Sicherheit in CORBA und TINA

U.G. Wilhelm, S. Staamann, G. Wolf und J. Zöllner

1 Einleitung

Immer mehr für den Betrieb von Unternehmen kritische Anwendungen werden als verteilte Systeme realisiert. Dies geschieht zum Beispiel im Rahmen der Umsetzung des Client/Server-Paradigmas, um Anwendungen besser skalierbar oder auf kostengünstigerer Hardware lauffähig zu machen. Darüberhinaus gewinnen inhärent verteilte Anwendungen, wie etwa Buchungssysteme oder Telekommunikationsanwendungen, mehr und mehr an Bedeutung.

Wurden letztere in der Vergangenheit meist auf der Basis von nicht-öffentlichen proprietären Protokollen und Architekturen entwickelt, so geht der Trend dahin, diese auf leistungsfähige offene Plattformen, wie sie für Client/Server-Anwendungen genutzt werden, zu portieren.

Insbesondere im Bereich der Telekommunikationssysteme nimmt die Nachfrage nach neuen komplexen Diensten (wie etwa Telekonferenzsysteme, Universal Personal Telecommunications (UPT) oder mobiler Kommunikation) ständig zu. Um diese Nachfrage der Nutzer zu befriedigen, wird eine flexiblere Infrastruktur für Kommunikationsdienste benötigt, in die neue Dienste einfacher, schneller und reibungsloser eingeführt werden können. Es wird weithin angenommen, daß der Übergang von proprietären zu offenen Plattformen für verteilte Systeme dies leisten kann, da dort aufgrund der breiteren Nutzerbasis leistungsfähigere Entwicklungswerkzeuge zur Verfügung stehen und diese aufgrund ihrer Offenheit eine bessere Interoperabilität bieten.

Ein ähnlicher Trend ist auch in klassischen Einsatzgebieten der Computer-Kommunikation sichtbar, in denen vormals proprietäre Systeme den Endnutzern direkt zugänglich gemacht werden sollen (etwa direkter Zugriff auf Flugbuchungssysteme).

Da es sich hierbei oft um sicherheitskritische Anwendungen handelt, müssen diese geschützt werden. Ein solcher Schutz war in zentralisierten Systemen noch relativ einfach zu erreichen, da der Zugang zu diesen Systemen physisch kontrolliert werden konnte. In inhärent verteilten Anwendungen wurde ebenfalls – mit mehr oder weniger großem Erfolg – versucht, diese physische Zugriffskontrolle so weit als möglich zu erhalten, indem diese nicht an öffentliche Kommunikationsmedien gekoppelt wurden und zusätzlich ihre Funktionsweise weitgehend geheimgehalten wurde. Eine solche Sicherheit durch Verbergen *(Security by obscurity)* war eigentlich im Falle von physikalisch nicht geschützten Übertragungswegen schon damals leicht zu unterlaufen [For94].

Die angeführten Sicherheitsmaßnahmen reichen beim Übergang zu offenen Systemen, in denen potentiell alle Objekte miteinander kommunizieren können, und in denen die Protokolle für die Interaktion zwischen Objekten allgemein bekannt sind, keinesfalls aus. Stattdessen muß eine Sicherheitsarchitektur definiert werden, die trotz völliger Offenheit des Systems die Sicherheit der Objekte und ihrer Daten garantieren kann. Eine solche Sicherheitsarchitektur kann nur auf kryptographischen Verfahren und einem entsprechenden Schlüsselmanagement basieren.

2 Relevante Architekturen

Wir wollen uns im folgenden stellvertretend mit zwei Architekturen auseinandersetzen, CORBA als einem Beispiel für eine Kommunikations-Middleware für verteilte Systeme und TINA als einem vielversprechenden Ansatz für die zukünftige Entwicklung von Telekommunikationssystemen. Andere Systeme wie etwa DCE (Distributed Computing Environment) [OSF92, RKF92] oder IN (Intelligent Networks) [ITU93b, GRKK93] hätten ebenfalls eine Untersuchung verdient, was allerdings den zur Verfügung stehenden Rahmen sprengen würde. Jedoch finden sich in TINA grundlegende Konzepte aus DCE und IN wieder.

2.1 CORBA

Der *CORBA* (Common Object Request Broker Architecture) Standard [OMG95b] ist eine von der *Object Management Group* (OMG) definierte Architektur zur Konstruktion und Integration von objektorientierten Client/Server-Applikationen in heterogenen Netzwerkumgebungen. Erklärtes Ziel ist es dabei, die Verteilung und Kooperation von Softwarebausteinen auf verschiedenen Systemplattformen zu ermöglichen und damit die Portabilität, Interoperabilität und Wiederverwendbarkeit von Software zu maximieren. Dazu sollte eine Basisarchitektur für verteilte, objektorientierte Anwendungen definiert und standardisiert werden, die sogenannte *Objekt Management Architecture* (OMA) [OMG95c]. Diese unterteilt eine allgemeine verteilte Anwendung in die folgenden vier Bereiche (Abbildung 1):

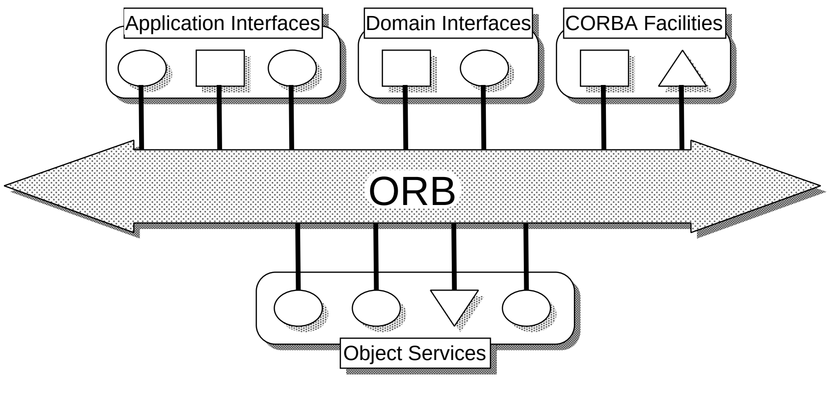

Figure 1: Die Objekt Management Architecture

- Object Request Broker (ORB): der Kern von CORBA; näheres hierzu in Abschnitt 2.1.2;

- Object Services: die Grundelemente, die für das Funktionieren eines verteilten Systems nötig sind (z.B. Nameserver, Ereignismanagement, Transaktionen oder Persistenz);
- Common Facilities: eine Anzahl von generischen Objekten für immer wiederkehrende Probleme (z.B. Benutzerschnittstelle, Datenbankeneinbindung oder Dokumentenmanagement);
- Domain Interfaces: standardisierte Lösungen für Probleme in speziellen Anwendungsbereichen (z.B. Telekommunikation, Finanzwirtschaft oder Managementanwendungen);
- Application Interfaces: die spezialisierten Objekte die die eigentliche Anwendung ausmachen.

Im Jahr 1991 wurde die CORBA-Version 1.1 verabschiedet, die Client/Server Objekt-Interaktionen ermöglichte. Die seit Ende 1994 aktuelle Version 2.0 erweitert die Interoperabilität von Objekten und spezifiziert, auf welche Weise verschiedene ORBs, von möglicherweise verschiedenen Herstellern, miteinander kooperieren.

2.1.1 Interoperabilität von Objekten

Eines der wichtigsten Ziele von CORBA ist, die Austauschbarkeit und Kooperation von Objekten zu ermöglichen, unabhängig von: Lokation der Objekte im Netzwerk, Hardware und Betriebssystem auf denen sie ausgeführt werden, unterschiedlichen internen Datenformaten, Implementierungssprachen und Netzwerkprotokollen.

Dazu ist es notwendig, die Schnittstelle eines Objekts noch stärker von der Implementierung zu trennen, als dies in der üblichen objektorientierten Programmierung der Fall ist. Üblicherweise ist die Schnittstelle in der selben Programmiersprache realisiert wie die eigentliche Implementierung. In CORBA wurde für Objektschnittstellen mit der *CORBA-IDL* (Interface Description Language) ein standardisiertes Format geschaffen, welches von der implementierungsspezifischen Schnittstelle abstrahiert. In Abbildung 2 ist diese Trennung zwischen Schnittstelle und Implementierung eines Objektes dargestellt.

Mit Hilfe der IDL ist es möglich, Objektverhalten und -funktionalität unabhängig von der verwendeten Programmiersprache und Plattform zu beschreiben. Die Erstellung der IDL-Spezifikation für ein Objekt ist Aufgabe desjenigen, der das Objekt und dessen Methoden entwirft und implementiert. Um diese IDL-Schnittstelle des Servers zu nutzen, kann der Client zwischen zwei verschiedenen Varianten wählen:

- statische Aufrufe mittels IDL-Compiler oder
- dynamische Aufrufe über die Schnittstellen-Verwaltung.

Bei der statischen Variante wird aus der IDL-Spezifikation eines Server Objektes von einem speziellen Compiler der Code zum transparenten Zugriff eines Clients auf die Server-Objekte generiert. Das geschieht z.B. durch Umsetzung der Spezifikation in eine C++-Klassendefinition. Die Einbindung dieser Klassendefinition in den Client

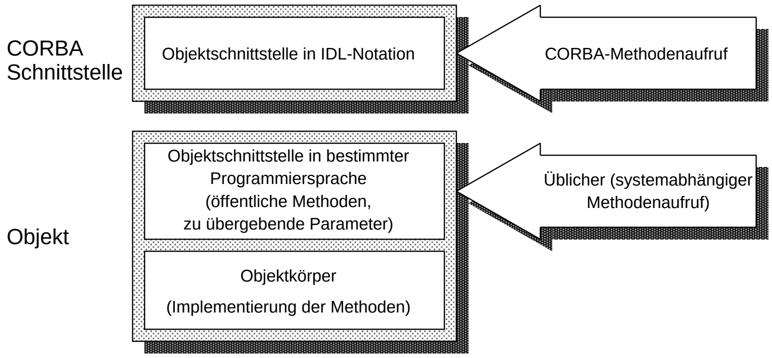

Figure 2: Objekt mit Schnittstellen und Implementierung

geschieht während der Compilierung des Clients. Die zweite Variante kann alternativ zur ersten oder zusätzlich verwendet werden. Sie bietet den Vorteil, daß hier die Informationen über vorhandene Schnittstellen zu benötigten Server-Objekten durch den Client erst zur Laufzeit ermittelt werden müssen.

Beide Verfahren arbeiten eng mit dem ORB zusammen (siehe Abschnitt 2.1.2), der die IDL-Schnittstellen in einer speziellen Datenbank (dem Interface Repository) ablegt und allen Clients zugänglich macht.

2.1.2 Der ORB als Vermittlungszentrale für Objektaufrufe

Der *ORB* (Object Request Broker) ist ein zentrales Element der CORBA-Architektur und vor allem dafür zuständig, dem Client die Lokalisierung von Objekten und Diensten zu erleichtern oder ganz abzunehmen, sowie Objektaufrufe zu vermitteln. Von CORBA werden alle Funktionen, Dienste und Schnittstellen definiert, die der ORB unterstützen muß, um einen solchen Objektaufruf zu ermöglichen.

Die Auswahl des Zielobjektes geschieht im allgemeinen mittels einer Objektreferenz, die der Client entweder von einem speziellen Dienst (Nameserver, Broker oder Service Factory) oder einem anderen Objekt erhält. Eine solche Objektreferenz ist unabhängig von der Lokation des Clients und bezeichnet eindeutig das Zielobjekt (Server-Objekt), welches sich ebenfalls an einer beliebigen Stelle im CORBA-System befinden kann. Der Client kann nun die vom Zielobjekt in dessen IDL-Spezifikation definierten Methoden aufrufen. Die Funktionalität sowie die externe Schnittstelle des Server-Objektes wurden dem ORB bei der Registrierung mitgeteilt und dort (im Interface Repository) gespeichert. Dem Zielobjekt übermittelt der ORB die auszuführende Methode und die Parameter. Nach der Ausführung durch das Zielobjekt nimmt der ORB die Ergebnisse entgegen (oder unter Umständen auch eine Ausnahmebedingung (Exception)) und reicht sie an den Client zurück (siehe Abbildung 3). Das Zielobjekt selbst ist ein gekapseltes Objekt, welches seine Dienste ausschließlich über seine IDL-Schnittstelle anbietet.

Ein ORB kann Aufrufe auch an andere, ihm bekannte ORBs weiterleiten, falls ein vom Client nachgefragter Dienst nicht lokal verfügbar ist (die Föderation von ORBs

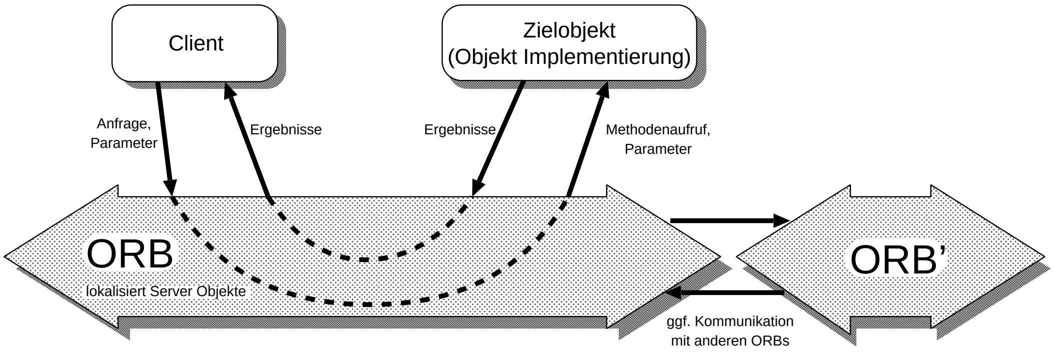

Figure 3: Der ORB als zentrales Glied der Objektkommunikation

wurde in der Version 2.0 von CORBA definiert).

Die Unabhängigkeit von Betriebssystemen und Programmiersprachen, verbunden mit der Interoperabilität verschiedener ORBs ermöglicht die Nutzung von Server-Objekten über Rechner-, Rechnerarchitektur- und Programmiersprachgrenzen hinweg. Es kann ein hoher Grad der Wiederverwendung von implementierten Objekten erreicht werden. Produkte verschiedener Hard- und Softwareproduzenten können zusammenarbeiten. Die Lokalisierungsfunktionalität ermöglicht die dynamische Änderung der Lokation von angebotenen Diensten. All dies macht CORBA auch zur geeigneten Plattform für die Entwicklung verteilter Anwendungen im Bereich der Telekommunikation, wie z.B. in TINA.

2.2 TINA

TINA (Telecommunications Information Networking Architecture) [NDC95, DNI95] beschreibt eine offene Architektur für Telekommunikationssysteme, welche die effiziente Einführung von Telekommunikationsdiensten und deren Management in zukünftigen, erwartungsgemäß breitbandigen und multimedialen, Telekommunikationsnetzen unterstützen soll. Diese Architektur wird von einem internationalen Konsortium (TINA-Consortium, TINA-C) definiert und validiert, welches sich aus namhaften Telekommunikationsunternehmen, Telekommunikationszulieferern und Computerherstellern zusammensetzt. Das Konsortium wurde im Jahr 1993 ins Leben gerufen, um möglichst schnell eine sinnvolle Basis für die anstehende Integration von verteilten Systemen und Telekommunikationssystemen zu schaffen.

Einer der wichtigsten Grundsätze in TINA ist, daß sowohl die Diensterbringung als auch die Verwaltung der Netzressourcen als softwarebasierte Anwendungen realisiert sind, die auf einer verteilten Verarbeitungsplattform, dem *Distributed Processing Environment* (DPE), ausgeführt werden.

Die in TINA beschriebene Architektur basiert auf den Prinzipien der verteilten Systeme, der Objektorientierung sowie auf anderen Konzepten und Standards aus den Bereichen Telekommunikation und Computernetze, wie Open Distributed Processing (ODP) [ISO93], Intelligent Networks (IN) [ITU93b, GRKK93], Telecommunication Management Network (TMN) [ITU93a], Asynchronous Transfer Mode (ATM) [Pry91], Distributed Computing Environment (DCE) [OSF92, RKF92] und CORBA. Der gewählte Ansatz soll die Entwicklung von wiederverwendbaren Softwarekomponenten unterstützen und den Dienstentwickler von der Komplexität der unterliegenden verteilten und heterogenen Technologien, wie unterschiedliche Hardware oder Netzwerktechnologien, isolieren.

Das technische Ziel von TINA ist die Bereitstellung von Konzepten und Prinzipien, die auf den Entwurf, die Erstellung, den Betrieb und das Management von Telekommunikationssoftware angewandt werden sollen. Diese werden dann im Rahmen von externen Projekten (*auxilliary projects*) überprüft und verfeinert.

2.2.1 Die TINA-C Architektur

Die TINA-C Architektur ist in vier technische Gebiete aufgeteilt, für die jeweils eine Architektur definiert ist. Unter einer solchen Architektur eines Problemgebietes wird dabei eine Menge von Konzepten, Prinzipien und Richtlinien für Entwurf und Aufbau des Systems verstanden. Es existieren folgende Architekturen:

- Computing Architecture: stellt die Konzepte und Prinzipien zur Verfügung, wie interagierende Softwarekomponenten (Computational Objects) spezifiziert werden und wie sie miteinander kommunizieren. Ein Distributed Processing Environment (DPE) definiert die Plattform für die verteilte Ausführung solcher Komponenten. Die Computing Architecture wird im folgenden Paragraphen 2.2.2 näher beschrieben;
- Service Architecture: liefert Konzepte und Prinzipien für Spezifikation, Analyse, Entwurf und Betrieb von dienstbezogenen Telekommunikationskomponenten. Sie wird im folgenden Beitrag [SW97] in diesem Buch näher beschrieben;
- Management Architecture: zielt darauf ab, generische Konzepte anzubieten, die auf das Management der anderen Architekturen angewandt werden können;
- Network Architecture: definiert eine Anzahl von generischen Netzwerkkomponenten, die hohe Abstraktionen der realen Resourcen darstellen, so daß diese problemlos von anderen Diensten genutzt werden können.

In den einzelnen Architekturen finden die in den anderen Architekturen definierten Konzepte, Prinzipien und Methoden Verwendung. Beispielsweise werden die Dienstkomponenten der Service Architecture mit den Mitteln der Computing Architecture beschrieben.

Eine grundlegende Neuerung in TINA gegenüber existierenden Telekommunikationsnetzen (z.B. IN) ist die Existenz eines DPE mit einheitlicher Benutzungsschnittstelle in allen dem TINA-System zugehörigen Bestandteilen (Computer, Vermittlungsrechner, Desktop Boxes, lokale Netzwerke etc.). Jede Interaktion zwischen verschiedenen Dienst-Komponenten (oder genauer zwischen den Anwendungsobjekten, die diese realisieren) geschieht gemäß einem standardisierten Protokoll (vergleichbar einem RPC) über das unterliegende DPE. Die dafür nötigen Mechanismen sind im DPE realisiert und können über die zugehörige DPE-Schnittstelle genutzt werden. Einen Überblick über die Gesamtarchitektur gibt Abbildung 4.

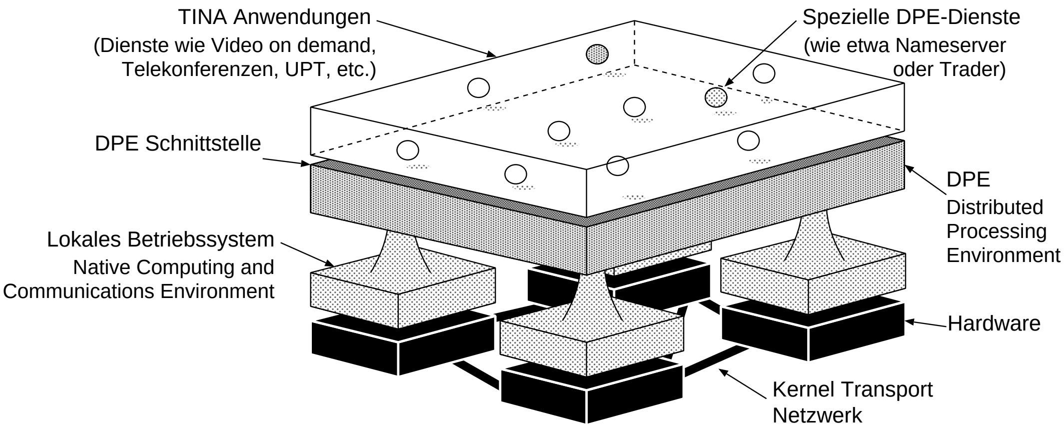

Figure 4: Die TINA-C Architektur

Das DPE selbst ist eine Kommunikations-Middleware, die auf dem lokalen Betriebssystem (native computing and communications environment) des entsprechenden Rechners aufsetzt. Um die Interaktion der Anwendungsobjekte weiter zu vereinfachen, werden in TINA zusätzliche DPE-Dienste definiert (z.B. Nameserver oder Trader), ähnlich den *object services* in CORBA.

2.2.2 Die Computing Architecture

Wie in CORBA sind auch in TINA verteilte Anwendungen aus interagierenden Objekten aufgebaut. Die Interaktion der Computational Objects geschieht über die *Computational Interfaces* der Objekte. Es existieren zwei Arten von Objekt-Schnittstellen, die *Operational Interfaces* und die *Stream Interfaces*. Ein Operational Interface ist vergleichbar einer IDL-Schnittstelle in CORBA. Es definiert eine Menge von Operationen mit Argumenten, Resultaten und Ausnahmebedingungen eines in der Serverrolle agierenden Objekts, die von einem Clientobjekt aufgerufen werden können. Ein Stream-Interface hat keine Operationen. Die Etablierung eines *Streams* zwischen zwei Stream Interfaces erlaubt den Fluß von unstrukturierter Information, wie Audio- oder Videoströme. Dabei ist ein Objekt der Erzeuger (Producer) und ein Objekt Konsument (Consumer) des Streams. Abbildung 5 zeigt die verschiedenen Interfaces.

Im Gegensatz zu CORBA kann ein Objekt in TINA nicht nur ein, sondern beliebig viele Interfaces haben. Außerdem können mehrere Objekte zu einer *Object Group* zusammengefaßt werden. In diesem Fall sind nicht mehr alle Interfaces der Mitglieder von den Objekten außerhalb der Object Group ansprechbar, sondern nur die explizit als nach außen sichtbar spezifizierten Interfaces, die sogenannten *Contracts* (Abbildung 5).

Als Sprache zur Beschreibung der Computational Objects definiert die TINA-C

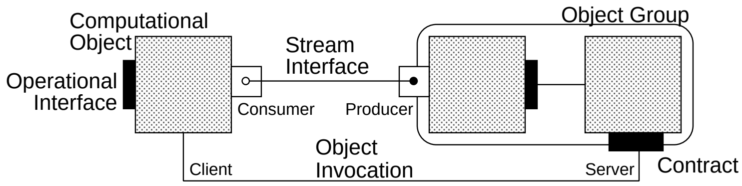

Figure 5: Die TINA Abstraktionen: Object Interfaces

Computing Architecture die *Object Definition Language* (ODL). Diese Spezifikationssprache ist eine Obermenge der CORBA-IDL. Sie ist dieser gegenüber um einige, für Telekommunikationsdienste wichtige, Konzepte, wie die Möglichkeit zur Beschreibung von Stream Interfaces, erweitert. Weiter erlaubt sie die Beschreibung von Objekten mit mehreren Interfaces und von Object Groups. Die ODL findet auch Verwendung zur Spezifikation von Dienstkomponenten, deren Abbildung auf konkrete Computational Objects oder Object Groups dem Hersteller überlassen bleibt. In diesem Fall dient die ODL als Sprache zur Spezifikation der Gesamtsicht auf die Komponente.

2.3 Beziehung zwischen CORBA und TINA

Die beiden vorgestellten Systeme sind grundsätzlich für völlig unterschiedliche Anwendungsfelder vorgesehen. CORBA ist eine anwendungsneutrale Plattform für verteilte Systeme, wohingegen TINA explizit für Telekommunikationssysteme vorgesehen ist, also für die Steuerung von Kommunikations- und Informationsdiensten. Andererseits sind sie jedoch in ihren Konzepten und hinsichtlich des Einsatzes in zukünftigen Kommunikationsnetzen eng miteinander verknüpft. Beide sind offene Systeme, explizit objektorientiert und zielen auf Interoperabilität, Portabilität und Wiederverwendbarkeit von Softwarekomponenten ab.

Das in TINA beschriebene DPE ist zwar nicht mit CORBA identisch, jedoch ist die Überdeckung hinreichend groß, so daß man davon ausgehen kann, daß eine für die Bedürfnisse von TINA erweiterte Version von CORBA die Rolle des DPE in TINA einnehmen kann, wie dies allgemein im Umfeld des TINA-Consortiums erwartet und angestrebt wird. Es gibt bereits Bestrebungen, die CORBA-IDL um die in der ODL zusätzlich vorhandenen Konzepte zu erweitern, so daß CORBA ein integraler Bestandteil von TINA werden kann. Dies hat insbesondere Auswirkungen auf die Sicherheitsarchitektur von TINA, da diese folglich auf der von CORBA aufbaut.

3 Umsetzung von Sicherheit

Im Rahmen von CORBA wurde bereits im Dezember 1995 ein erster Entwurf für eine recht umfangreiche Sicherheitsarchitektur erstellt [OMG95a]. Zwar haben viele ORB-Anbieter am Entwurf dieser Sicherheitsarchitektur mitgearbeitet, und mehrere von diesen beabsichtigen auch ihre Produkte entsprechend zu erweitern; dennoch gibt es nach dem Wissen der Autoren nur eine CORBA-Implementierung, die die Sicherheitsarchitektur zumindest teilweise umsetzt[1].

In TINA existieren erheblich komplexere Probleme, sowohl im Bereich klassischer Sicherheit, da über das System ebenfalls die Nutzungsabrechnung geregelt wird, als auch im Bereich Datenschutz (privacy protection), da in diesem System eine Vielzahl von sensiblen Daten anfallen, wie etwa die Kommunikationsbeziehungen verschiedener Teilnehmer. Bisher hat sich das Gebiet Sicherheit in TINA nicht allzuweit entwickelt. Zwar wird momentan an vielen Prototypen für TINA-konforme Systeme gearbeitet, die jedoch zumeist zur Evaluierung und Validierung der Gesamtarchitektur dienen. Keines von diesen Projekten ist konkret auf Fragen der Sicherheit spezialisiert.

3.1 CORBA Sicherheitsarchitektur

Das Hauptaugenmerk der CORBA Sicherheitsarchitektur liegt auf den klassischen Schutzzielen:

- Vertraulichkeit,
- Integrität,
- Verfügbarkeit und
- Verbindlichkeit als Option.

Diese werden für das verteilte System definiert, um Problemen wie dem Verlust von Daten, verfälschten Daten oder Aufdeckung von Daten (gegenüber Konkurrenten) entgegenzutreten. Die angeführten Probleme können dabei aus zufälliger Fehlbedienung wie auch aus offenen Angriffen resultieren.

All diese Probleme existeren nicht nur in verteilten Systemen, sondern auch in klassischen zentralisierten Systemen. Die Verletzlichkeit eines verteilten Systems ist jedoch aufgrund der weit größeren Anzahl von möglichen Angriffspunkten um vieles größer.

3.1.1 Flexibilität

Einer der wichtigsten Grundsätze der CORBA Sicherheitsarchitektur ist die Flexibilität sowohl bei der Implementierung durch den ORB-Anbieter, bei der Installation durch den ORB-Anwender als auch bei der Benutzung durch den Anwendungsprogrammierer. Deshalb wird in ihr keine bestimmte Sicherheitspolitik vorgeschrieben. Stattdessen wird ein Rahmenwerk für die Unterstützung von unterschiedlichen Sicherheitspolitiken festgelegt (eine Metadefinition). Dieser Ansatz wurde gewählt, um alle von der OMA unterstützten Sicherheitsmodelle realisieren zu können. Die Sicherheitsarchitektur ist also generisch gestaltet, so daß sie auf möglichst viele Anwendungen angepaßt werden kann; dennoch sind Schnittstellen und zugehörige IDL-Konstrukte weitgehend definiert, um die Nutzung zu vereinheitlichen.

[1] Weitergehende Informationen über DAIS von ICL sind unter `http://www.icl.com/products/dais/home.html` verfügbar.

Der ORB-Anbieter kann zwischen vielen verschiedenen Optionen für die Realisierung der Sicherheitsarchitektur wählen und sogar die Technologie von auf Sicherheit spezialisierten Anbietern in seinen ORB integrieren. Der ORB-Anwender kann durch die geeignete Definition von Sicherheitspolitiken ein Mindestmaß an Sicherheit für die Interaktion zwischen Objekten, die zu einer gewissen Sicherheitspolitikdomäne (siehe 3.1.5) gehören, festlegen. Letztendlich kann der Anwendungsprogrammierer noch über das Mindestmaß hinausgehende Sicherheitsmaßnahmen (wie etwa erweiterte Zugriffskontrolle) einbauen, wenn dies für die entsprechende Anwendung geboten scheint.

3.1.2 Principals

Die CORBA Sicherheitsarchitektur beruht auf agierenden Entitäten, sogenannten *Principals*, die bestimmte Rechte für Objektzugriffe besitzen, welche zur Laufzeit vom System überprüft werden. Ein Principal ist dabei ein menschlicher Nutzer oder eine Systemeinheit, die im System registriert und authentisierbar ist. Er kann mehrere Identitäten haben (durch Attribute repräsentiert), die für verschiedene Zwecke wie Abrechnung, Objektzugriffe oder Absenderzuordnung verwendet werden können. Die verschiedenen Identitäten sollten mit unterschiedlichen Methoden geschützt werden. Mit dieser Vielzahl von Identitäten kann eine gewisse Anonymität im System erreicht werden, da keine expliziten Verknüpfungen zwischen den Identitäten hergestellt werden können. Der Nachteil ist jedoch, daß das Gesamtsystem für einen Nutzer komplexer wird.

Sicherheitsattribute kann ein Principal auf drei Wegen erhalten:

- öffentliche Attribute sind ohne Authentikation allen Principals zugänglich;
- mittels Authentikation, z.B. Identitätsattribute und Zugriffsrechte;
- durch Delegation von anderen Principals.

Die Gültigkeit eines Attributes kann zeitlich oder anderweitig begrenzt werden, um zu kontrollieren, wann und wo es benutzt werden darf.

Ein Attribut, welches durch Delegation von einem anderen Principal stammt, kann für weitere Objektzugriffe genutzt werden. Dadurch entstehen sogenannte Aufrufketten, in denen Zugriffsrechte auf verschiedene Art und Weise delegiert werden können.

Bei der einfachen Delegation erlaubt der Client den zwischenliegenden Objekten, seine Zugriffsrechte komplett zu übernehmen, wobei das Zielobjekt keine Informationen über dazwischenliegende Objekte erhält. Bei der zusammengesetzten Delegation hingegen kann das Zielobjekt genau feststellen, welches Zugriffsrecht von welchem Objekt stammt. Andere, komplexere Schemata werden ebenfalls definiert.

3.1.3 Sicherer Objektaufruf

Objektaufrufe beinhalten sowohl auf Sender- als auch Empfängerseite bestimmte Sicherheitsmechanismen wie etwa Zugriffskontrolle, Auditing oder Schutz der Inhaltsdaten. Die Objekte müssen sich sowohl voreinander als auch dem System gegenüber authentisieren. Wenn ein Client Operationen auf einem Zielobjekt aufruft, wird eine neue Verbindung aufgebaut. Dieser Verbindungsaufbau erfolgt folgendermaßen:

- Herstellung von Vertrauen in die Identität des Kommunikationspartners;
- Übermittlung der Legitimation (incl. Sicherheitsattribute) des Clients an das Zielobjekt und Überprüfung der Sicherheitsattribute durch das Zielobjekt;
- Einrichten eines Sicherheitskontextes, der zum geforderten Schutz der Nachrichten zwischen Client und Zielobjekt verwendet wird.

Die Art des Verbindungsaufbaus hängt von den jeweiligen Sicherheitspolitiken der Kommunikationspartner und den darunterliegenden Sicherheitsmechanismen ab. Jedes Objekt ist automatisch durch die in der für dieses Objekt relevanten Sicherheitspolitik definierten Schutzmaßnahmen geschützt (siehe Sicherheitspolitik-Domäne 3.1.5); und zwar völlig unabhängig davon ob der Implementierer dieses Objektes dies vorgesehen hat oder nicht. Der Schutz wird dadurch sichergestellt, daß jegliche Interaktion eines Objektes mit einem anderen Objekt durch den CORBA-Kern geleitet wird. Dort wird getestet, welche Sicherheitspolitik für diese Interaktion angewandt werden soll, und selbige dann in der Interaktion umgesetzt. Verlangt zum Beispiel die Sicherheitspolitik einen Integritätsschutz für versandte Nachrichten, dann werden alle versandten Nachrichten abgefangen (intercepted) und mit Hilfe des vorhandenen Integritätsmechanismus in eine integritätsgeschützte Nachricht transformiert.

Welcher Mechanismus für diesen Integritätsschutz angewandt wird, ist wiederum nicht in der CORBA Sicherheitsarchitektur festgelegt, sondern in der ORB-Implementierung bzw. erst in der ORB-Installation. Eventuell nötige Initialisierungen der benötigten Datenstrukturen wurden entweder bereits in einer vorangegangenen Interaktion durchgeführt oder werden automatisch bei Bedarf vom ORB ausgeführt.

Wird eine Nachricht integritätsgeschützt übertragen, so sind keine weiteren Modifikationen der geschützten Nachrichtenteile an Gateways erlaubt (z.B. Nachrichtenlänge). Für solche Modifikationen müssen spezielle Sicherheitsgateways benutzt werden, über die keine weiteren Aussagen getroffen werden.

Alle Objektaufrufe werden mittels Zugriffskontrolle reglementiert, die auf sogenannten Zugriffsentscheidungsfunktionen basieren. Die dafür benötigten Parameter sind:

- die aktuellen Attribute des Clients,
- die Restriktionen für die Attribute (wie z.B. die Gültigkeitsdauer),
- die auszuführende Operation und
- die Kontrollattribute des Zielobjektes (z.B. ACL).

Diese Zugriffskontrolle wird ebenfalls automatisch vom CORBA-Kern durchgeführt, so daß weder der Client noch das Zielobjekt diese umgehen kann.

3.1.4 Nichtabstreitbarkeit

Dieser Sicherheitsdienst befindet sich unter der Kontrolle der Anwendungen und wird nicht automatisch bei Objektaufrufen angewandt. Er kann also nur von Applikationen verfügbar gemacht werden, die diesen Dienst kennen.

Die Nichtabstreitbarkeitsdienste, die in CORBA beschrieben sind, können sich auf verschiedenste Aktionen beziehen, nicht nur, wie im folgenden angeführt, auf Senden und Erhalt von Nachrichten. Sie umfassen die Beweiserzeugung und -überprüfung und damit folgende Funktionalität:

- Erzeugung und Überprüfung von Beweisen für eine Aktion;
- Erzeugung und Erhalt von Beweisanforderungen bezüglich einer verschickten oder empfangenen Nachricht;
- Analyse von Beweisdetails;
- Sammlung von für Langzeitspeicherung benötigten Beweisen; diese müssen eventuell umfangreicher als üblich sein.

Je nach verwendeten Sicherheitsmechanismen kann es erforderlich sein, für die Nutzung von Nichtabstreitbarkeitsdiensten dritte Instanzen hinzuzuziehen.

3.1.5 Domänen

Eine Domäne ist eine Verwaltungseinheit, innerhalb derer gewisse gemeinsame Merkmale existieren und in der die gleichen Regeln befolgt werden. Hinsichtlich Sicherheit werden drei verschiedene Typen von Domänen unterschieden:

- Sicherheitpolitik-Domäne: der Bereich, in dem eine bestimmte Sicherheitspolitik herrscht. Es kann Sub-Domänen geben (hierarchisch) und verschiedene Domänen können sich föderativ zusammenschließen.
- Sicherheitsumgebungs-Domäne: der Bereich, in dem die Durchsetzung einer bestimmten Sicherheitspolitik durch lokale Mechanismen außerhalb des Objektsystems realisiert werden kann (z.B. müssen Nachrichten zwischen Objekten auf einer Maschine üblicherweise nicht verschlüsselt werden).
- Sicherheitstechnologie-Domäne: der Bereich, in dem gleiche Mechanismen zur Durchsetzung einer Sicherheitspolitik eingesetzt werden.

Zwischen verschiedenen Sicherheitstechnologie-Domänen können Sicherheitstechnologie-Gateways eingesetzt werden, um die Interoperabilität zwischen den Systemen und den dort angesiedelten Objekten zu ermöglichen.

3.1.6 Datenschutz und mehrseitige Sicherheit

Fragen des Datenschutzes werden in der CORBA Sicherheitsarchitektur nicht angesprochen. Daten werden völlig neutral betrachtet und der Besitzer der Daten kann diese nach Gutdünken verwenden. Dieser Ansatz ist aufgrund der Anwendungsunabhängigkeit von CORBA gerechtfertigt und sinnvoll. Die Definition einer entsprechenden Politik zum Datenschutz kann nur auf der Anwendungsebene vorgenommen werden, wo den verschiedenen Principals Rollen zugeordnet sind.

Die benötigten Bausteine für den Entwurf eines Systems, das mehrseitige Sicherheit realisiert, sind vorhanden, wie etwa gegenseitige Authentikation oder Nichtabstreitbarkeitsdienste. Diese zielen darauf ab, ein Gleichgewicht zwischen verschiedenen Teilnehmern herzustellen, so daß keiner die Möglichkeit hat, den anderen zu übervorteilen. Nichtsdestotrotz bietet die CORBA Sicherheitsarchitektur weder eine konkrete Unterstützung noch einen inhärenten Zwang für einen solchen Entwurf.

3.2 TINA

Wie bereits einleitend gesagt, ist für TINA noch keine umfangreiche Sicherheitsarchitektur definiert. Es sind bislang nur einige grundsätzliche Prinzipien festgelegt, wie Sicherheit für Dienste in TINA und deren Management realisiert werden soll. Interessant ist hier jedoch die Tatsache, daß bereits in diesen Grundsätzen mehrseitige Sicherheit, also sowohl für Telekommunikationsnutzer, für Telekommunikationsbetreiber als auch für unabhängige Telekommunikationsdiensteanbieter, als ein explizites Ziel festgelegt ist. Des weiteren wird Telekommunikationsnutzern ausdrücklich das Recht auf Schutz der Privatsphäre und auf Anonymität[2] zugesprochen, wobei jedoch nicht präzisiert wird, wie dies im einzelnen umgesetzt werden soll. Darüberhinaus stellt sich die Frage, inwieweit rechtliche Regelungen überhaupt Mechanismen für anonyme Nutzer erlauben werden.

In bezug auf klassische Sicherheit werden von TINA die gleichen Schutzziele identifiziert wie in CORBA: Vertraulichkeit, Integrität und Verfügbarkeit. Für die Datenkommunikation wird außerdem noch die Nichtabstreitbarkeit (die in CORBA als optional definiert wurde) sowie Datenursprungsauthentikation (data origin authentication) gefordert.

3.2.1 Sicherheitsklassifikation

Da unterschiedliche Teile eines TINA-konformen Systems unterschiedliche Sicherheitsanforderungen haben, wird in TINA die folgende Sicherheitsklassifikation eingeführt:

- Sicherheit des Telekommunikationssystems und der Telekommunikationsdienste: hier geht es um Mechanismen, die das Gesamtsystem gegen Gebührenbetrug (toll fraud) oder ein Telefongespräch gegen unbefugtes Abhören schützen sollen.
- Sicherheit des Managementsystems: das Managementsystem ist für die Administration und Wartung des Telekommunikationssystems und der dort realisierten Dienste verantwortlich; hierzu bedarf es z.B. des Zugriffs auf Konfigurationsdaten, die normalen Nutzern nicht zugänglich sein sollten.

[2] Es wird hingegen nicht detailliert, ob nur die Anonymität der Nutzer voreinander oder auch die Anonymität der Nutzer gegenüber dem Telekommunikationsdienstbetreiber gemeint ist.

- Sicherheit des DPE und der DPE Sicherheitsdienste: hier werden die Basismechanismen bereitgestellt, um die obigen Sicherheitskategorien zu realisieren.
- Sicherheitsmanagement: hier geht es um das Management der angeführten Sicherheitskategorien (wie etwa Schlüsselverwaltung).

Aus der Sicht des DPE ist die Sicherung des Telekommunikationssystems und des Managementsystems prinzipiell sehr ähnlich, da es sich bei beiden um Anwendungsobjekte oberhalb des DPE handelt. Das Managementsystem sollte vermutlich mit leistungsfähigeren (und damit kostenintensiveren) Algorithmen gesichert werden.

3.2.2 Sicherheit des DPE

Wie wir bereits in Abschnitt 2.3 festgestellt haben, ist CORBA ein sehr aussichtsreicher Kandidat für das DPE in TINA. Es kann somit davon ausgegangen werden, daß das DPE in TINA eine Sicherheitsarchitektur sehr ähnlich zu der in CORBA beschriebenen realisieren wird. Die Nutzung von CORBA im Rahmen von TINA ist jedoch dahingehend interessant, weil hier nun viele der Probleme, die in CORBA für die Erhaltung der Flexibilität ausgeklammert wurden, explizit gelöst werden müssen.

Insbesondere muß die Interoperabilität von verschiedenen DPE-Implementierungen sichergestellt werden. Dies ist erforderlich, weil unterschiedliche Betreiber von TINA-konformen Systemen nicht notwendigerweise die gleiche Technologie verwenden müssen. Die Interoperabilität ist also nötig, um Sicherheitsdienste (oder zumindest eine ausreichende Untermenge davon) auch über Betreibergrenzen hinweg sicherzustellen. Die Problematik der Interoperabilität von unterschiedlichen Implementierungen ist auch für CORBA noch nicht eingehend untersucht worden.

3.2.3 Sicherheit für Anwendungsobjekte

Aufgrund der Abstützung auf Sicherheitsmechanismen im DPE sind folglich die erforderlichen Schutzziele für Objektinteraktionen auf Anwendungsebene mit den gleichen Mechanismen gewährleistet wie in CORBA. Der nächste Schritt ist nun, diese Sicherheitsdienste auch den Nutzern zugänglich zu machen, so daß zum Beispiel der Inhalt von Telefongesprächen vor unbefugter Aufdeckung geschützt werden kann (kryptographisch geschützte Kommunikation). Ein solcher Dienst ist in TINA noch nicht definiert, aber mit den vorhandenen Mitteln relativ einfach zu realisieren.

Da wir auf die TINA-Architektur nicht detailliert eingegangen sind, können wir nur eine sehr abstrakte Version dieses Dienstes vorstellen. Nutzer sind in der Verwaltungsdomäne des Telekommunikationsbetreibers durch *user agents* (`UA`) repräsentiert. Diesem `UA` gegenüber authentisiert sich der Nutzer, wenn er einen Dienst in Anspruch nehmen will. Den Zugang zum System erhält der Nutzer über eine sogenannte *user application* (`UAP`). Diese ist im Endgerät des Nutzers angesiedelt. Will ein Nutzer eine inhaltsdatengeschützte Kommunikation mit einem anderen Nutzer aufbauen, so nimmt er mit dessen `UAP` Kontakt auf, was in den Grundfunktionen von TINA enthalten ist. Die Nutzer können sich dann gegenseitig authentisieren, wobei dieselben Mechanismen verwendet werden können, die auch zur Authentikation gegenüber dem `UA` genutzt wurden. Das Resultat einer solchen Authentikation ist i.A. ein gemeinsames Geheimnis, welches im folgenden als Sitzungsschlüssel für die Konzelation genutzt werden kann. Der Verbindungsaufbau der benötigten Kommunikationsleitung zwischen den beiden Nutzern erfolgt wie für eine normale Kommunikation in TINA. Letztendlich muß nun die `UAP` in Echtzeit die Ende-zu-Ende-Verschlüsselung der Kommunikationsdaten vornehmen, bevor diese über die Kommunikationsleitung geschickt und von der `UAP` des Kommunikationspartners wieder entschlüsselt werden.

Es ist durchaus vorstellbar, daß ein solcher Dienst von unabhängigen Diensteanbietern in Kooperation mit den Geräteherstellern definiert werden kann. Damit dies jedoch großflächig genutzt werden kann, müssen Standards definiert werden, was allerdings nicht unbedingt im Rahmen von TINA erfolgen muß. Die für eine solche Kommunikation nötige Hardware ist heute bereits verfügbar [Zim96], so daß keine grundsätzlichen technischen Probleme bei der Umsetzung des beschriebenen Dienstes zu erwarten sind.

3.2.4 Mehrseitige Sicherheit

In dem Anwendungsfeld von TINA kommt der mehrseitigen Sicherheit eine große Bedeutung zu. Hier müssen nun auch Fragen, wie etwa *ob ein Teilnehmer ohne ersichtlichen Grund eine mächtigere Position hat als ein anderer Teilnehmer*, behandelt und zufriedenstellend beantwortet werden.

Angenommen, ein Nutzer möchte die Rechnung seines Telekommunikationsbetreibers in Frage stellen. Falls der Nutzer in dieser Situation den Nachweis zu erbringen hat, daß die Rechnung nicht korrekt ist, ohne daß er dabei auf das Informationssystem des Telekommunikationsbetreibers zurückgreifen kann, so ist der Telekommunikationsbetreiber in einer sehr viel mächtigeren Position als der Nutzer (abgesehen davon, daß der Nutzer mit einem Zugriff auf das Informationssystem vermutlich nicht viel anfangen könnte). Eine Möglichkeit, diese Ungleichheit zu beseitigen, ist, die Bürde des Nachweises dem Telekommunikationsbetreiber aufzuerlegen, so daß dieser nachweisen muß, daß der Nutzer alle ihm in Rechnung gestellten Dienste tatsächlich angefordert hat[3]. Kann der Telekommunikationsbetreiber keine eindeutig vom Nutzer digital unterschriebene Anfrage für einen Dienst vorweisen, so muß er den entsprechenden Posten von der Rechnung nehmen.

4 Zusammenfassung

Im vorliegenden Artikel haben wir die grundlegenden Konzepte von CORBA und TINA vorgestellt und einen Überblick über die jeweiligen Sicherheitsarchitekturen sowie deren Beziehung zueinander gegeben.

Beide Plattformen übertragen das Paradigma des objektorientierten Aufbaus von Softwaresystemen in die Welt der verteilten Systeme. Darüberhinaus handelt es sich um offene Systeme, in denen die Kommunikationsprotokolle für die Interaktion zwischen, potentiell auf verschiedene Knoten verteilten, Objekten standardisiert und

[3] Wobei ein solcher Anforderungsnachweis nur ein erster Schritt sein kann, da hier wichtige Daten wie Nutzungsdauer oder Menge der übertragenen Daten nicht enthalten sind.

allgemein bekannt sind. Die Nutzung von offenen Plattformen für die Realisierung von verteilten Anwendungen und insbesondere für Telekommunikationssysteme wirft prinzipiell keine neuen Sicherheitsprobleme auf, sondern macht lediglich bereits existierende Sicherheitsprobleme deutlicher sichtbar.

Schwerpunkte in CORBA sind Authentikation und Zugriffskontrolle auf Objekte. Unterstützt werden weiterhin Integrität und Vertraulichkeit der zwischen den Knoten übertragenen Nachrichten, sowie die Nichtabstreitbarkeit des Sendens und Empfangs von Nachrichten. Die Sicherheitspolitik für die Interaktionen wird durch die Anwendungen und durch die Administration der Plattform bestimmt. Ein ORB erlaubt es der Anwendung, genau das gewünschte Niveau an Sicherheit zu bestimmen, sowie die dafür erforderlichen Sicherheitsmechanismen auszuwählen, was allerdings recht umfangreiche Kenntnisse über Sicherheit beim Anwendungsprogrammierer voraussetzt.

CORBA und TINA sind auch offen, was die Integration von existierenden Implementierungen von Sicherheitsdiensten angeht. In CORBA wurde versucht, möglichst viele Konzepte für die Umsetzung von Sicherheitsdiensten zuzulassen, um generisch hinsichtlich der verwendeten Sicherheitstechnologie zu bleiben. Strikt vorgegeben sind nur die Schnittstellen für Nutzung und Management. Problematisch ist, daß die Spezifikationen so offen sind, daß Implementierern der CORBA Sicherheitsdienste sehr viel Spielraum bleibt und die ersten Implementierungen abzuwarten sind. Darüberhinaus ist ohne eine sehr genaue Analyse der verwendeten Techniken und Hilfsmittel praktisch keine Aussage bezüglich der von einem CORBA-System garantierten Sicherheit möglich. Dies gilt insbesondere für Interaktionen von Objekten in verschiedenen Sicherheits- oder Sicherheitstechnologiedomänen.

Die Funktionalität des DPE in TINA deckt sich zu großen Teilen mit der von CORBA in Computernetzen. Auch die Sicherheitsanforderungen von Telekommunikationssystemen weisen auf dieser Ebene Übereinstimmungen auf. Da das DPE in TINA voraussichtlich auf CORBA basieren wird, muß folglich eine leistungsfähige Sicherheitsarchitektur auf der Grundlage des von CORBA bereitgestellten Rahmenwerks definiert werden. Dies ist nach Wissen der Autoren noch nicht geschehen, so daß kaum konkrete Aussagen über die tatsächliche Sicherheit in TINA getroffen werden können. Angesichts des weiten Spektrums zu erwartender Kommunikations- und Informationsdienste, sind in der TINA Sicherheitsarchitektur auch die Anforderungen des Datenschutzes für die Teilnehmer zu beachten. Entsprechende Arbeiten für eine präzise Definition der TINA Sicherheitsarchitektur sind anstehende Forschungsthemen, an denen wir im Rahmen eines Projektes mitarbeiten werden.

References

[DNI95] F. Dupuy, G. Nilsson, and Y. Inoue. The TINA Consortium: Toward networking telecommunications information services. *IEEE Communications Magazine*, November 1995.

[For94] W. Ford. *Computer Communications Security*. Prentice Hall, Englewood Cliffs, NJ, 1994.

[GRKK93] J. J. Garrahan, P. A. Russo, K. Kitami, and R. Kung. Intelligent network overview. *IEEE Communications Magazine*, 31(3):30–36, March 1993.

[ISO93] International Standards Organization. *Draft Recommendation X901, Basic Reference Model of Open Distributed Processing - Part1: Overview and Guide to Use.* ISO/IEC JTC1/SC 21/WG7, June 1993.

[ITU93a] International Telecommunication Union. *ITU-T Rec. M.3010: Principles for a Telecommunications Management Network.* 1993.

[ITU93b] International Telecommunication Union. *ITU-T Recommendations Q120x, Q121x, Q122x: The Intelligent Network.* 1993.

[NDC95] G. Nilsson, F. Dupuy, and M. Chapman. An overview of the telecommunicatinos information networking architecture. In *TINA'95*, pages 1–11, Melbourne, Australia, February 1995.

[OMG95a] Object Management Group. *CORBA Security.* December 1995.

[OMG95b] Object Management Group. *CORBA: The Common Object Request Broker Architecture (Revision 2.0).* July 1995.

[OMG95c] Object Management Group. *Object Management Architecture Guide.* John Wiley & Sons, Inc, 3rd edition, June 1995.

[OSF92] Open Software Foundation. *Introduction to OSF DCE.* Prentice-Hall, 1992.

[Pry91] M. de Prycker. *Asynchronous Transfer Mode: Solution for Broadband ISDN.* Ellis Horwood, 1991.

[RKF92] W. Rosenberry, D. Kenney, and G. Fisher. *Understanding DCE, OSF Distributed Computing Environment.* Addison-Wesley, September 1992.

[SW97] S. Staamann and U. G. Wilhelm. Mehrseitige Sicherheit in der Telecommunications Information Networking Architecture. In G. Müller and A. Pfitzmann, editors, *Sicherheit in der Kommunikationstechnik.* Addison-Wesley, 1997.

[Zim96] P. R. Zimmermann. *PGPfone Owner's Manual.* Pretty Good Privacy, Inc., July 1996. `http://www.pgp.com/products/fone-docs/fone_01.cgi`.