CrySTINA: Security in the Telecommunications Information Networking Architecture

L.Buttyán, S. Staamann, U.Wilhelm
Swiss Federal Institute of Technology - Lausanne
EPFL-DI, CH-1015 Lausanne, Switzerland,
fon: +41 21 693 5267, fax: +41 21 693 6770,
email: {Levente.Buttyán, Sebastian.Staamann, Uwe.Wilhelm}@epfl.ch

Abstract

TINA specifies an open architecture for telecommunication services in the broadband, multimedia, and information era. Its characteristics most relevant for security are a variety of services, a multitude of service providers, a well defined business model, a middleware platform for service development and provision, and the assumption of advanced costumer premises equipment. Concepts for its security architecture are developed in the CrySTINA project. We introduce the TINA-C architecture, analyse it with regard to security and present the CrySTINA security architecture. CrySTINA is aligned with the OMG's CORBA Security specification, but enhances it with regard to security interoperability despite the heterogeneity of security policies and technologies that must be expected in TINA networks. Thus, we present a model for the enforcement of security policies that supports the negotiation of security contexts.

Keywords

TINA, Security, CORBA, DPE, Interoperability, Security Contexts

1 INTRODUCTION

Computers are increasingly used for the control of telecommunication systems. The use of computers enables more flexibility in control as well as the fast and cheap introduction of new telecommunication services. The basic ideas for this approach stem from the Intelligent Network (Garrahan *et al.* 1993) (Magedanz and Popescu-Zeletin 1996). Network and service control functions of these networks are more and more realized as software. TINA, which stands for Telecommunications Information Networking Architecture (Dupuy *et al.* 1995), takes these developments even further. It is currently the most encompassing effort to define an open architecture for telecommunication services implemented as distributed applications in the emerging broadband, multimedia and information era. This effort is carried out by the TINA Consortium (TINA-C), a multinational consortium consisting of major network operators, as well as telecommunication equipment and computer system suppliers (Barr *et al.* 1993). The software model of TINA is based on the concept of distributed object computing.

TINA reflects several developments, which cause security problems that are new to the traditional telecommunications world. The diminishing cost of transmission bandwidth enables distributed multimedia real-time applications. The powerful Costumer Premises Equipment (CPE) available due to the progressive use of computing technology on the user's side enables a multitude of services to be delivered via a common telecommunication infrastructure to multipurpose end-user terminals. The worldwide deregulation of the telecommunication environment creates an open market for the provision of telecommunication services. Thus, telecommunication networks are not only populated by a multitude of users but also by a multitude of service providers. The cooperation and, at the same time, competition of various providers in the same physical network, as well as the expected significance of the network acting as the infrastructure for services varying from teleconferencing and video-on-demand to electronic commerce and electronic banking raise a strong demand for security and privacy of service usage and communication.

The security problem domain requires a thorough analysis of the network as a whole. In the end, a security infrastructure that belongs to the network must fulfill the security requirements of all available types of services. The implementation of this infrastructure is closely coupled to the concept of middleware, which decouples the service implementation from the underlying hardware. It is conceivable that the middleware in TINA will be based on products conforming to the Common Object Request Broker Architecture (CORBA) specified by the OMG (OMG 1995_1). Important aspects with regard to security are the self-administration of the domains (including security) and thus the probable heterogeneity of their security policies and security technologies. Both require the negotiation of security contexts to enable secure interactions between domains.

In the CrySTINA project, we develop a security architecture for TINA that is aligned to the CORBA security architecture, but additionally supports the negotiation of security contexts. In this article, we present this architecture and the rationale behind it. Section 2 introduces the TINA-C architecture focusing on the security relevant aspects. In Section 3, we analyse TINA with regard to security and propose the vertical allocation of the security functionality to the middleware layer, which is provided by CORBA. Thus, we introduce the CORBA security specifications in Section 4. In Section 5, we present the CrySTINA security architecture. Section 6 describes an implementation of its model for the enforcement of security. Section 7 summarizes the work presented and gives an outlook on our ongoing and further work.

Familiarity with TINA is not required, while knowledge about CORBA and security concepts is assumed.

2 THE TINA-C ARCHITECTURE

In TINA, services are realized as distributed applications. They consist of service components that interact with each other via a Distributed Processing Environment (DPE). The DPE is a software sub-layer that operates above the Native Computing and Communications Environment (NCCE), which is an abstraction of the computing hardware and the operating system of the service nodes. While the NCCE is technology dependent, the DPE offers a uniform interface to the distributed environment. The DPE will consist of CORBA implementations as the DPE kernel and additional TINA specific services.

Components in the application layer are divided into three categories; service components, resource components, and elements. Service components address the service logic, service access, and service management. Services can make use of common resources by interacting with resource components. The resource components are high-level abstractions of available resources, which enable the usage and the management of these resources in a technology independent way. Elements are software representations of individual networking and computing resources, such as transmission equipment, switches, or computers. Figure 1 shows the layering into applications, DPE, and NCCE, as well as the structuring of the application layer into component categories.

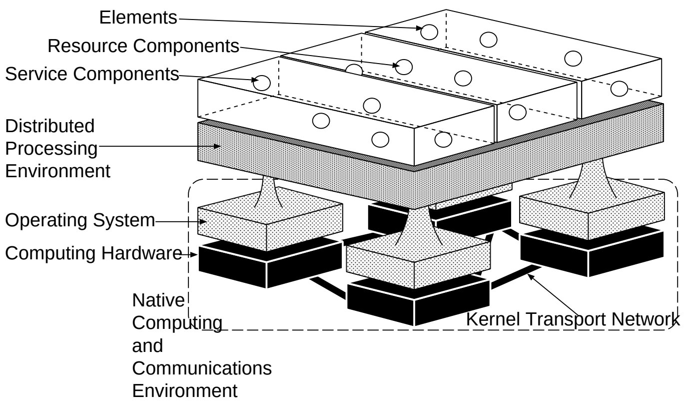

Figure 1 TINA overall architecture

The concepts for how service components are specified and how they interact are provided by the Computing Architecture. It defines the DPE as the computer and communication platform support and provides the computational modelling concepts, such as the Object Description Language (ODL), which is a superset of CORBA's IDL. Service components consist of computational objects (COs) or CO groups. COs can have two kinds of interfaces: operational interfaces, which are comparable to object interfaces in CORBA, and stream interfaces, which are designed to convey an arbitrary sequence of bytes between two components (e.g., audio or video bit streams). Messages from and to the operational interfaces are exchanged via the Kernel Transport Network (KTN), whereas streams are transferred via the Transport Network.

TINA has a general business model with various roles: the supermarket. The following roles for stakeholders are identified: Consumer, Retailer, Third Party Service Provider, Connectivity Provider, and Broker. Consumers buy services from Retailers, but the actual services are provided by third Party Service Providers. Connectivity Providers offer the necessary connectivity (streams) for the transport of content information between stakeholders. The Broker acts as a kind of yellow page and white page service (i.e., it delivers references for services that can be described by service characteristics but also by names of providers).

Each stakeholder has its own administrative domain and can act in one or more roles. Any arbitrarily complex relationship for service use is composed of simple two-party user-provider relationships. A user-provider relationship contains two types of interaction: access and usage. The access part is concerned with the establishment of a trusted and reliable temporary relationship between the user domain and the provider domain that is a prerequisite for usage interactions. Interoperability between domains is guaranteed by the definition of interdomain reference points. Figure 2 shows the business model with the defined roles and reference points.

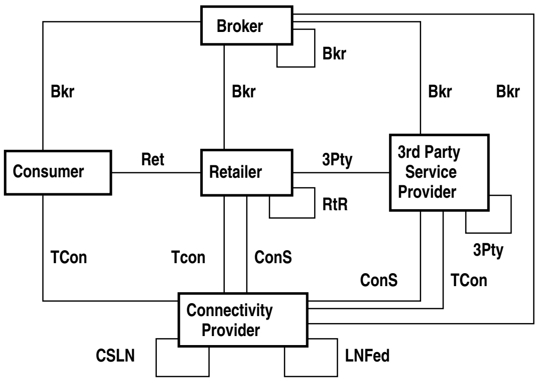

Figure 2 TINA business model

The traditional call concept of telecommunications is substituted by the more flexible session concept. At the service control level, there are access sessions and service sessions (the latter concerned with the service usage). For contents delivery, a communication session controlled by the service session is responsible for establishing and maintaining the necessary stream connections. The service session represents the instances of service usages and the state of the service logic. Exactly one provider is involved in a service session, but one or more users participate in it. Before being able to participate in a service session, each user must establish an access session with the provider. This is comparable with a login session on a multiuser computer.

Sessions and other information objects are mapped onto service components. Service control is achieved by the interaction of service components in the administrative domains of the users and the provider. Figure 3 shows the service components in their administrative domains and their relation to different sessions using the example of a video conference with two participants. For a full description of all service components, we refer to (TINA 1997) or (Staamann and Wilhelm 1997).

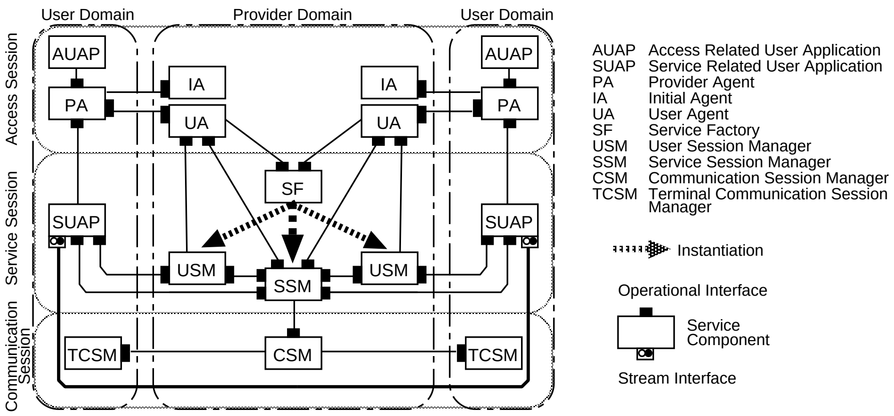

Figure 3 TINA service architecture

Most relevant for security are the service components for the access session, namely the Provider Agent (PA) in the user domain, as well as the Initial Agent (IA) and the User Agent (UA) in the provider domain, since authentication takes place between these components. The UA represents the user in the provider domain. It is the user's contact point to start or resume a service session. In order to interact with its UA (i.e., to have an access session), the user contacts the IA using the PA. The IA is the initial contact point of the provider for all users. Authentication between the user and the provider is then performed between the PA and the IA as part of the establishment of the access session. After the authentication, a reference to the UA is delivered to the PA. Additionally, a security association for further interactions between service session related service components should have been established.

3 SECURITY ANALYSIS

Security concerns all parts of a TINA system; it is pervasive and cannot be addressed in isolation. To cope with this complexity, it is necessary to structure the security problem domain in an appropriate way. All services and resources may be the subject to attacks. Attacks may be the illegitimate use of components or the modification of data, state, or programs. They may occur through direct access to systems, data, or services from outside or through modification of messages exchanged between interacting components. Potential attackers are outsiders, but also other stakeholders in the TINA network. Motives of attackers may be the illegitimate use of services, fraud (in online businesses, as well as with regard to the charging of service use), eavesdropping on and observation of consumers or providers, or the deliberate prevention of service provision (denial of service attack). The ultimate goal of an attack may be achieved directly or indirectly. In the latter case, an attacker may install a backdoor during a first successful attack, which enables him later on (and possibly at multiple times) the actually intended misuse. We structure the security problem domain according to two criteria: the architectural levels defined in the overall architecture, and the type of information (the network used for the transport), i.e., control messages (KTN) or communication contents (Transport Network). We identified the following subdomains of the security problem domain:

- **System Security** shall ensure that systems, mainly the hardware and the operating system, are not subject to intrusions or modifications. This concerns networking resources (e.g., network switches) and computing resources. It also includes the NCCE (operating system and communication ports), since intrusions may not only occur over communication ports of the NCCE that are used by the DPE, but also over other ports of the NCCE. The latter point concerns mainly the administrative domains of end users (consumers) whose CPE (e.g., PCs or workstations) cannot be assumed to be exclusively used as the endpoint of the TINA network.
- **Service Security** is mainly concerned with the preservation of the integrity of service control. Service control includes, among others, the verification of whether a user is allowed to use a service (subscription) and the accounting for billing purposes. Both rely on the authenticated identity of the user. This must be supported by a protocol for authentication of the user. The integrity of service control includes integrity of subscription verification and accounting. Access to the service functionality is controlled at two levels, the DPE level and the service level. At the DPE level, a coarse-grained access control based on the authenticated identities of the

users involved in a session prevents attempts by others to invoke operations of the service components involved in the session. At the service level, the service logic implemented in the service component controls the access to service specific information and functionality based on the authenticated identities, context, and state information (authorization). Integrity and confidentiality of the messages exchanged between the service components via operational interfaces must be achieved by the activation of the appropriate features of the DPE security services. These features must provide not only the protection of the integrity of the messages and their order but also protection against interruption of the control connection itself through interception of all messages up from a certain moment (Staamann and Wilhelm 1997).

- **DPE Security** is mainly concerned with the prevention of illegal access to service components as well as the protection of transmitted messages containing arguments, results, and exceptions of object invocations and notifications. DPE node security also provides the means to audit and report security relevant events on the node according to the audit specifications defined by the administrator. DPE security includes the security of the DPE implementation and its basic services. Since our architectural placement of security functionality allocates the general security services and mechanisms to the DPE (see Section5), also the security of the security services themselves is part of DPE security.
- **Communication Contents Security** is concerned with the authenticity, integrity, and confidentiality of the service contents information. Since all service content information in TINA is delivered in the form of streams, it deals only with streams. Streams are protected using cryptographic mechanisms, preferably stream ciphers (Rueppel 1986) (Schneier 1996) or special ciphers for certain information formats (e.g., voice or video data). If the service implemented in the provider's domain does not require any modification of the stream between two users, they can have end-to-end security. Otherwise, only user-provider security can be provided. The management of the necessary keys is part of the service control.

The most important criterion for the horizontal allocation of security functionality is: who administers a domain and the security functionality installed in the domain and has the physical control over both. In TINA, each stakeholder in the network has its own administrative domain. We make the assumption that the administrative domain is the trust domain of the stakeholder. This assumption is based on the fact that in the regular case the installed hardware is under the physical control of the stakeholder and the software is installed by the stakeholder. Thus, we assume complete trust between the stakeholder and its current administrative domain. Security within the administrative domain (intradomain security) is domain specific and is achieved by local means (e.g., operating system security measures). Within its own domain, the stakeholder trusts in the correctness of the installed software. Towards the outside, the administrative domain must be protected against illegitimate access. For interactions with other domains (interdomain interactions), limited trust relationships must be established. The communication channels between domains cannot be assumed to be secure. Protection must be achieved by cryptographic means. In order to be generally applicable but also controllable by the applications, the security functionality has to be independent from the applications above and the supplier specific NCCE below. Thus, the natural vertical allocation is the DPE. That means, it has to be aligned with the DPE implementation (i.e., CORBA). Horizontally, we allocate the necessary security functionality and the responsibility for its administration to each domain and the corresponding stakeholder. This decision must be supported by an appropriate scheme of security identities. The security identities have to be the identities of the stakeholders (their domains), which are specified in the TINA naming framework. All objects in a domain must act to the outside under the identity of their domain. From the security point of view, the objects in one domain are seen by the outside as one entity. This assumption is reasonable, since we have only one user (the stakeholder) in each domain.

Since there is no central security administration, it must be assumed that each stakeholder has its own security policy. This heterogeneity includes also different preferences for cryptographic algorithms and protocols to achieve the same security goal. Thus, for each interaction between domains, a security context has to be negotiated between the stakeholders. Such a negotiation must be supported by the security architecture and the middleware (i.e., CORBA). Since CORBA provides some security functionality, which we strive to use whenever possible in our security architecture for TINA, we introduce and analyse CORBA security in the next section and later on return to our security architecture.

4 CORBA SECURITY

The CORBA Security specification (OMG 1995_2) has been released by the OMG to provide the model and the architecture for security in CORBA systems. Besides the general model it specifies the security facilities and interfaces available to application developers, security administrators, and implementors of secure CORBA systems. It touches on the problem of secure interoperability between different CORBA implementations, although this issue is discussed in more detail by a companion document, the Common Secure Interoperability specification (OMG 1996).

The CORBA Security specification defines security to be a compound notion that is concerned with confidentiality and integrity of information, accountability of the users for their actions, and availability of the system. The latter is not covered in the document later on.

Security is enforced using security functionality built into the system. The CORBA Security specification defines the following security functionality:

- **Identification** and **authentication** of principals to verify they are who they claim to be.
- **Authorization** and **access control** to decide whether a principal can access an object.
- **Security auditing** to make principals accountable for their security related actions. Auditing mechanisms are coupled with authentication functions in order to be able to identify the principal correctly.
- **Secure communication** between objects, which requires the **establishment of security associations** between clients and targets objects, and **integrity and/or confidentiality protection** of messages in transit between them.
- **Non-repudiation** to provide evidence of actions such as proof of origin or receipt of data.
- **Administration** of security information.

Most of these security functions are performed during a secure object invocation, which is the basic notion of the specification (Figure 4). Each secure object invocation requires an established security association between the client and the target object. The security association is established by authentication between the client and the target, making the client's security attributes (identity and privileges) available at the target side, and establishing the security context that will be used when protecting messages in transit between the client and the target object. The way of establishing a security association (e.g., whether simple client to target authentication or mutual authentication is used) depends on the security policies that govern both the client and the target object. Associations will normally persist for many interactions.

For each object invocation, the request from the client to the target object is subject to access control. Access control may take place at the client side, the target side, or both sides according to the access control policy. The access decision (i.e., whether this client can perform this operation on this target object) is based on the client's security attributes, the target's control attributes (e.g., access control list), and other relevant information about the action (e.g., the operation and data) and about the context (e.g., the current time). This general model enables a large variety of access control schemes, ranging from access control lists, over capabilities, to label based schemes. The scale of access control is not specified, but it can be assumed that implementors will provide access control down to the granularity of operations.

In many cases, objects perform operations on behalf of the initiator of a chain of object invocations. In such cases, the initiator, which can be a human user or a system entity, needs to delegate some or all of its privilege attributes

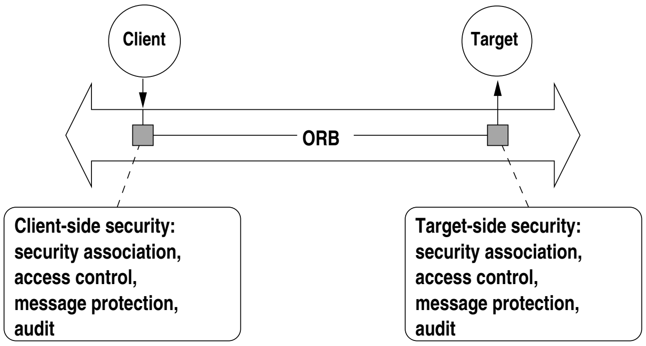

Figure 4 Secure Object Invocation in CORBA

to the intermediate objects that will act on its behalf. The CORBA Security specification is very general and enables virtually all kinds of delegation models (simple, composite, combined and traced delegation). The actual type of delegation is selected according to the delegation policy either by the ORB system automatically, or by applications via well defined interfaces.

Depending on the security policy, the integrity and/or confidentiality of the messages between the client and the target object may be protected, and optionally non-repudiation may be provided by cryptographic means. For the detection of actual or attempted security violations, security auditing is performed. Depending on the implementation, recording security relevant events may involve writing event information to a log, and/or generating an alarm. Audit policies specify which events should be audited under which circumstances.

A distributed object system may consist of a huge amount of objects with a possibly even larger amount of security associations between them. This fact raises the issue of scalability. In order to cope with the scalability problem, the notion of domains is introduced. Three types of domains with regard to security are defined by the CORBA Security specification: security policy domains, security environment domains, and security technology domains. A security policy domain is the scope over which a security policy is enforced. A security policy domain is administered by a single security authority. A security environment domain is a domain in which the enforcement of the security policy is achieved by local means (e.g., objects on the same machine). The security technology domain is a set of objects for which the same security technology (e.g., Kerberos (Neuman and Ts'o 1994)) is used to provide security.

Each security domain contains a domain manager object that references the valid security policy objects for that domain. A security policy object is the representation of a security policy (e.g., access control, delegation, secure invocation, or audit policy), which is defined and managed by the security administrator of the domain. Upon the creation of an object, the ORB implicitly associates the object with one or more security domains according to the construction policy (the objects can be moved between domains later on) and then transparently enforces the security policies of those domains. In this way, security is provided to all applications, even to those that are unaware of it. Additional security measures may be enforced by the applications themselves. This may be done by additional enforcement of administrator defined policies, and/or direct use of security features (e.g., non-repudiation) via application interfaces. The security measures enforced by the applications cannot override the security policies defined by the administrator. The rationale behind these two levels of security is the fact that in the general case, application developers cannot be expected to be aware of all the threats to which the system will be subject, and to put the right countermeasures in place. On the other hand, there are mission critical applications (e.g., in the banking or the telecommunications world) that require the application programmer to have more control over security.

The security functionality described above (i.e., authentication, access control, message protection, auditing, etc.) is provided by ORB security services that may rely on some underlying security technology, which itself may use operating system mechanisms and additional security hardware (Figure 5). During secure object invocations, the ORB intercepts the requests and replies between the client and the target object and calls the appropriate security services. Some of the security services can be invoked by the applications directly, to enforce their own security preferences. An important aspect of this architecture is that the components that implement the security services are independent of any specific security technology. The specification allows the use of an isolating interface (e.g., GSS-API (IETF 1993)) between this level and the security technology, allowing different security technologies to be accommodated within the architecture, such as technologies based on operating system protection mechanisms, existing security components (e.g., cryptographic libraries), or a set of distributed security services.

The specification of secure interoperability between different CORBA implementations extends the CORBA 2.0 standard. A new protocol, the Secure Inter-ORB Protocol (SECIOP) is specified, which enables secure interactions of clients and target objects that reside on different ORBs, as long as the same security technology is used on both sides. The information, which security technology an object supports is part of the interoperable object reference (IOR). Based on the information in the IOR, a security context acceptable for both sides can be established. The establishment of the security association and the protection of messages are controlled by security tokens that are added to the inter-ORB protocols. Key management is not explicitly dealt with in the CORBA Security specification.

The Common Secure Interoperability document (OMG 1996) specifies the use of the protocols of three security technologies within SECIOP, namely

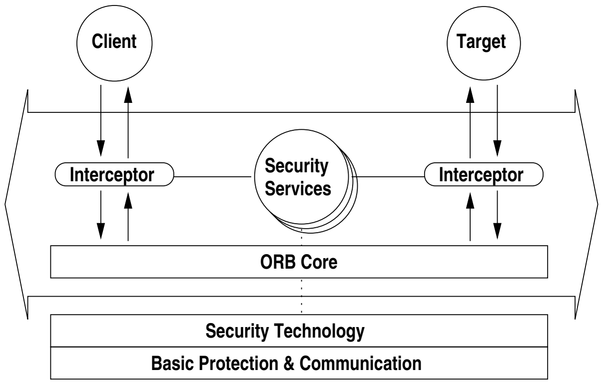

Figure 5 Model of Security in CORBA

SPKM, Kerberos, and the ECMA security protocol. If the interoperability between the ORBs is based on DCE (OSF 1992), then the DCE security technology (X/Open 1994) based on the Kerberos protocols can also be used. A recent addendum (OMG 1997) allows also the Secure Socket Layer (SSL) (Netscape 1996) to be the basis of inter-ORB security.

5 THE CRYSTINA SECURITY ARCHITECTURE

Security features in TINA are implemented at various levels. In our approach, the DPE offers general security functionality to the applications on each DPE node as part of the DPE functionality. To ease integration in applications, as much functionality as possible should be provided as self-contained DPE security services. However, the DPE should also provide lower level DPE security mechanisms to the applications for handling application specific security tasks. The layering of the security functionality is illustrated in Figure 6. The DPE security services are exclusively based on the DPE security mechanisms. The implementation of these mechanisms may directly use cryptographic mechanisms or may be built on available higher level security technology, such as Kerberos. The underlying security technology may use the same cryptographic mechanisms as the DPE security mechanisms or proprietary implementations. The use of cryptographic mechanisms and/or higher level security technology may be accomplished through standardized interfaces (e.g., GSS-API) to facilitate the integration of existing products into the DPE. Above the DPE level, there are special security services, which also rely exclusively on the DPE security services and mechanisms. They are used by TINA applications, but are applications themselves. The special security services are not implemented on each DPE node. Examples are electronic cash support or notary services.

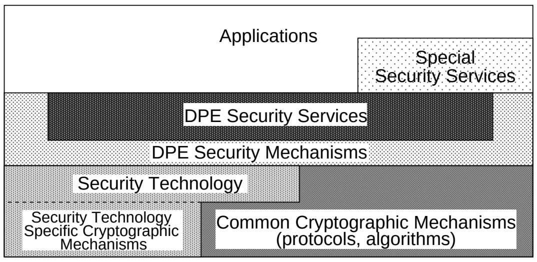

Figure 6 Layering of CrySTINA security features

Since the TINA DPE is provided by CORBA products, a natural starting point for the TINA security architecture is the CORBA Security specification. The generality of the CORBA Security specification makes it suitable to be the basis of security for a broad spectrum of business applications. In some respects, it is even more general than required, and it is questionable whether this rich functionality is necessary for certain families of applications, such as telecommunication services based on the TINA architecture. Let us consider, for instance, the access control schemes and delegation models described in the CORBA Security specification. TINA service components are implemented as TINA COs or CO groups. TINA COs may have multiple interfaces, as opposed to CORBA objects, which have exactly one interface. We assume that each interface of a TINA CO will be implemented by a dedicated CORBA object, and thus each TINA CO will be realized as a set of CORBA objects (Kitson 1995). Since the functionality offered by a TINA CO is structured into interfaces according to the coherence of subfunctionalities, access control to a TINA CO can be applied at the granularity of TINA CO interfaces, which means that we only need to control access to whole CORBA objects. Furthermore, TINA service components always act on behalf of the stakeholder that owns them, therefore it is sufficient to support identity based access control schemes at the target side and no delegation is required.

For other aspects, secure interoperability provided by CORBA security may not be sufficient for the TINA architecture. Secure interoperation between objects depends on the membership of the objects to security policy domains, security technology domains and ORB technology domains. We assume that each TINA administrative domain is mapped onto one security policy domain and one ORB technology domain, and that each boundary between TINA administrative domains is also a boundary between security technology domains (Staamann *et al.* 1997). The latter reflects that stakeholders with various kinds of customer premises equipment, varying priorities regarding security, and under possibly different national laws cannot be assumed to have the same security technologies. Interoperability between objects in different security policy domains can, thus, only be achieved if both domains agree on a common security policy for the respective interactions. This common security policy can be negotiated at invocation time or in advance. Objects in different ORB technology domains can interact securely using the SECIOP, as long as the same security technology is used at both sides. According to the CORBA Security specification, interaction of objects in different security technology domains (e.g., objects belonging to different stakeholders in different countries) requires a security technology gateway. However, this may cause a trust problem, because such a gateway cannot be realized without the administrators of both security domains trusting each other or a third party that runs the gateway. A less restrictive solution that is not supported by CORBA would be to negotiate the security technology, as well.

Below in Figure 7, we present our model of security for distributed object based telecommunication architectures, such as TINA, that is based on the CORBA Security specification with some modifications according to the observations above.

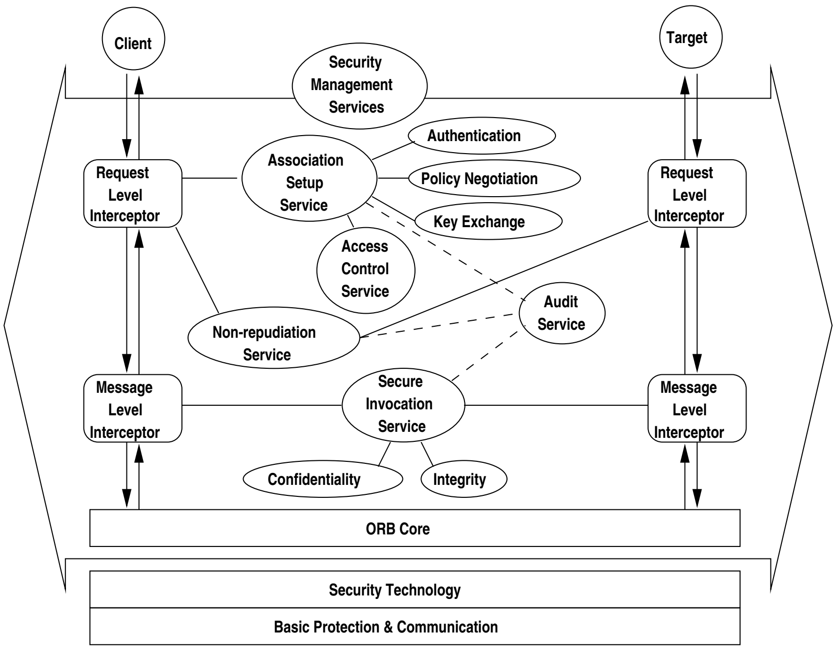

Figure 7 CrySTINA Model of Security

When a client invokes an operation on a target object, a request and in most cases a reply are passed between them. According to the CORBA Security specification and based on the observation of various CORBA implementations, we assume that the request and the reply can be intercepted at two levels: at the request level, where we have access to the request and the reply as structured data, and at the message level, where the request and the reply are available as an unstructured buffer containing the respective messages in a serialized form. These two levels of interceptions are very well adapted to support the enforcement of security, since some of the security services (e.g., access control) can best be performed on structured requests where the information about the involved principals and the operation is directly available, while other security functions (e.g., encryption) can more naturally be applied to unstructured raw data.

Each request is intercepted by the request level interceptor at the client side. If there is no security association established between this client and target object, then the Security Association Setup Service is called and a security association is established between them. This means mutual authentication, security policy negotiation, and exchange of security related parameters (e.g., cryptographic keys, initialization vectors, etc.). The Security Association Service uses the identity information of the stakeholder that owns the client in the authentication process. Based on the authenticated identity of the client, access control on the target object can be performed in this phase. If the access is not allowed, then the association is not established at all, and the client is notified (e.g, an exception is raised). An established security association is represented by security context information on both sides. Then the request is processed further according to the negotiated security policy (e.g., the Non-repudiation Service is called, if it is mandated).

On its way to the network, the request is intercepted by the message level interceptor as well, which calls the Secure Invocation Service. According to the negotiated security policy, integrity and/or confidentiality protection is applied using the security context information established before. The request is then passed to the ORB Core, which transfers it to the target side. At the target side, the applied services are called in reverse order (with the exception of the Security Association Setup Service, since the association has already been established).

Each service can call the Audit Service, if an event occurs that should be audited (e.g., an integrity violation has been detected). The placement of Security Management Services at the edge of the ORB system indicates that these services are usually called by management applications on behalf of the security administrator.

6 IMPLEMENTATION OF THE MODEL

In the following, we discuss an object-oriented implementation of the model described above. Figure 8 shows this implementation. The Security Association Setup Service, the Audit Service, and the Security Management Services are realized by a Domain Security Manager object as an independent CORBA service. The Domain Security Manager object provides interfaces

- to applications, through which they can request the setup of new security associations, get established context information, and set security preferences that are not conflicting with the domain security policy,
- to other Domain Security Manager objects, through which they can authenticate each other, negotiate policies and exchange keys, and
- to management applications, through which they can manage credential and policy objects.

The Domain Security Manager object is unique in each domain and available to all applications in the domain. The established security association between a client and a target object is represented by the Security Context objects at the client and the target side that are local to the client and the target. These Context objects contain all the information of the association and provide the Secure Invocation and Non-repudiation Services. Access control is performed explicitly at association setup time, therefore it does not have to be performed by the Context objects. Access control is implicitly provided for each invocation within a security association by the identification of the client. In order to identify the right local Context object that should be applied to handle the current invocation, we require a Local Context Manager object, which manages the various local Contexts. The Local Context Manager requests the association establishment and downloads the context information from the Domain Security Manager. The Context and the Local Context Manager objects are not CORBA objects, they do not have visible interfaces.

Let us now look again and in more detail at how this implementation realizes the CrySTINA model by tracing a secure object invocation. The client's request is intercepted by the request level interceptor at the client side. This interceptor invokes the Local Context Manager to obtain the context that should be applied to the request. If there is a context already established between this client and target, then the Local Context Manager returns a reference to it. If there is no such context, then the Local Context Manager calls the Domain Security Manager to establish one. The Domain Security Manager object uses the identity information of the stakeholder that ownes the client in the context establishment process. This information is stored in the Credentials object. Once the context is established, the Local Context Manager downloads it, and returns a reference for it to the request level interceptors. Explicit access control is performed by the Domain Security Manager in the association establishment phase. Access is allowed or denied at object granularity level. If access is allowed, then the association is established between the client and the target, and the Local Context Manager returns a reference to the appropriate Context object; otherwise no association is established, and the Local Context Manager raises an exception. When the request level interceptor obtains the reference to the Context object, it calls it with the request as the parameter. The Context object will perform the required services on the request (e.g., non-repudiation of origin) according to the security policy.

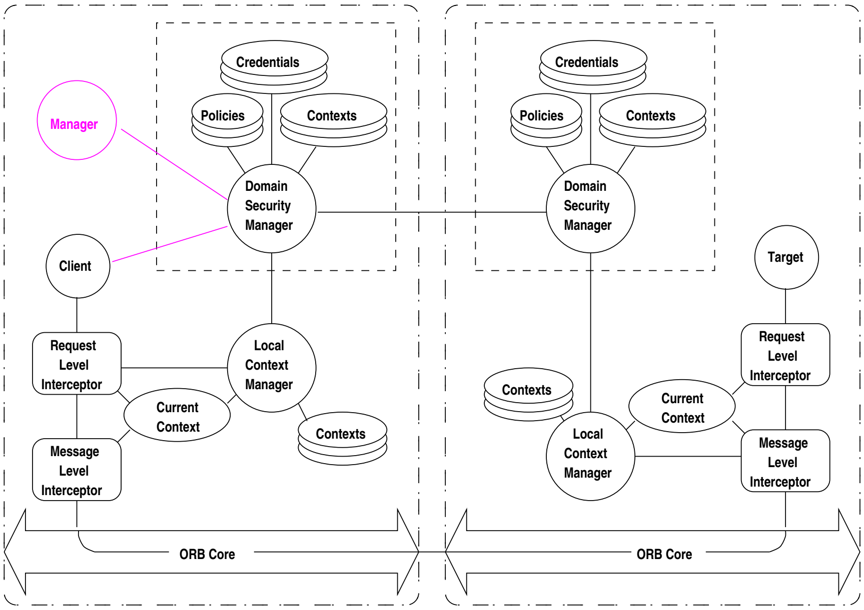

Figure 8 Implementation of the CrySTINA Model

Then, the request is passed on and it is intercepted by the message level interceptor. This interceptor already knows which Context to apply, and it can call this Context object with the message as the parameter. Note, that at this level we have access to the message as an unstructured stream of bytes, so the Context object can easily perform the Secure Invocation Services, according to the security policy. The last step is to put a special security header at the beginning of the message that contains the necessary information for the target side message level interceptor to identify which context it should use to reclaim the message. Then the protected message is passed to the ORB Core that sends it to the target.

At the target side, the incoming message is intercepted by the message level interceptor. This interceptor interprets the security header, and calls the Local Context Manager with the parameters found in the security header (e.g., a context ID) to obtain a reference to the Context object that should be used for this message. If the context is already available (i.e., this is not the first message from the given client), then a reference to it is returned by the Local Context Manager, otherwise the Local Context Manager first downloads the context from the Domain Security Manager, and then passes the reference to the interceptor. Note, that the context is already available at the Domain Security Manager, because the association is already established between the client and the target by the time when the message arrives at the target. When the message level interceptor receives the reference to the appropriate

Context object, it calls it with the message as the parameter. The Context object reclaims the original clear message.

In the next step, the request is intercepted by the request level interceptor. The interceptor already knows which Context to apply for this request, and it can call this Context object with the request as the parameter. The Context object performs the appropriate operations (e.g., non-repudiation of origin) on the request. The fact that the request arrived at the target side request level interceptor already means that access is allowed. Finally the request is passed to the target object.

The reply is handled in a similar way. The difference is that the interceptors already know which context to apply because they have temporarily stored this information when handling the request.

7 CONCLUSION

We introduced the TINA-C architecture, an open architecture for telecommunication services ranging from teleconferencing over video-on-demand to electronic commerce. We provided an analysis of the security problem domain in this architecture. As a result of this analysis, we allocated the necessary security functionality to each administrative domain and within the domain to the middleware layer (DPE), which is basically provided by CORBA products. CrySTINA, our security architecture for TINA, was presented. Because of its allocation to the middleware layer, the implementation of our architecture is closely related to CORBA security. Unlike the CORBA security architecture, CrySTINA can cope with the heterogeneity of security policies and security technologies, which must be expected as a side effect of the self-administration of the administrative domains in TINA. This is achieved by the negotiation of security contexts. Our future work will be concerned with the generalization of CrySTINA's negotiation concept to CORBA as a general middleware platform. Ongoing work is concerned with the prototypical implementation of the concept using a commercial ORB product (Orbix 1997) and a free CORBA implementation (Brose 1997).

ACKNOWLEDGMENT

This work has been supported by the Swiss National Science Foundation as part of the Swiss Priority Programme Information and Communications Structures (SPP-ICS) under project number 5003-045364.

REFERENCES

Barr, W.J. Boyd, T. and Inoue, Y. (1993) The TINA Initiative. *IEEE Communications Magazine*, March 1993, 70-76.

Brose, G. (1997) JacORB: Implementation and Design of a Java ORB. *proc. DAIS'97, IFIP WG 6.1 International Working Conference on Distributed Aplications and Interoperable Systems*, Cottbus, Germany, September 1997, Chapman & Hall.

Dupuy, F. Nilsson, G. and Inoue, Y. (1995) The TINA Consortium: Toward Networking Telecommunications Information Services. *IEEE Communications Magazine*, November 1995, 78-83.

Garrahan, J.J. Russo, P.A. Kitami, K. and Kung, R. (1993) Intelligent Network Overview. *IEEE Communications Magazine*, March 1993, 30-36.

Internet RFC 1508 (1993) Generic Security Service - Applications Program Interface (GSS-API).

IONA Technologies (1997) Orbix 2.1.

Kitson, B. (1995) CORBA and TINA: The Architectural Relationships. *proc. TINA'95 Conference*, Melbourne, Australia, February 1995.

Magedanz, T. and Popescu-Zeletin R. (1996) *Intelligent Networks*. International Thomson Computer Press, London, 1996.

Netscape (1996) Secure Socket Layer (SSL).

Neuman, C. and Ts'o, T. (1994) Kerberos: An Authentication Service for Computer Networks. *IEEE Communications Magazine*, September 1994, 33-38.

Object Management Group (1995) The Common Object Request Broker, Architecture and Specification, Revision 2.0.

Object Management Group (1995) CORBA Security.

Object Management Group (1996) Common Secure Interoperability (CSI).

Object Management Group (1997) CORBAsecurity/SSL Interoperability.

Open Software Foundation (1992) *Introduction to OSF DCE*. Prentice Hall.

Rueppel, R. (1986) *Analysis and Design of Stream Ciphers*. Springer, 1986.

Schneier, B. (1996) *Applied Cryptography*, 2nd edition. Wiley, 1996.

Staamann, S. Buttyán, L. Hubaux, J-P. Schiper, A. and Wilhelm, U. (1997) Security in the Telecommunications Information Networking Architecture – the CrySTINA Approach. *proc. TINA'97 Conference*, Santiago, Chile, November 1997, IEEE CS Press.

Staamann, S. and Wilhelm, U. (1997) Cryptographic Protection of Connection Integrity with Interruption Detection in TINA. *proc. DAIS'97, IFIP WG 6.1 International Working Conference on Distributed Aplications and Interoperable Systems*, Cottbus, Germany, September 1997, Chapman & Hall.

TINA Consortium (1997) Service Architecture, Version 5.0.

X/Open Guide G410 (1994) Distributed Security Framework.