkungen bzw. Kryptoregulierungen eine Schwächung der Architektur dar.

2.3 DCE und Kerberos

DCE (Distributed Computing Environment) ist die Architektur der OSF (Open Software Foundation) für Client/Server-Anwendungen, vgl. [Schi_97]. Der Aufbau ist in Abbildung 2.5 zu sehen.

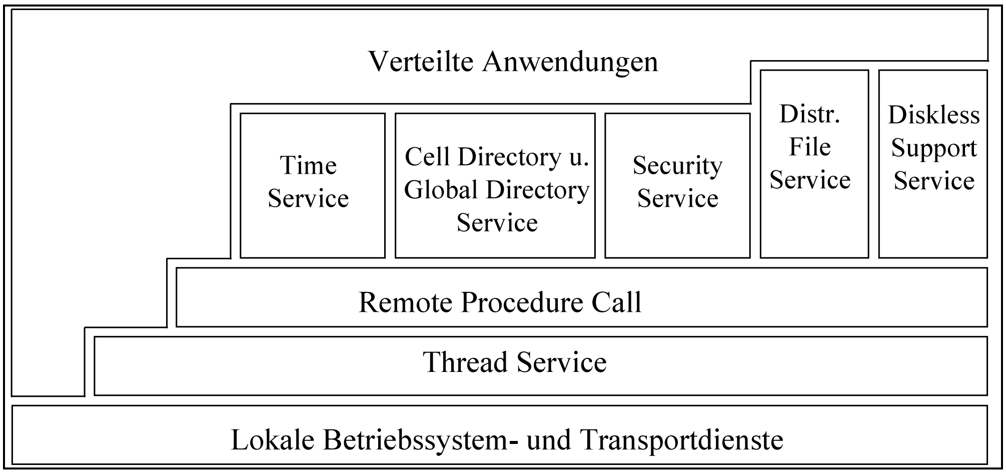

Abbildung 2.5: Aufbau des OSF DCE

Aufsetzend auf Betriebssystemdiensten (z.B. Unix, OS/2, Windows/NT, Windows95, MVS, OpenVMS u.a.) und Transportprotokollen (z.B. TCP/IP oder UDP/IP) werden Kommunikationsmechanismen mittels Remote Procedure Call (RPC) angeboten. Der RPC stellt bei DCE - wie bei den meisten Client/Server-Systemen - die Basis für alle Interaktionen zwischen Anwendungskomponenten dar. Ergänzt wird dies durch einen Thread Service für nebenläufige/parallele Verarbeitung, durch den unten näher beschriebenen Security Service, den Directory Service für die verteilte Namensverwaltung, den Time Service für die verteilte Uhrensynchronisation sowie das Distributed File System für die netzweite Dateiverwaltung. Auf DCE aufsetzend sind verschiedene Produkte für die verteilte Transaktionsverwaltung verfügbar; hier ist insbesondere das System Encina [Sher_93] zu nennen, das eine Integration von DCE-basierten Client/Server-Umgebungen mit Host-basierten Transaktionsmonitoren ermöglicht.

DCE ist vollständig standardisiert und bereits auf breiter Basis verfügbar; es wird von allen großen Rechnerherstellern für die gängigen, oben genannten Betriebssysteme angeboten. Die Runtime-Lizenz des RPC ist kostenlos verfügbar. Dies gilt allerdings nicht für die Server-Lizenzen und die Entwicklungslizenzen.

Grundlegende Sicherheitsmechanismen, die von DCE unterstützt werden, sind Authentikation, Zugriffskontrolle und (symmetrische) Verschlüsselung, wie in Abbildung 2.6 dargestellt.

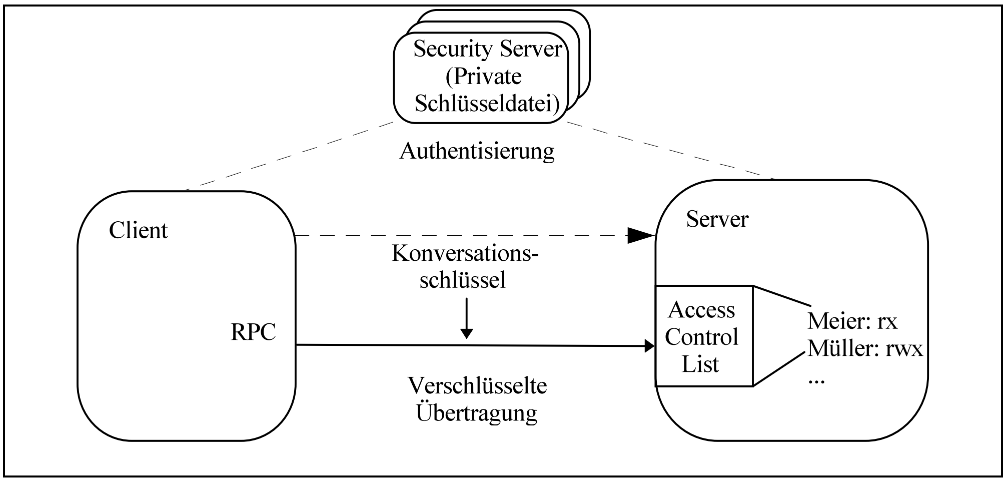

Abbildung 2.6: Einsatz der DCE-Dienste am Beispiel einer Client/Server-Anwendung

Die Autorisierung ermöglicht die selektive Vergabe von Zugriffsrechten auf Server an Clients. Sie basiert auf Zugriffskontrollisten, die von den einzelnen Servern dezentral verwaltet werden und z.B. auch Gruppeneinträge sowie Einträge für Nutzer aus anderen DCE-Verwaltungsbereichen (DCE Cells) umfassen können.

Zur Authentikation und zur Verteilung von Schlüsseln wird das ursprünglich am Massachusetts Institute of Technology (MIT) entwickelte System Kerberos eingesetzt [Hugh_96]. Es erlaubt die gegenseitige Authentikation von Clients und Servern in einem „unsicheren" Netz. Kerberos ist ein Trusted Third Party-System und arbeitet mit symmetrischer Verschlüsselung (DES). Der Schlüsselaustausch erfolgt nach dem Needham-Schroeder-Protokoll [NeSc_78]. Fordert ein Nutzer einen Dienst an, so überprüft das System dessen Identität mit Hilfe des Schlüssels und generiert einen Session-Key, der für die Dauer der Sitzung gültig ist. Der entsprechende Ablauf ist in Abbildung 2.7 detaillierter dargestellt. Zunächst wendet sich ein Client im Rahmen der Login-Phase an den Security Server seines DCE-Verwaltungsbereiches. Dieser überprüft die Identität des Clients und übergibt ihm ein mit dem privaten Schlüssel des Clients verschlüsseltes Ticket Granting Ticket (TGT). Die Schlüssel der Teilnehmer müssen dazu beim Security Server bekannt sein und dort (in verschlüsselter Form) abgelegt sein. Falls der Client authentisch ist, ist er in der Lage, das TGT zu entschlüsseln. Nach bestimmten Modifikationen der TGT-internen Daten sendet er das TGT zusammen mit weiteren Informationen wieder an den Security

Server. Nach endgültiger Prüfung der Authentizität des Clients liefert der Security Server ein Privilege Attribute Certificate (PAC) an den Client. Damit ist die Login-Phase abgeschlossen.

Wenn nun seitens des Clients der Aufruf eines Anwendungsservers (z.B. Dateiserver, Druckerserver etc.) gewünscht wird, so ist eine erneute Interaktion mit dem Security Server erforderlich: Der Client fordert unter Angabe des PAC ein spezielles Ticket und einen Konversationsschlüssel (Session Key) für den gewünschten Server an. Nun wird ein mit dem Session Key verschlüsselter RPC an den Server gesendet. Daraufhin erfolgt zunächst noch eine weitere Authentikationsphase zwischen Anwendungsserver und Client, wobei der jeweilige Partner durch Entschlüsselung von Zufallszahlen (Challenge) nachweisen muß, daß er authentisch ist.

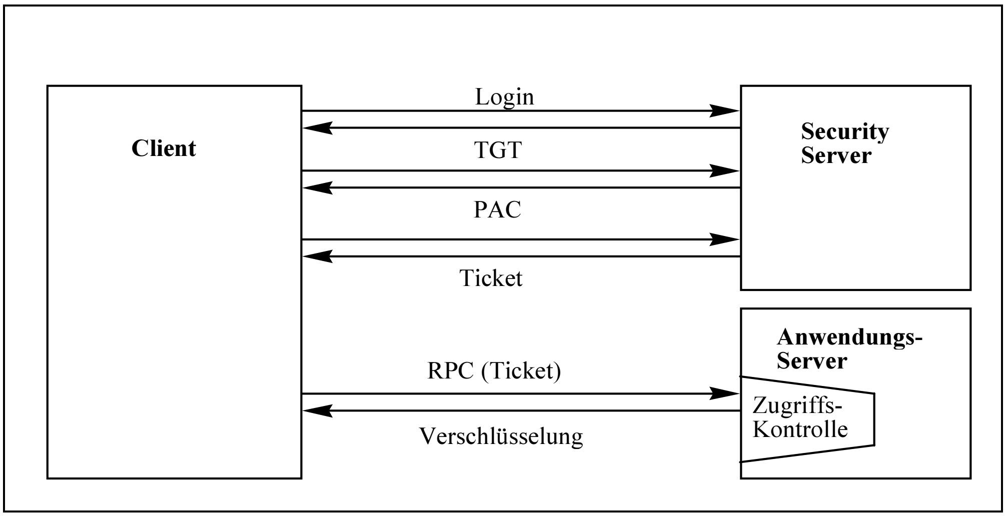

Abbildung 2.7: Authentikation und RPC-Interaktion bei DCE

Der RPC-Aufruf durchläuft schließlich noch die Zugriffskontrolle des Servers, die durch einen sogenannten ACL (Access Control List) Manager realisiert wird und die Zugriffsrechte des Clients überprüft. Bei positiver Prüfung wird der RPC ausgeführt, und die Ergebnisse werden wiederum verschlüsselt an den Client zurückgesendet. Weitere RPC-Interaktionen nach dem gleichen Muster können folgen.

Je nach Anforderung der Anwendung sind verschiedene Sicherheitsgrade (mit unterschiedlichem Laufzeitaufwand) wählbar, von einfacher Authentikation vor der Kommunikation bis hin zu Authentikation und Verschlüsselung für jede einzelne Nachricht. Anstelle der vollständigen Verschlüsselung ist es auch möglich, nur die Integritätsprüfung von Nachrichten auf der Basis verschlüsselter Prüfsummen zu wählen.

Neben neu entwickelten Anwendungen nutzen auch die DCE-Dienste selbst die verfügbaren Sicherheitsmechanismen. So werden die Einträge des Namensraums des DCE Directory Service durch Zugriffskontrollisten vor unberechtigtem Zugriff geschützt. Gleiches gilt auch für die vom DCE Distributed File System verwalteten Dateien sowie für einige weitere Systemdienste.

Derzeit werden von DCE keine asymmetrischen Kryptoverfahren unterstützt. Ein Einsatz solcher Mechanismen ist in künftigen DCE-Versionen vorgesehen. Aufgrund der fehlenden Unterstützung asymmetrischer Algorithmen sind auch Konzepte wie digitale Signaturen und Schlüsselzertifizierungsdienste derzeit nicht in DCE enthalten. Erwartungsgemäß ist der Aufwand für die Schlüsselverteilung aufgrund der ausschließlichen Nutzung symmetrischer Kryptoverfahren hoch. Sichere Schlüsselverteilung ist aber besonders in einem verteilten System mit vielen Nutzern von entscheidender Bedeutung. Sie ist in DCE noch nicht sinnvoll gelöst.

2.4 CISS

CISS (Comprehensive Integrated Security System) stellt den Kern eines Entwurfs einer Sicherheitsarchitektur für offene verteilte Systeme dar und wurde in [MPSC_93] dargelegt. Die Architektur baut auf von ISO/OSI standardisierten Sicherheitsdiensten auf, ist jedoch um einige zusätzliche Dienste erweitert worden.

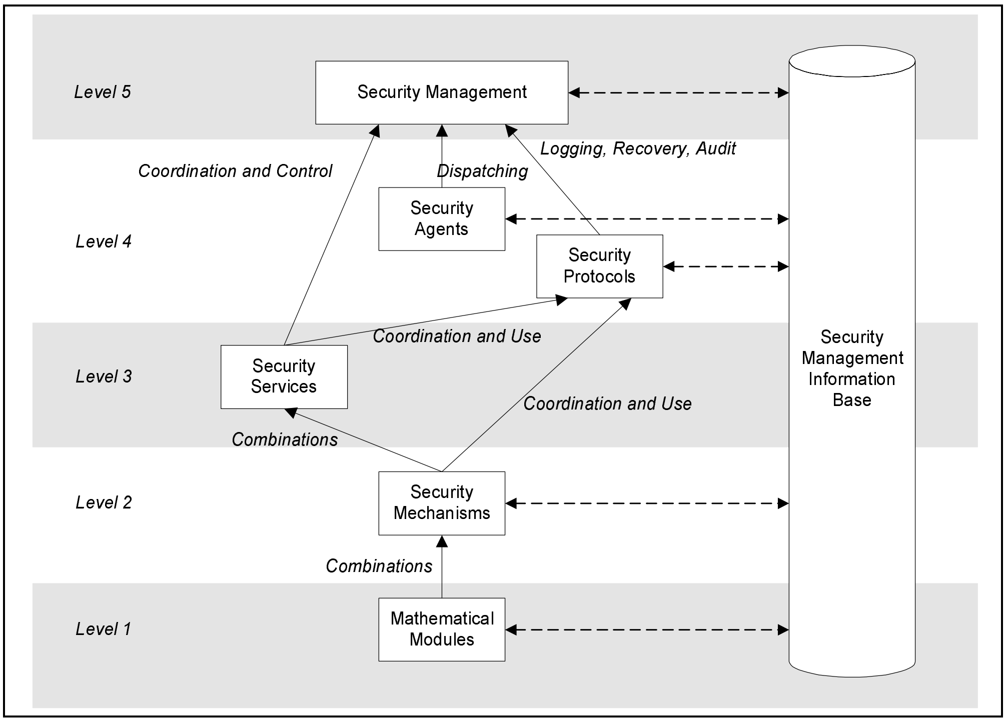

Abbildung 2.8: Konzeptuelles Modell von CISS (nach [MPSC_93])

Die Konzeption für offene verteilte Systeme enthält interessante Ansätze, die in eine Architektur für mehrseitige Sicherheit eingehen können. Zu nennen ist das konzeptuelle

Modell, welches „Security Management“, „Security Agents“, „Security Protocols“, „Security Services“, „Security Mechanisms“, „Mathematical Modules“ und eine „Security Management Information Base“ vereint.

Das konzeptuelle Modell von CISS (Abbildung 2.8) sieht eine Unterteilung in fünf Ebenen vor, die zur Verringerung der Komplexität beitragen. In jeder Ebene sind Architekturkomponenten angesiedelt, denen jeweils Segmente in einer Datenbank (der „Security Management Information Base“) entsprechen. Dabei handelt es sich um ein Repository aller Parameter, die zur Steuerung des Ablaufs in der CISS-Architektur dienen. Es gibt statische Bereiche z.B. für Zugriffskontrolle und Identifizierung neben dynamischen Bereichen z.B. zur Aufnahme von Sitzungsschlüsseln, sowie Logbucheinträge. Die „Security Management Information Base“ stellt eine Erweiterung der X.500-Directory-Informationsdatenbank dar.

Die unterste Ebene (Level 1) enthält Module, die elementar für die Implementation verschiedener Sicherheitsmechanismen sind. Die folgenden Ebenen (Level 2 bis Level 5) enthalten Module, die sich jeweils aus den Modulen darunterliegender Ebenen zusammensetzen. Bis auf die Sicherheitsdienste (Level 3) greifen die Module der Ebenen auf die ihnen zugeordneten Segmente der „Security Management Information Base“ zurück.

Schwächen bei CISS liegen in der fehlenden Beschreibung des Zustandekommens der Repositoryeinträge und allgemeiner einer Konfiguration der Architektur durch die Umsetzung einer nutzerbezogenen Sicherheitspolitik.

2.5 Zusammenfassende Bewertung

In diesem Kapitel werden die zuvor beschriebenen Ansätze für Sicherheitsarchitekturen anhand der im Abschnitt 1.2 aufgestellten Anforderungen vergleichend gegenübergestellt. In den tabellarischen Vergleich werden weitere Ansätze einbezogen, deren hervorzuhebende Eigenschaften zuvor kurz beschrieben werden.

BirliX

Das verteilte Betriebssystem BirliX wurde bei der Gesellschaft für Mathematik und Datenverarbeitung (GMD) konzipiert. Neben dem Ziel, Erweiterbarkeit und Flexibilität in das Design eines Betriebssystems zu integrieren, bildeten verteilte Umgebungen und ihre Einwirkungen auf Fehlertoleranz, Leistung und Sicherheit den Hauptfokus des Projektes [Lupp_95].

Das Grundprinzip des Birlix-Kernels ist die Datenabstraktion. Das System ist ein Managementsystem für abstrakte Datentypen mit Diensten zum Definieren und Instanziieren von Typen, zum Identifizieren von Instanzen und zur Unterstützung von Kommunikation zwischen diesen. Durch dieses Design gewinnt BirliX eine leichte funktionale Erweiterbarkeit und Anpaßbarkeit an neue Anforderungen.

SecuDE

SecuDE (<u>Secu</u>rity <u>D</u>evelopment <u>E</u>nvironment) [Secu_96] wird von der GMD Darmstadt entwickelt und ist ein portables Sicherheitstoolkit für UNIX und MS-DOS-Plattformen. In seiner Bibliothek von Sicherheitsfunktionen mit C-Programmierschnittstelle beinhaltet es öffentlich bekannte und etablierte symmetrische und asymmetrische kryptographische Verfahren.

Im Gegensatz zu vielen anderen Sicherheitsbibliotheken bietet SecuDE u.a. Funktionen zur Zertifizierung, Handhabung von Zertifizierungspfaden, Cross-Zertifizierung und schwarzen Listen. Im Zusammenhang damit enthält SecuDE weiterhin Kommandos und Bibliotheksfunktionen für die Operation von Zertifizierungsinstanzen und die Interaktion zwischen Zertifizierungsinstanzen und zertifizierten Benutzern.

LiSA

LiSA (Library for Secure Applications) [BKMS_96] wurde von der Entwicklungsgruppe am Lehrstuhl für Kryptographie und algorithmische Zahlentheorie an der Universität des Saarlandes konzipiert und realisiert. Lisa ist eine C++-Klassenbibliothek für Daten- und Kommunikationssicherheit und repräsentiert somit eine objektorientierte Krypto-Bibliothek.

LiSA beinhaltet eine integrierte Schlüsseldatenbank und Schlüsselverwaltung, die unabhängig vom verwendeten Verschlüsselungsverfahren ist. Ein auf einem verteilten Client-Server-Konzept beruhender Schlüsselserver erlaubt es, automatisch öffentliche Schlüssel unbekannter Kommunikatiosnteilnehmer über das Internet zu erfragen, auf ihre Gültigkeit zu überprüfen und bei Bedarf in die persönliche Datenbank zu übernehmen.

CryptoManager++

Der CryptoManager++ ist eine an der Universität Hildesheim in C++ entwickelte objektorientierte kryptographische Softwarebibliothek, vgl. [Kann_94]. Hervorzuheben ist die Klassenhierarchie, die jedem kryptographischen Algorithmus eine spezifische Klasse zuordnet. Durch das verwendete objektorientierte Konzept ist aus Programmierersicht eine einfache Erweiterbarkeit gegeben. Der Quellcode steht zur Validierung zur Verfügung.

Mit der Entwicklung des CryptoManagers verbunden waren insbesondere die Ziele Korrektheit, Robustheit, Wiederverwendbarkeit, Erweiterbarkeit und Portabilität. Aus Nutzersicht interessant sind die Konfigurationsmöglichkeiten, die der CryptoManager++ bietet. So lassen sich bei allen Kryptosystemen, die das zulassen, Schlüssellängen einstellen, Betriebsarten festlegen, Hybridsysteme zusammensetzen u.a.m. Es lassen sich Vorgabewerte für die Parameter angeben, so daß sich eine Standardkonfiguration ergibt.

GSS-API

Die „Common Authentication Technology (CAT)“-Arbeitsgruppe der Internet Engineering Task Force (IETF) erarbeitet eine Reihe von APIs und Mechanismen zur Unterstützung von Anwendungsentwicklern durch Sicherheitsdienste. Seit der Veröffentlichung der RFC 1508 bis 1511 im Jahre 1993 fand das GSS-API weite Beachtung.

Als besondere Stärken des GSS-API werden in [NSA_95] hervorgehoben: die Unabhängigkeit des APIs von kryptographischen Algorithmen, von kryptographischen Modulen (es verbirgt z.B., ob Algorithmenimplementierungen in Hard- oder Software vorliegen), das modulare Design und die Möglichkeit, Anwendungen einzubinden, die ohne kryptographische Algorithmen oder Schnittstellen implementiert sind.

In den folgenden Tabellen 2.2 und 2.3 erfolgt ein Vergleich der untersuchten Ansätze in tabellarischer Form:

Rahmenbedingungen	BirliX	MS Crypto API	SecuDE	LISA	Krypto-manager	GSS-API
Validierbarkeit	ja	nein			ja	
Sicherheitspolitik	neutral	neutral	neutral	neutral	neutral	neutral
überprüfbare Rechteentstehung	ja	nein	nein	nicht anwendbar	nicht anwendbar	
Voraussetzung vertrauenswürdiger Instanzen	ja?			nein	nein	nein
Zertifizierung der Architektur	nein			nein	nein	nein
Standardisierung der Architektur	nein	nein	nein	nein	nein	ja
Referenz-implementierung	ja	ja	ja	ja	ja	ja
Marktverfügbarkeit	nein	kommer-ziell	spezielle Bedingun-gen	frei verfügbar?	frei verfügbar	
angebotene Funktionalität						
Unterstützung spezieller Hardware	nicht anwendbar	ja, je nach CSP	Chipkarten	nein	möglich	möglich
Anonymität von Instanzen	nein	neutral	neutral		neutral	neutral
Sicherheits-mechanismen[1]	ZK	sV, aV, sA, aA, flexibel erweiterbar	sV, aV, sA, aA, Sz	sV, aV, sA, aA	sV, aV, sA, aA	
Anpaßbarkeit	ja	hoch				

[1] Kurznotationen für Sicherheitsmechanismen: *sV:* symmetrische Verschlüsselung, *aV:* asymmetrische Verschlüsselung, *sA:* symmetrische Authentisierung, *aA:* asymmetrische Authentisierung = digitale Signatur, *Sa:* Schlüsselaustausch, *Sz:* Schlüsselzertifizierung, *Zk:* Zugriffskontrolle

Implementierung	BirliX	MS Crypto API	SecuDE	LISA	Krypto-manager	GSS-API
Schnittstellen-spezifikation	offen		fest		offen	
Erweiterbarkeit		ja	ja, über GSS-API	ja	ja	ja
Anforderungen an Hardware	keine	keine	keine	keine		keine
Verwendung von Standards	UNIX	ja	ja, X.509			Internet-RFCs
Verwendung zerti-fizierter Bausteine	nein	vor-gesehen				

Tabelle 2.2: Bewertung von Sicherheitsarchitekturen (1)[2]

SEMPER

Bei SEMPER (Secure Electronic Marketplace for Europe) handelt es sich um ein EU-Projekt zur Schaffung der Grundlagen für sicheren Handel über das Internet und andere offene Netze, vgl. [Waid_96]. Es wird eine offene und systemunabhängige Architektur für sicheren elektronischen Handel, die das Konzept der mehrseitigen Sicherheit verwirklichen soll, definiert.

Durch die geplante Einbindung externer Zahlungssysteme als Module wurde eine generische Importschnittstelle geschaffen, die durch den sog. „Payment Manager“ verwaltet wird. Aufgrund der starken Unterschiede zwischen den einzelnen Zahlungssystemen (z.B. tokenbasiertes e-cash, Kreditkartensystem iKP) wurden Adapter notwendig, die zwischen den Systemen und der Importschnittstelle liegen.

PLASMA

PLASMA (Platform for Secure Multimedia Applications) ist ein Projekt des Frauenhofer-Instituts für Graphische Datenverarbeitung, Darmstadt. Es ist in eine Reihe von mehreren Projekten eingebettet, die stark mit Arbeiten bei der DeTeBerkom zusammenhängen [GeKo_95]. Im Projekt PLASMA wird eine Sicherheitsplattform entwickelt, welche die erforderlichen Sicherheitsdienste für multimediale Telekommunikations- und Informationsdienste anbietet [Kran_96].

Mit PLASMA ist es möglich, kryptographische Protokolle, die angewendet werden sollen, auszuhandeln, bevor die eigentliche Kommunikation gestartet wird. Das erlaubt Kommunikation zwischen autonomen Domänen mit verschiedenen Sicherheitspolitiken. Die Kooperations-Sicherheits-Politik (KSP) wird aus den beiden Sicherheitspolitiken der kommuni-

[2] Die Aussagen wurden nach bestem Wissen der Autoren getroffen.

zierenden Partner, die zwischen den Partnern vor der sicheren Kommunikation ausgetauscht werden, errechnet [Kran_96].

Sicherheitspolitiken bestehen normalerweise aus einer Reihe von Regeln, die das sicherheitsbezogene Verhalten von Kommunikationsinstanzen definieren. Diese Regeln bestimmen zum Beispiel, welche Zugriffe zwischen Subjekten und Objekten erlaubt sind, welche kryptographischen Algorithmen für die sichere Kommunikation oder welche Protokolle zur Authentikation benutzt werden. Das Ziel des Konzeptes der „formalisierten Sicherheitspolitik", das in [GeKo_95] vorgeschlagen wird, ist, die Sicherheitspolitik außerhalb von Anwendungen explizit abzubilden, um Algorithmen eine anwendungsunabhängige Interpretation, Aushandlung und den Vergleich der Sicherheitspolitiken zu ermöglichen.

Rahmenbedingungen	SET	SEMPER	DCE und Kerberos	CORBA[3]	CISS	PLASMA
Validierbarkeit	ja	angestrebt	ja	nein	ja	nein
Sicherheitspolitik	fest	mehr-seitige Sicherheit	fest, nicht mehrseitig	neutral, verschied. möglich	offen	neutral, offen
überprüfbare Rechteentstehung	ja	ja	ja	ja		
Voraussetzung vertrauenswürdiger Instanzen	ja, für Zertifikate	derzeit für Zertifizie-rung	ja	nein	teilweise ja	
Zertifizierung der Architektur	nein	nein	nein	nein	nein	nein
Standardisierung der Architektur	ja	angestrebt	ja (OSF)	ja (OMG)	nein, nutzt aber OSI	nein
Referenz-implementierung	ja	Prototyp mit SET, e-cash	ja	nein	Prototyp	
Marktverfügbarkeit	kommer-ziell	nein	verfügbar	verfügbar		

[3] Für ausführliche Beschreibungen zu CORBA vgl. Artikel „Sicherheit in CORBA und TINA" im vorliegenden Buch.

angebotene Funktionalität	SET	SEMPER	DCE und Kerberos	CORBA	CISS	PLASMA
Unterstützung spezieller Hardware	möglich	geplant	nein	möglich	möglich	Chipkarten
Anonymität von Instanzen	ja?	ja	nein	ja?	neutral	
Sicherheits-mechanismen[1]	sV, aA	sV, aV, sA, aA, ...	sV, sA, asymm. geplant	ja, Zk, Vertrau-lichkeit	vgl. OSI	sV, aV, sA, aA, Sa, Sz
Anpaßbarkeit	gering	hoch	nein?		ja	ja
Implementierung						
Schnittstellen-spezifikation	fest	offen		offen		
Erweiterbarkeit		ja	bedingt	ja	ja	ja
Anforderungen an Hardware		nein	nein	neutral	nein	nein
Verwendung von Standards	ja	nein		möglich	ja, OSI	X.509
Verwendung zerti-fizierter Bausteine				möglich		möglich

Tabelle 2.3: Bewertung von Sicherheitsarchitekturen (2)[2]

Zusammenfassung

Die Ergebnisse der unter dem Aspekt der mehrseitigen Sicherheit besonders interessanten Kriterien werden in diesem Abschnitt noch einmal zusammengefaßt. Zum Teil werden die gestellten Kriterien von den diskutierten Systemen erfüllt; jedoch unterschiedlich gut zu den verschiedenen Fragestellungen aus Kapitel 1.2. Dies ist sicherlich zum großen Teil in den unterschiedlichen Zielrichtungen der Konzeption der Systeme begründet, wie z.B. Betriebssysteme, Krypto-Bibliotheken und Sicherheitsarchitekturen. Eine weitere Ursache liegt aber in der Herangehensweise an die Realisierung von Sicherheitsfunktionalität in den Systemen. Bei manchen wurde ein durchgängiges Sicherheitskonzept verfolgt, in anderen Systemen wurde lediglich eine Auswahl von Funktionen implementiert.

Sowohl die Validierbarkeit als auch die Zertifizierung der eingesetzten Software sind Grundvoraussetzungen für die Bildung eines Vertrauensbereiches, der das eigene Rechnersystem des Nutzers einschließt. In der Untersuchung wurde festgestellt, daß bei einigen Ansätzen eine Validierung möglich ist, Zertifizierungen von Sicherheitsarchitekturen liegen aber noch nicht im Bereich der praktischen Realität. Gleichzeitig sind große Defizite bei der Unterstützung der Verwendung zertifizierter Bausteine zu verzeichnen. Die meisten der untersuchten Ansätze verhalten sich neutral gegenüber der eingesetzten Sicherheitspolitik.

Das bedeutet aber auch, daß das Konzept der mehrseitigen Sicherheit nicht befriedigend und nicht durchgängig unterstützt wird. Es konnten - außer in SEMPER und PLASMA - keine Ansätze für einen Aushandlungsprozeß zwischen den verschiedenen Politiken der Kommunikationspartner gefunden werden. Diese Aushandlung ist jedoch eine wesentliche Bedingung für die Realisierung mehrseitiger Sicherheit in Kommunikationssystemen.

Die angebotene Funktionalität der Systeme kann als zum Teil befriedigend bezeichnet werden. So existieren Ansätze für die Einbindung spezieller (manipulationssicherer) Hardware, jedoch nicht durchgängig. Einige Systeme bieten die Möglichkeit, die angebotene Funktionalität auf spezielle Bedürfnisse der Nutzer abzustimmen. Ein deutliches Defizit liegt derzeit noch in der Unterstützung von Anonymitätskonzepten; sie sind nur in wenigen Systemen realisiert. Weiterhin kann eingeschätzt werden, daß es bereits gute Sammlungen von kryptographischen Mechanismen und Sicherheitsdiensten gibt. Wünschenswert wäre aber eine durchgängige Realisierung von asymmetrischen Krypto-Verfahren und solchen zur Zertifizierung. Auch werden die vorhandenen Mechanismen auf unterschiedliche Weise (anwendungsspezifisch oder generisch) eingesetzt. Dabei kann es zur Festschreibung unterschiedlicher Sicherheitspolitiken kommen, die so nicht wünschenswert ist.

Als Schlußfolgerungen für Sicherheitsarchitekturen, die die mehrseitige Sicherheit der Nutzer unterstützen sollen, ergibt sich daraus zunächst die Notwendigkeit der Realisierung eines Aushandlungsprozesses, in dem die verschiedenen Sicherheitsanforderungen der Kommunikationspartner formuliert und eine gemeinsame Basis zur Kommunikation ausgehandelt werden kann. Weiterhin ist es von großer Bedeutung, daß Nutzer durch die Architektur in die Lage versetzt werden, deren Funktionalität auf ihre Bedürfnisse anzupassen, anonym zu kommunizieren und manipulationssichere Hardware einzusetzen. Notwendig ist nicht zuletzt der Aufbau einer geeigneten Infrastruktur, um den Einsatz von Zertifikaten zu ermöglichen.

3 Sicherheitsarchitektur

In den folgenden Abschnitten werden eigene Konzepte für eine Sicherheitsarchitektur zur Ermöglichung mehrseitiger Sicherheit zwischen Kommunikationspartnern dargelegt. Bei der Erarbeitung dieser Architektur werden insbesondere die bei der Analyse in Kapitel 2 gewonnenen Erkenntnisse berücksichtigt.

3.1 Rollen und Komponenten

Aus Gründen der Komplexitätsreduzierung wird die Architektur aus klar voneinander abgrenzbaren Komponenten aufgebaut. Jeder Komponente kann jeweils eine Teilaufgabe zugeordnet werden, die sie bei der Erfüllung der Schutzziele des Nutzers innehat und die mit einer oder mehreren Rollen zusammenhängen kann. Die Rollen, Komponenten und ihr Zusammenwirken werden im folgenden näher beschrieben.

Rollen

Anwendungsentwickler sollen durch die Sicherheitsarchitektur insofern unterstützt werden, daß sie bei der Entwicklung von Anwendungen Sicherheitsdienste unabhängig vom speziellen Algorithmus eines konkreten Sicherheitsmechanismus einbinden können. So wird dem Entwickler eine Auswahl von Sicherheitszielen auf einer abstrakten Ebene angeboten werden (z.B. Integrität, Schlüsselverwaltung, etc.), aus der er die Ziele auswählen kann, die die zu implementierende Anwendung erfüllen soll.

Nutzer sind Personen, die das Interesse haben, die Arbeit mit Anwendungen, insbesondere bei der Kommunikation über Systemgrenzen hinaus, ihren Sicherheitsanforderungen gemäß zu gestalten.

Als *Kommunikationspartner* wird im Folgenden derjenige Nutzer bezeichnet, der das entfernte Rechnersystem bedient, mit dem die Anwendung des Nutzers kommunizieren wird. Somit ist der Kommunikationspartner auch derjenige, mit dem bzw. dessen System die globale Aushandlung, also ein Abgleich der durch die Sicherheitsmechanismen repräsentierten Sicherheitsziele, durchgeführt wird.

Komponenten

Anwendung: Verteilt oder lokal laufende Applikation, die zur gesicherten Ausführung von Aktionen und Befehlen die Sicherheitsarchitektur nutzt. Sie ist eine Schnittstelle zum Nutzer, die diesem alle Informationen kontrollierbar, auswählbar und sicher darstellt, sowie „sichere“ Eingaben ermöglicht.

Protokolle sind Abarbeitungsreihenfolgen, die durch die Kombination von allgemeinen Diensten und von durch die Sicherheitsarchitektur bereitgestellten Sicherheitsdiensten definiert werden. Diese Protokolle können durch die Konfiguration und Aushandlung parametrisiert werden.

Sicherheitsdienste sind abstraktere Zusammenfassungen von Sicherheitsmechanismen, also Abstraktionen von konkreten kryptographischen Algorithmen und Funktionen. So können zum Beispiel die Verfahren DES und IDEA in die Klasse „symmetrische Verschlüsselung“ eingeordnet werden.

Sicherheitsmechanismen: Die Sicherheitsarchitektur wird eine Sammlung an kryptographischen Algorithmen und Funktionen, wie z.B. RSA, DES, IDEA, bereitstellen, die sich unter bestimmten Gesichtspunkten einem oder mehreren Sicherheitsdiensten zuordnen lassen.

In der *Konfigurationsdatenbank* werden die Konfigurationen (die Grundkonfiguration als auch Anwendungs- und Verbindungskonfiguration) gespeichert. Diese Datenbank enthält verschiedene Einstellungen für die erforderlichen Protokolle, Sicherheitsdienste und -mechanismen. Sie bildet somit deren Informationsschnittstelle und fungiert auch für die Aushandlungskomponente als solche.

Der *Konfigurationsmanager* erlaubt dem Nutzer, eine interaktive Konfigurierung, d.h. eine Grundeinstellung der Konfigurationsdatenbank vorzunehmen, indem er eine Grundkonfiguration eingibt, womit er das Verhalten von Sicherheitsdiensten und -mechanismen seinen Anforderungen entsprechend vordefinieren kann.

Die *Aushandlungskomponente* ist neben der Konfigurierungsmöglichkeit der wesentliche Teil der Sicherheitsarchitektur für die Realisierung mehrseitiger Sicherheit. Sie ermöglicht dem Nutzer, seine Interessen gegenüber dem jeweiligen Kommunikationspartner in einem Aushandlungsprozeß zu vertreten. Im erfolgreichen Fall kann für die Kommunikationsbeziehung trotz unterschiedlicher, eventuell auch teilweise konkurrierender Sicherheitsanforderungen eine gemeinsame Basis zur sicheren Kommunikation ausgehandelt werden. Als Ergebnis der Aushandlungsphase wird eine Konfiguration in die Konfigurationsdatenbank geschrieben.

Das *Betriebssystem* stellt keine direkte Komponente der Sicherheitsarchitektur dar. Es steht aber insofern in engem Zusammenhang, als es bestimmte Anforderungen erfüllen muß, die durch den Einsatz einer Sicherheitsarchitektur gefordert werden. Können diese Anforderungen durch das Betriebssystem nicht erfüllt werden, kann nicht verhindert werden, daß die Sicherheitsarchitektur unterlaufen werden kann und somit ihre Funktionalität außer Kraft gesetzt würde.

3.2 Zusammenspiel der Rollen und Komponenten

Sobald eine Anwendung mit der Unterstützung der Sicherheitsarchitektur implementiert worden ist, ist diese dazu in der Lage, zur Laufzeit die Funktionalität der Sicherheitsarchitektur zu nutzen. Zu der lauffähigen Anwendung wurde eine den Empfehlungen des Entwicklers entsprechende Beschreibung von *Anwendungsforderungen* bezüglich der zu verwendenden Sicherheitsdienste mit abgespeichert. Diese Anwendungsforderungen stellen ein Mindestmaß an Sicherheitsmechanismen, -diensten und Protokollen dar, die zur Sicherheit und zum Schutz des Nutzers benutzt werden sollten.

In ähnlicher Weise muß der Nutzer, bevor er eine Anwendung startet, eine *Grundkonfiguration* für die seine Sicherheitsinteressen repräsentierenden Sicherheitsmechanismen und -dienste eingeben, die für alle Anwendungen gleichermaßen gilt. Will der Nutzer eine Anwendung starten, ohne vorher eine Grundkonfiguration in die Konfigurationsdatenbank eingegeben zu haben (die Konfigurationsdatenbank ist also leer), wird der Konfigurationsmanager für den Nutzer automatisch aufgerufen und eine Abfrage der gewünschten Einstellungen durchgeführt.

Beim ersten Start der Anwendung übergibt diese ihre Anwendungsforderungen an die Aushandlungskomponente, die die lokale Aushandlung, also einen Abgleich zwischen der Grundkonfiguration des Nutzers und den Anwendungsforderungen, durchzuführen versucht. Anwendungen können nur über die Aushandlungskomponente Einfluß auf die Konfiguration nehmen, also nie direkt auf die Konfiguration schreiben.

Das Ergebnis der erfolgreichen lokalen Aushandlung bildet die *Anwendungskonfiguration*, die getrennt von der Grundkonfiguration in der Konfigurationsdatenbank abgelegt wird. Die Anwendungskonfiguration ist im Gegensatz zur Grundkonfiguration nicht für alle Anwendungen gültig, sondern wird nur für jeweils eine einzelne Anwendung durchgeführt.

Dementsprechend befindet sich pro Anwendung nach der lokalen Aushandlung eine Anwendungskonfiguration in der Konfigurationsdatenbank.

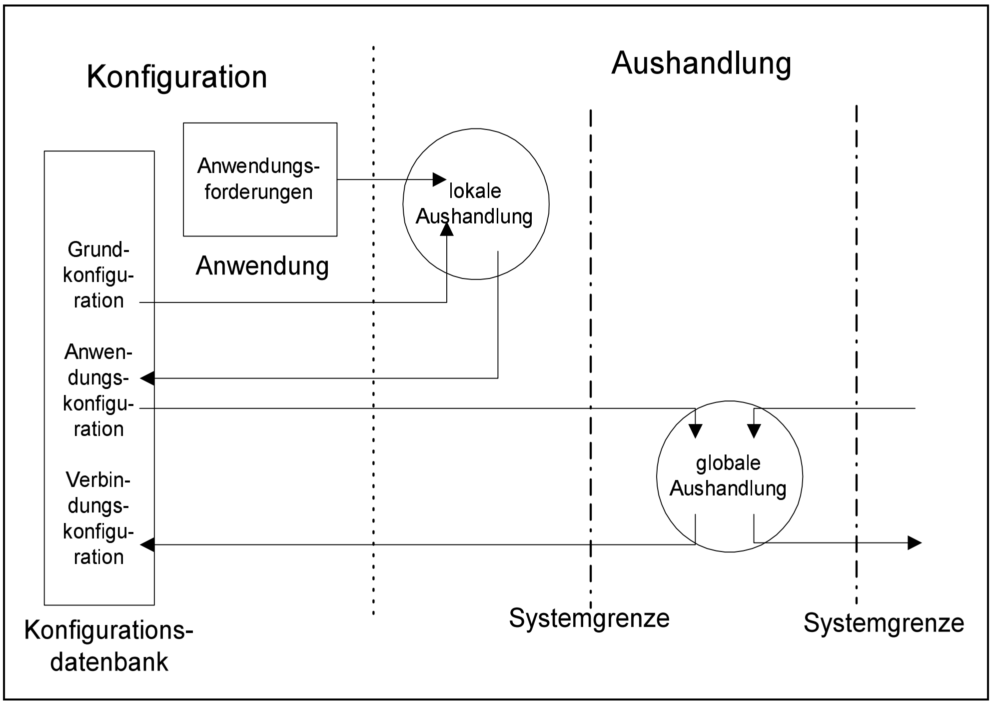

Abbildung 3.1: Zusammenspiel von Konfiguration und Aushandlung

Nach der erfolgreichen lokalen Aushandlung sind zwei (oder mehrere) potentielle Kommunikationspartner in der Lage, eine globale Aushandlung durchzuführen. Dazu werden - wie in der Abbildung 3.1 dargestellt - die Anwendungskonfigurationen von allen Beteiligten aus der Konfigurationsdatenbank an die Aushandlungskomponente übergeben. Diese versucht nach einem speziellen Protokoll, zwischen den Sicherheitsinteressen zu vermitteln und so eine *Verbindungskonfiguration* zu erreichen. Es werden zwei Stufen der Aushandlung unterschieden:

- Automatische Aushandlung: Falls die Obermenge der Sicherheitsanforderungen, die in den Anwendungskonfigurationen erfaßt sind, beiden potentiellen Kommunikationspartnern zur sicheren Kommunikation ausreicht, werden sie lediglich über das Aushandlungsergebnis informiert. Sie können diesem - als Grundlage der folgenden Kommunikation - zustimmen.

- Interaktive Aushandlung: Falls kein für beide bzw. alle Seiten akzeptables Ergebnis entsteht, wird mittels Interaktionen zwischen Nutzer und Aushandlungskomponente erlaubt, die Konfiguration zu ändern. Auf diese Weise soll eine aus Sicht aller Kommunikationspartner befriedigende Verbindungskonfiguration ausgehandelt werden, wobei die jeweils eigenen Sicherheitsinteressen gewahrt bleiben sollen. Sind die Nutzer mit den ihnen gebotenen Alternativen zu keiner „sicheren“ Kommunikation bereit, so wird die Aushandlung abgebrochen. Die Anwendungskonfiguration bleibt unverändert erhalten. Es kann keine Verbindungskonfiguration in die Konfigurationsdatenbank eingetragen werden.

In Tabelle 3.1 ist der zeitliche Ablauf der Aushandlung noch einmal im Überblick dargestellt.

Schritt	Ausführung durch	Aktion	Ergebnis
1	Anwendungsentwickler	Definition von Anwendungsforderungen bei Anwendungsentwicklung	Anwendungsforderungen in Anwendung
2	Nutzer	Eingabe der Grundkonfiguration mittels Konfigurationsmanager	Grundkonfiguration in Konfigurationsdatenbank
3	Nutzer (Anwendung)	Start der Anwendung (Übergabe der Anwendungsforderungen)	Anwendungsforderungen in Aushandlungskomponente
4	Aushandlungskomponente	lokale Aushandlung (Abgleich von Grundkonfiguration und Anwendungsforderungen)	Anwendungskonfiguration in Konfigurationsdatenbank oder Abbruch der Anwendung
5	Aushandlungskomponente bzw. Nutzer und Kommunikationspartner	globale Aushandlung (Aushandlung mit Kommunikationspartner)	Verbindungskonfiguration in Konfigurationsdatenbank oder Abbruch der Aushandlung

Tabelle 3.1: Zeitlicher Ablauf von Konfigurierung und Aushandlung

Nach einer beendeten erfolgreichen oder nicht erfolgreichen automatischen Aushandlung und natürlich auch bei Beendigung der interaktiven Aushandlung wird der Nutzer durch die Aushandlungskomponente über das Aushandlungsergebnis benachrichtigt.

Aushandlungsergebnisse können temporären oder permanenten Charakter besitzen, wobei temporär gültige Konfigurationen jeweils nach Beendigung der Anwendung bzw. dem Verbindungsabbau aus der Konfigurationsdatenbank entfernt werden, permanente Konfigurationen dagegen längerfristig in der Datenbank abgespeichert werden.

Anwendungskonfigurationen besitzen demnach permanenten Charakter. Ähnlich könnten Verbindungskonfigurationen, die einem bestimmten Kommunikationspartner zugeordnet werden, als eher permanent gelten, da z.B. innnerhalb eines gewissen Zeitraumes wieder mit dem gleichen Partner kommuniziert werden könnte.

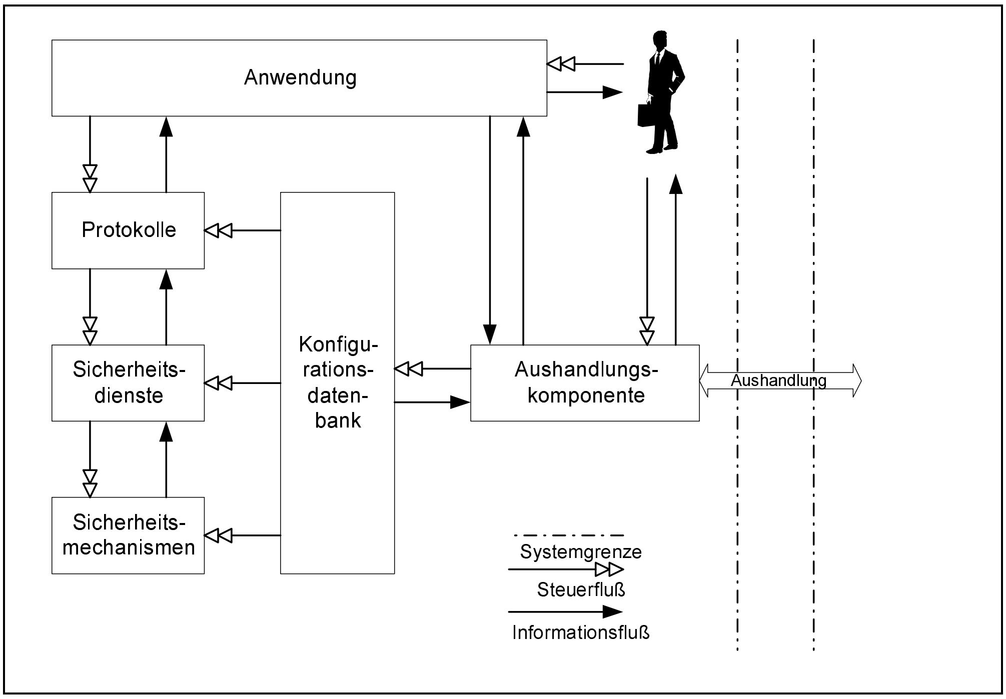

Abbildung 3.2: Zusammenspiel der Komponenten bei Anwendungsausführung

Nachdem die zwischen den Kommunikationspartnern ausgehandelte Verbindungskonfiguration in der Konfigurationsdatenbank abgelegt worden ist, kann die Anwendung mit der Kommunikation beginnen. Erst ab diesem Zeitpunkt kommen die Sicherheitsmechanismen, -dienste und Protokolle zum Tragen (vgl. Abbildung 3.2). Sie werden nun von der Anwendung aufgerufen und dieser entsprechend der Parametrisierung in der Verbindungskonfiguration durch die Sicherheitsarchitektur zur Verfügung gestellt.

Dabei setzen sich Sicherheitsdienste primär aus Sicherheitsmechanismen und Protokolle aus Sicherheitsdiensten zusammen. Das bedeutet aber nicht, daß auf der Ebene der Sicherheitsdienste zwingend nur Mechanismen der darunterliegenden Ebene eingebunden werden können, sondern es können auch wieder direkt Funktionen der gleichen Ebene aufgerufen werden.

3.3 Anwendung zur Validierung

Zur Veranschaulichung des Funktionsprinzips der erarbeiteten Sicherheitsarchitektur und zur Demonstration des Zusammenwirkens der einzelnen Komponenten wird z.Zt. ein Demonstrationsprogramm entwickelt. Dieses zeigt anhand des Anwendungsbeispiels Teleshopping das Zusammenspiel von drei verschiedenen Teilnehmern mit unterschiedlichen Interessen: Kunde, Bank und Händler. Dabei liegt der Schwerpunkt auf dem Darstellen des Ablaufes der Interaktionen zwischen den Teilnehmern gemäß den Protokollen der

Sicherheitsarchitektur. Der Teleshopping-Demonstrator ist als erster Ansatz einer Anwendung der prototypischen Implementation der erarbeiteten Sicherheitsarchitektur zu verstehen. Mit Hilfe des Demonstrators soll es möglich sein, das beschriebene Sicherheitskonzept praktisch zu untermauern und zu zeigen, daß es tatsächlich realisierbar ist. Erste Erfahrungen, die bei der Validierung der Sicherheitsarchitektur anhand des Teleshopping-Demonstrators gesammelt wurden, werden in die weitere Gestaltung der Architektur einfließen. Der Demonstrator soll mögliche Engpässe in der Implementierung aufzeigen, die durch entsprechende rechtzeitige Überarbeitung des Entwurfes beseitigt werden können.

4 Ausblick

Verhandlung zwischen den Sicherheitsanforderungen von Kommunikationspartnern ist nicht nur wünschenswert, sondern auch technisch-organisatorisch notwendig, wenn mehrseitige Sicherheit gewährleistet werden soll. Es ist nicht zu erwarten, dass sich ein de-facto-Standard durchsetzt, der Aushandlung vorsieht. Für die Verfeinerung der Architektur für mehrseitige Sicherheit zeichnet sich ab, daß die Architekturkomponenten zur Konfiguration und Aushandlung eine zentrale Stellung einnehmen. Hier muß noch wesentliche konzeptionelle Arbeit geleistet werden. Als Grundlage ist zu erarbeiten, wie das Protokoll der globalen Aushandlung aussehen wird. Dabei stellt sich besonders die Frage, ob ein Kommunikationspartner seine Sicherheitsanforderungen zuerst dem Anderen zur Verfügung zu stellen hat. Ein möglicher Lösungsansatz zur Vermeidung eines diesbezüglichen Ungleichgewichtes ist der Einsatz eines Commitment-Schemas [Schn_96, S. 86f]. Ein weiterer Diskussionspunkt ist, welche Konfigurationsdaten dem Kommunikationspartner grundsätzlich zur Verfügung gestellt werden sollen. Es ist z.B. vorstellbar, daß die Nutzer jeweils ein Minimum, Maximum und ein Optimum an Sicherheitsforderungen formulieren könnten. Grundlage dafür wäre eine vertiefte Diskussion über die Konfigurationsdaten, die durch den Konfigurationsmanager erfaßt und verwaltet werden. Daneben sind noch Metaregeln und atomare Elemente zu definieren, die die Architektur mit ihren Komponenten strukturiert aufbauen. Es ist noch näher zu umreißen, was Sicherheitspolitik im Zusammenhang mit der konzipierten Sicherheitsarchitektur bedeutet, welchen Wirkungs- und Einflußbereich sie jeweils hat und wie z.B. die Aushandlungsstrategien der Nutzer im einzelnen aussehen können.

Um die erarbeiteten Konzepte in effektiver Form validieren zu können, umfassen die weiteren Arbeiten am Teleshopping-Demonstrator im wesentlichen die Erweiterung um sicherheitsrelevante Funktionalität, wie sie von der Sicherheitsarchitektur unterstützt werden soll. Das beinhaltet in erster Linie die Implementierung eines Konfigurationsmanagers und einer Komponente zur Realisierung eines Aushandlungsprozesses.

Aus der Untersuchung anderer Szenarien neben dem Teleshopping ergeben sich weitere Problemfelder, für die es gilt, mehrseitig sichere Lösungen zu entwickeln. Um die angestrebte Allgemeingültigkeit der Sicherheitsarchitektur überprüfen zu können, ist geplant, Szenarien im Bereich des Teleteaching als weitere Validierungsbeispiele für die Konzepte der Sicherheitsarchitektur heranzuziehen.

5 Literatur

[BKMS_96] I. Biehl, H. Kenn, B. Meyer, J. Schwarz, C. Thiel. LISA - Library for Secure Applications. Universität des Saarlandes, 1996

[Cust_95] H. Custer. Inside Windows NT. Microsoft Press, Redmond, 1995

[EKPT_96] W. Eßmayr, F.Kastner, G. Pernul, A. M. Toja. The security architecture of IRO-DB. In *Proc. 12th Int. Information Security Conference (IFIPSEC)*, Greece, 1996

[FaPe_96] J. Fabre, T. Pérennou. FRIENDS: A Flexible Architecture for Implementing Fault Tolerant and Secure Distributed Applications. At *Second Dependable Computing Conference*, Italy, 1996

[GeKo_95] M. Gehrke, E. Koch. A Security Platform for Future Telecommunication Applications and Services. In *Proc. of the 6th Joint European Networking Conference (JENC6)*, Israel, 1995, http://www.igd.fhg.de/www/igd-a8/pub/pub.html

[HäKK_92] H. Härtig, O. Kowalski, W. Kühnhauser. The BirliX Security Architecture. In *Journal of Computer Security*, 2(1). 5-21, 1993, http://os.inf.tu-dresden.de/~haertig/bsa.ps

[Hugh_96] L. J. Hughes, Jr. Actually Useful Internet Security Techniques. New Riders Publishing, Indianapolis, USA, 1996

[Kann_94] R. Kanne. Eine objektorientierte Klassenbibliothek für kryptographische Systeme. Diplomarbeit, Universität Hildesheim, 1994

[Kran_96] A. Krannig. PLASMA - Platform for Secure Multimedia Applications. In *Proceedings Communications and Multimedia Security II*, Essen, 1996

[Linn_93] J. Linn. Generic Security Service Application Program Interface. RFC 1508, 1993

[Lupp_95] A. Lupper. Distributed Operating Systems - BirliX. 1995, http://www-vs.informatik.uni-ulm.de/DOSinWWW/DOS.html

[MPSC_93] S. Muftic, A. Patel, P. Sanders, R. Colon u.a.. Security Architecture for Open Distributed Systems. Wiley, Chichester 1993

[NSA_95] NSA Cross Organization CAPI Team. Security Service API. Cryptographic API Recommendation. NSA, 1995

[NeSc_78] R. M. Needham, M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. Communications of the ACM 21/12 (1978)

[OMG_95] OMG. CORBA Security. Document Number 95-12-1. December 1995

[PiRi_94] J-M. Piveteau, H. P. Rieß. Eine generische Sicherheitsarchitektur für Telekommunikationsnetze. In *Proc. Der Fachtagung Sicherheit in Informationssystemen*, Schweiz, 1994

[Plas_1] PLASMA - Platform for Secure Multimedia Applications. In *DeTeBerkom: Security - a Cornerstone of the Information Society* http://www.deteberkom.de/projekte/texte/Sec.eng.html

[Plas_2] PLASMA - Plattform für sichere Multimedia-Anwendungen. http://www.igd.fhg.de/www/igd-a8/projects/plasma/PLASMA-d.html

[Schi_97] A. Schill. DCE - Das OSF Distributed Computing Environment: Grundlagen und Anwendung (2., erweiterte Auflage); Springer-Verlag, 1997

[Schn_96] B. Schneier. Applied Cryptography. 2nd Edition. John Wiley & Sons, New York, 1996

[Secu_96] SecuDE - Ein portables Sicherheitstoolkit. Informationsblatt, GMD, 1996

[SET_96] Secure Electronic Transaction (SET) Specification. Book 2: Technical Specifications. http://www.visa.com/cgi-bin/vee/sf/set/intro.html, Feb. 1996

[Sher_93] M. Sherman. Distributed Transaction Processing with Encina. In *Parallel and Distributed Information Systems*, IEEE Computer Society Press, Los Alamitos, Ca., USA, 1993

[Waid_96] M. Waidner. Development of a Secure Electronic Marketplace for Europe. In *E. Bertino, H. Kurth, G. Martella, E. Monolivo: „Computer Security - ESORICS 96*“, Springer, Berlin 1996, 1-14

[Wiew_96] E. Wiewall. Secure Your Applications with the Microsoft CryptoAPI. In *Microsoft Developer Network News*, 3/4 1996, Microsoft Press

[WWZ_96] G. Wicke, G. Wolf, J. Zöllner. Sicherheitskonzepte am Beispiel konkreter Anwendungen - 1. Zwischenbericht. Projektbericht SSONET. TU Dresden, August 1996