CLOSED USER GROUPS IN INTERNET SERVICE CENTRES*

Sebastian Staamann[1]
Levente Buttyán[1]
Allan Coignet[1]
Ernesto Ruggiano[2]
Uwe Wilhelm[1]
Marc Zweiacker[2]

[1] Swiss Federal Institute of Technology Lausanne
EPFL-DI, CH-1015 Lausanne, Switzerland

[2] Swisscom Corporate Technology
Ostermundigenstrasse 93, CH-3000 Bern 29, Switzerland

{ Sebastian.Staamann | Levente.Buttyan | Allan.Coignet | Uwe.Wilhelm } @epfl.ch
{ Ernesto.Ruggiano | Marc.Zweiacker } @swisscom.com

Abstract: The paper presents a model for end-user directed access control to services in Internet service centres that, beside the classical Internet services (e.g., e-mail), offer a multitude of new services (e.g., on-line conferencing and auctioning) over the Internet. The model is based on the concept of closed user groups. The main idea is that at creation time each service instance and its components are assigned to a user group previously formed by a subset of the end-users, and access control is performed for access attempts through checking the group assignment of the accessed resource against the group memberships of the authenticated accessing end-user. Access control is directed

---

Research partly funded by Swisscom Corporate Technology as well as partly funded by the Swiss National Science Foundation (SNSF) as part of the Swiss Priority Programme Information and Communications Structures (SPP-ICS) under project numbers 5003-045364 and 5003-054575.

by the end-users through the management of group memberships. We describe the concept of closed user groups, the management of group memberships, the enforcement of access control, and the realisation with off-the-shelf software for a middleware based service environment, which is characterised by the use of CORBA, Java, and WWW technology.

Keywords: Access control, Access rights management, Authorisation, Closed user groups, CORBA, Group management, Internet service centres, Middleware, Security

1 INTRODUCTION

With the Internet becoming a ubiquitous telecommunications infrastructure, providing contents and communications based services over the Internet to business and residential end-users is becoming an attractive business area. Many new specialised service providers have developed proprietary service architectures for the Internet, such as Amazon [15] in the area of online book selling or Pointcast [16] in the area of information dissemination. In most cases, these service architectures are based on World Wide Web (WWW) technology. However, classical telecommunication operators and Internet Service Providers (ISPs) are more interested in the development of general service architectures that enable them to develop and offer a variety of different application layer services in a homogeneous and cost-efficient way over the Internet to the end-users. Due to the state and connection oriented nature of most classical and new telecommunication services, the use of WWW technology only is, in most cases, not sufficient to realise complex services. Thus, Internet service architectures of general service providers require more powerful middleware technology. In most cases, this middleware layer consists of CORBA [12] products as the core, WWW technology as the means for the initial access to services by the end-user, and, potentially, Java applets as the glue between WWW and CORBA. The Java applets realise the end-user system part of the CORBA based distributed application, which can be downloaded from the provider's service centre and executed on nearly all types of systems at the end-user's side.

An essential requirement of any service architecture is the provision of security. As we have discussed in [7] for the TINA architecture[10], general security functionality in middleware based service architectures can most efficiently be implemented in the middleware layer. While user authentication and message protection in architectures based on CORBA or WWW technology can easily be provided by the Secure Socket Layer (SSL) [3], authorisation and access control for the services and their components are rather open questions. In this article, we present a model for end-user directed access control to services, which is based on the concept of closed user groups, and the realisation of this model in a multi-domain environment. The main features of our model are the easy integration with existing CORBA based distributed applications (services), user centric design as well as flexibility with regard to authorisation by giving control to the end-user, and intelligibility for ordinary end-users. The model was developed and implemented for Internet service centres to be run by Swisscom, the largest Swiss telecommunication network operator.

In Section 2, we describe some security relevant characteristics of the provision of telecommunication services on the Internet, and sketch the middleware based service platform, for which we have developed our security architecture. The security architecture itself is introduced in Section 3 after a short security analysis. Section 4 describes in detail the concepts of the closed user group environment, including the solutions for the access control enforcement and the end-user controlled management of the closed user groups. Section 5 reports on the realisation of the concepts. Section 6 concludes the article with a summary and an outlook on further work. Familiarity of the reader with the basics of CORBA, Java, and WWW technology is assumed, whereas deep knowledge in the area of computer and communications security is not required.

2 PROVISION OF TELECOMMUNICATION SERVICES ON THE INTERNET

Several recent publications discuss the state-of-the-art of the expected convergence of telecommunications and computing towards networking models for integrated services and applications (e.g., [1]), and the impact of the Internet on telecommunication architectures (e.g., [4]). In general, they stress the importance of Internet technology, and they acknowledge the usefulness of the middleware concept for the development and provision of telecommunication services. Many telecommunication operators (e.g., British Telecom [6] and Korea Telecom [8]) as well as some telecommunication equipment suppliers (e.g., Siemens [5]) have started to develop middleware based service architectures. In most cases, these service architectures can be seen as simplified versions of the TINA-C service architecture [11] tailored to the Internet. We have identified the following common characteristics in most of these projects:

- All services are implemented and offered by exactly one provider.
- Initial access to services is provided using WWW technology. The initial contact point for the user is the provider's home page, which is downloaded using the WWW's Hypertext Transfer Protocol (HTTP).
- In order to facilitate the consistent, fast, and cost saving development, introduction, and operation of services on a mature, standardised, scalable, and vendor-independent platform, the core of the middleware is provided by CORBA products.
- Together with the provider's web page, the user downloads Java applets, which implement the CORBA objects and user interfaces of the services to be executed on the user's side as well as (parts of) the underlying ORB functionality necessary for the interaction with the provider. The standard protocol to be used for service interactions between the distributed objects of the user and the provider as well as those within the provider domain is the Internet Inter-ORB Protocol (IIOP), which is the realisation of the General Inter-ORB Protocol (GIOP) over TCP/IP.
- Trust is unilateral as it is in the implicit trust model of traditional telecommunications. The provider is assumed to guarantee correct charging and security of

service operation. The user relies on the honesty of the provider not to misuse the user related information (including contents, usage, and charging information) and the competence of the provider to prevent any misuse of this information by other users or outsiders.

The service platform, for which we have developed the closed user group concept, is a typical instance of the class of TINA-oriented Internet-based telecommunication service architectures described above. It follows the approach of object orientation for the modelling, realisation, and delivery of telecommunication services as distributed applications, and strongly relies on a CORBA based middleware layer. The service platform in operation is partitioned between a service centre operated by the provider and the computers of the users. The distribution concerns the service components as well as some functionality of the middleware platform. The parts of the platform that are executed on the computers of the users are Java applets, which are downloaded from the service centre.

3 SECURITY ANALYSIS AND ARCHITECTURE

The users require security for the information handled by the distributed services. Following the standard structuring of the security problem domain, as presented in most of the text books (e.g., [2]), security of information and resources is concerned with confidentiality, integrity, availability, legitimate use, and accountability. Confidentiality means that information is not disclosed or revealed to unauthorised parties. Integrity means consistency of data; in particular, preventing unauthorised creation, alteration, or destruction of data, including the integrity of the information about the origin of transferred messages (authenticity). Availability ensures that legitimate users are not unduly denied access to information and resources. Legitimate use means that resources, including data, are not used by unauthorised parties or in unauthorised ways. Finally, accountability means that authorised users of the system can be held accountable for their actions in the system. How these objectives can be achieved depends on the scenario, for which security has to be enforced.

Concerning the usage scenario of our service platform, we made the following assumptions, which we consider to be typical for many middleware based telecommunication architectures with the characteristics mentioned in Section 2:

- All services are provided by a single provider, which operates its own network and the service centre. The users registered with the service centre have personal computers, which are connected to the provider's network via the public telecommunication network or the Internet.

- The provider network as a whole is the provider's security environment domain, in which correctness of the local execution of software and the non-disclosure of secret data can be assumed as a prerequisite. Each user's security environment domain consists of his personal computer including all hardware and pre-installed software. Temporarily, the software base in the user's domain is extended with securely downloaded code from a trusted source. Each party possesses, administers, and physically controls its own security environment domain.

- The users of the service platform trust the provider that it will not misuse any information that the users have handed over for the purpose of service usage. None of the users is trusted by the provider or any other user by default. However, a user who participates in a service session implicitly trusts the other members of the closed user group that has been assigned to the service session not to misuse the information exchanged in the course of the service use and not to attack the service execution in any other way.

- The communication channels between the users and the provider are unsecure, i.e., potentially subject to passive and active attacks by outsiders or other users.

The security solution for the service platform must provide the means to guarantee confidentiality and integrity of all messages exchanged. Availability in the sense of prevention of denial of service attacks (e.g., interception of all messages) cannot be expected from the service platform at this stage, since this would require the collaboration of the operators of the unsecure networks that are used to access the provider network. An additional requirement is effective and efficient access control for information and functionality. Since the middleware of the service platform is object oriented (CORBA), access control can naturally be enforced at the object interface level in order to achieve the access control for information and functionality in an integrated manner.

Guaranteeing the desired protection properties for the messages exchanged over public channels requires the use of cryptographic mechanisms. This is achieved in our environment by the use of SSL as the secure transport layer, which ensures connection integrity and confidentiality. For mutual authentication, we use a combined scheme of two different unilateral authentication schemes. For the authentication of the service centre to the end-user, we use the server authentication scheme, which is part of the SSL and which is based on the X.509 [9] compliant public-key certificate of the service centre signed by a certification authority, which is trusted by the end-user and whose public key is delivered to the end-user as part of the web browser package. Authentication of the end-user to the service centre is performed using a user name/password pair transmitted over the previously established confidentiality protected SSL connection. User name and password are securely communicated to the end-user off-line before he can connect to the service centre. The motivation for this asymmetric design is that in the current situation the service centre can be assumed to have a public key certificate, whereas, for the sake of fast acceptance of the service centre in the community of ordinary Internet users, the assumption that each user has a public-key certificate had to be avoided.

Our main concern in providing a security solution for the service centre is to prevent illegal access to objects in the service centre from outside. For objects that are purely internal, this is prevented by application layer firewall functionality that blocks all GIOP invocation request messages for operations of internal objects. For objects that are explicitly made visible to parties outside their own security environment domain through the export of their object interfaces, access must be controlled by an appropriate authorisation scheme. In order to solve this problem, we decided to adopt the concept of closed user groups, which turns the CORBA based service environment into a CLosed User group Environment (CLUE).

Until now, we have only discussed the operational phase of the CORBA based service platform. But all security implemented there can only be guaranteed if the code implementing the security measures cannot be corrupted by attackers. Correct execution can be assumed by the user, but only if the code has not been modified before. Since in the presented scenario, (parts of) the service implementation that shall be run on the user's side (including middleware functionality, e.g., security modules) are downloaded from the service centre over unsecure links, the authenticity and integrity of this mobile code must be guaranteed by additional means before the proprietary security functionality of the service platform can become active. We will not propose a solution to solve this problem, but we rely on the concept of *signed applets* provided by Java product suppliers and its integration into the most popular web browsers. Hence, in the following we assume that all the code executed as part of the service platform, including the downloaded code, is protected from any modification by any party other than the executor and the originator of the code.

4 THE CLOSED USER GROUP ENVIRONMENT

The fundamental idea of the CLUE is to give each user the ability to select or to create suitable environments for the usage of services that prevent other users outside these environments from interfering with the service sessions. This can be compared with entering a private room with another person in order to have a private conversation. Users should achieve their security objectives simply through running the desired telecommunication service in the closed user group environment. Before starting the service session, they establish a closed user group (CUG) with the intended participants of the service session as members. Then they start the service together with the specification of the CUG they want the service to be performed in. The CLUE guarantees that outsiders or other users who are not members of the respective CUG cannot interfere with the service session, which means that they cannot learn the contents of the messages exchanged between the parties in the session, neither they can have access to any service components that belongs to the session. Once established CUGs can be reused and the membership in a CUG can be dynamically changed even during a service session (e.g., in order to introduce a new user to a running video conference or to expel a current participant).

4.1 Access control to end-user services

The guarantees of the CLUE are based on the message protection provided by SSL, which ensures confidentiality, integrity, and authenticity of all messages exchanged, and the access control model described in this subsection. We start with a formal presentation of the model and then discuss the approach we have chosen to implement it.

In our access control model, the objects (which can or cannot be accessed) are service instances and the subjects (who can or cannot access the objects) are users. We denote the set of all users and the set of running service instances by $\mathcal{U}$ and $\mathcal{S}$, respectively. Users may belong to various closed user groups. A user group G is a set of users (i.e., $G \subseteq \mathcal{U}$). $\mathcal{G}$ stands for the set of all user groups. We note, that the sets defined above are not constant in time, however, in our model that underlies the realisation presented later the access decision will always depend on the current values of these sets, therefore, we omit the time and we always mean the current values.

We assume the existence of a function $g : \mathcal{S} \to \mathcal{G}$, which assigns a group $g(s)$ to each service instance s in $\mathcal{S}$. Intuitively, $g(s)$ may be thought of as the group, in which the service instance s has been started.

The access decision function $a : \mathcal{U} \times \mathcal{S} \to \{allowed, denied\}$ is, then, defined by the following formula:

$$a(u, s) = \begin{cases} allowed & \text{if } u \in g(s) \\ denied & \text{otherwise} \end{cases}$$

which means that in order for a user to access a service instance, he must be a member of the group, to which the service instance is assigned. It is easy to see that access control based on the decision function a ensures the required guarantees for the CLUE.

There are various ways to implement this model, which differ from each other in the way of how the information that is necessary to make the access decision is obtained and in the place of the access control enforcement. In our prototype implementation of the CLUE, which we will describe in detail in the next section, the access control information (group memberships and group assignments) is stored in a central database and an on-line server is required to provide it to the implementation of the access decision function upon request. An obvious disadvantage of this approach is that the on-line server may become a bottleneck if many access decisions have to be made at the same time. On the other hand, the management of the access control information is simple, because of its centralised nature. Since none of the users is trusted by the provider, access control is enforced at the server side. Regarding the architectural placement of the access control enforcement, our implementation is based on interceptors, a concept, which is part of the OMG's CORBA specification [12] and implemented in most commercial ORB products. An interceptor is a piece of code, which mediates operation requests and responses between the application and the ORB system. Interceptors can add extra data to the request at the client side and extract the added data at the server side, thereby transferring information from one side to the other transparently to the applications. Adding interceptors to an existing application is usually a routine programming task and does not require the modification of the application. The structure of the implementation is illustrated in Figure 1. The access decision function itself is implemented as an independent server process, which takes the access control information from a database.

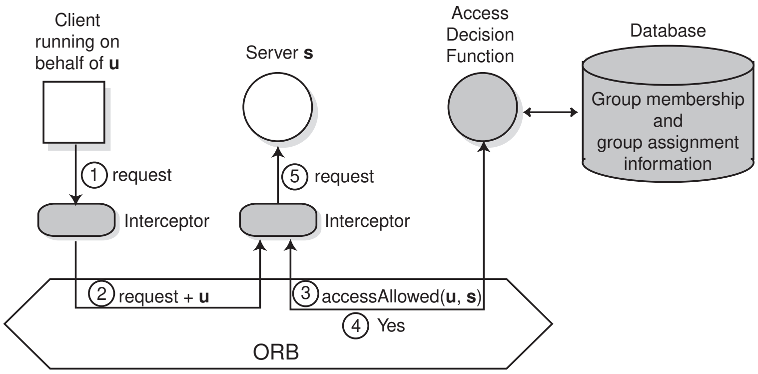

Figure 1 Implementation of the Access Control Model

4.2 Management of Closed User Groups

The management of the closed user groups is performed by registered users of the service centre. For this purpose, the CORBA interface of a dedicated CUG management object is exported by the provider. The client part that uses this interface has been implemented as an applet and is downloaded by the user from the service centre as part of the user side part of the service platform. Alternatively, the client part of the service implementation (e.g., a teleconference service) can contain the user side CUG management functionality in order to provide the seamless integration of the group management with the service in the same user interface.

Authorisation of CUG management is based on the identity of the originator and his role in the respective closed user group. Each group has two classes of members: a single administrator and an arbitrary number of ordinary members. The administrator has the rights of an ordinary member plus some additional privileges. As a matter of principle, each user can create a new closed user group if a group with the desired name does not already exist. Until he wishes to change this, the newly created group is administered by its creator. Administration can be offered to any member of the group. If a member accepts the offered administration, then he becomes the new administrator while the old administrator becomes a regular member of the group. Any user can join a closed user group if he is invited by the administrator of the group. Any member can leave the group or can be expelled from the group by the administrator. In general, all operations on a closed user group that trigger changes in the priviledges of the members are reserved for the administrator. All users in the environment can get information about their own membership in closed user groups.

The CUG management functionality in terms of operations is specified in the CUG management interface. Each operation requires the identity of the originator of the invocation as an input parameter. This identity is cryptographically authenticated using an authentication token. If the attempted invocation fails (e.g., authentication failure), an exception is raised. The exception always contains information about the reason,

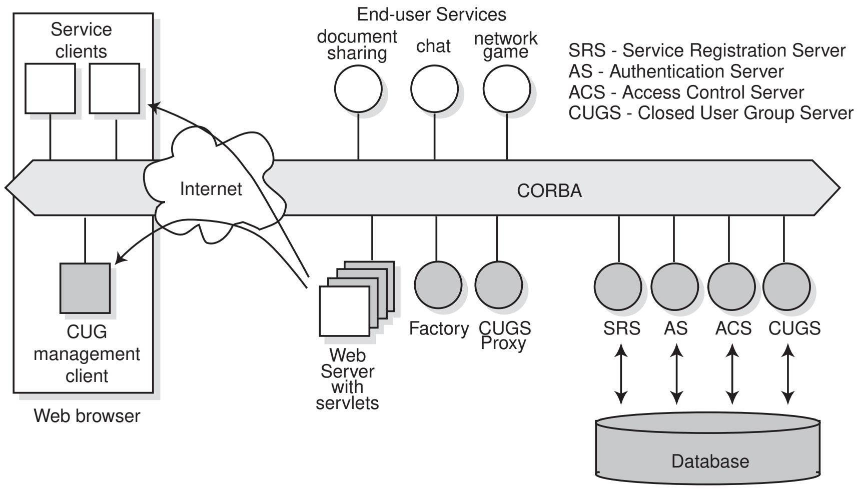

Figure 2 Architecture of the CLUE Prototype

why it has been raised. All functionality offered by the operations, including those reserved for CUG administrators, is unrestrictedly available for the provider, however through another interface, which also encompasses the actual user management and which is not exported to ordinary end-users.

5 REALISATION

In this section we describe our prototype implementation of the CLosed User group Environment. The architecture of the prototype is illustrated in Figure 2. The backbone of the architecture is CORBA together with WWW technology and Java applets as access technology. The end-user services (e.g., chat) are depicted in the upper part of the figure. They are realised as or wrapped up by CORBA server objects. The client parts of the services (service clients) are implemented as Java applets, downloaded to the users via the Internet using WWW technology, and executed by the users' web browsers. The service clients use the underlying CORBA layer, which is either built into the browser or downloaded together with the service client from the service centre, to interact with the end-user services. The components that realise the CLosed User group Environment are depicted in the lower part of the figure. They are implemented as CORBA server objects, Java applets, and Java servlets. The main components are the following:

- **Database:** It stores all the information about users (e.g., name and authentication information), running end-user service instances (e.g., object reference (IOR) and name of the assigned group), and groups (e.g., name and membership information).

- **Closed User Group Server (CUGS):** This server provides access to the user group information stored in the Database. It offers the group management operations to the users by which they can manage the closed user groups. The single operations are:

  - `createCug`,
  - `dissolveCug`,
  - `inviteToCug`,
  - `withdrawInvitationToCug`,
  - `expelFromCug`, `joinCug`,
  - `rejectInvitationToCug`,
  - `leaveCug`,
  - `offerAdministration`,
  - `withdrawOfferAdministration`,
  - `acceptOfferAdministration`,
  - `rejectOfferAdministration`,
  - `getCugs`,
  - `cugInfo`.

  The CUGS enforces its own access control policy at the application level (e.g., it ensures that only the administrator of a group can expel a member from the group).

- **CUG management client:** This Java applet is the client part of the CUGS. It basically implements a graphical user interface, which can be used by the users to manipulate the group membership information in the Database via invoking the management operations of the CUGS.

- **CUGS Proxy:** This proxy server is used to relay the messages between the CUG management client and the CUGS if the CUGS runs on a different computer than the web server. This is necessary because of the current Java applet security policy implemented in most web browsers, which does not allow an applet to connect to a computer other than the one, which it was downloaded from. Running the CLUE servers on a different machine than the web server has several advantages (e.g., hiding their object references, which include the internal IP addresses, from the users, load balancing, easy reconfiguration within the service centre, etc.)

- **Service Registration Server (SRS):** This server is used to register the object references of the currently running service instances, as well as their group assignment information.

- **Factory:** The Factory object starts up new instances of the end-user services upon request and registers their object references and group assignment information with the SRS.

- **Access Control Server (ACS):** This server implements the access decision function described in Subsection 4.1. It provides a single operation `accessAllowed`, which takes a user name and an object reference as input parameters and returns a boolean output according to the access decision. It takes the access control information (group membership of the given user and group assignment of the given service instance) from the Database.

- **Authentication Server (AS):** This server is used by the other CLUE servers, as well as the end-user servers to check the authentication information supplied by the users.

Each interaction between the web server of the service centre and the browsers of the users uses HTTP over an SSL connection (HTTPS) protocol, thus, it is cryptographically protected by SSL. Typically, the SSL protocol provides integrity and confidentiality protection of the messages exchanged, and authentication of the web server. We assume that not all of the users have digital certificates, thus, users are not authenticated in the SSL protocol. Therefore, a session between a user and the service centre begins with a login procedure, in which the user supplies a login name and a password. The login information is passed to a servlet, which calls the AS in order to check the authentication information. If the user is authenticated, then the AS generates a temporary authentication token, which is sent back to the user with the main page of the service centre. The user has to use this token for authenticating himself in every further requests sent to the service centre. The privacy of the authentication token is ensured by the protection provided by the SSL protocol.

Once the user has logged in, he can download the CUG management client applet and use the CUGS to modify the group membership information in the Database. The user can also start up new end-user services by specifying the name of the service and the group, to which he wants to assign the new service instance. After the verification of the authentication token by the AS and the matching of the specified group name with the user's group membership information by the CUGS, the Factory object is called, which starts up the new service instance, and registers it with the SRS. Then, the appropriate client applet is downloaded to the user.

Once the user has obtained a client applet (either by starting up a new instance of a service or specifying a running service instance), he can use it to send operation requests to the server. In the client side interceptor, each operation request is supplemented by the name and the authentication token of the user. This information is extracted from the request in the server side interceptor. The interceptor first calls the AS to verify the authentication token, then it calls the ACS, which decides whether the access to the server is allowed for the user or not. If the access is not allowed, then an exception is returned to the user.

We implemented the prototype using two ORB products. The CLUE servers, the servlets, and the CUG management client applet run on top of ORBacus/SSL from

Object Oriented Concepts [14]. The end-user services and their client applets use OrbixWeb/SSL from IONA [13]. All the components are written in Java. We run the SRS, the CUGS, the ACS, and the AS on a PC with Windows NT as the operating system. The Database is a MS Access database on the same machine, which is accessed by the servers via the JDBC-ODBC bridge. The web server, the Factory, the CUGS Proxy, and the end-user services run on a Sun workstation with a Unix operating system (Solaris). The SSL and servlet powered web server is part of the Sun Java Server package.

6 CONCLUSION AND FURTHER WORK

We presented a model, the CLosed User group Environment (CLUE), for end-user directed access control to services in Internet service centres, and the realisation of this model with off-the-shelf software for a middleware based service environment, which is characterised by the use of CORBA, Java, and WWW technology. The model is based on the concept of closed user groups, the membership of which is managed by the end-users.

Acceptance evaluation with test persons using CLUE enhanced primary services and the auxiliary service for the user-controlled management of closed user groups, using the respective graphical user interface, have shown that the notion of closed user groups as a means to provide security functionality is a promising model, which makes security understandable and manageable for ordinary end-users and allows the service provider to distinguish itself in the market of service provision on the Internet. Evaluation tests of the CLUE with application programmers have shown that the integration of the CLUE into already existing distributed applications, which have not been designed with the existence of CLUE in mind, was an easy standard programming task. Thus, CLUE can be regarded as an easily applicable solution for enforcing access control. Evaluation with test administrators of service centres has confirmed that the CLUE solution for authorisation allows easy administration of access control.

Future work will be concerned with transaction management, with the provision of adequate logging and auditing functionality, which will allow the service provider to detect and trace potential violations of the security of the service centre, with the prevention of denial-of-service attacks (e.g., flooding) and with the provision of cryptographic multicast within closed user groups. Another issue to be examined is the scalability of the proposed model.

References

[1] Decina, M. and Trecordi, V., *Convergence of Telecommunications and Computing to Networking Models for Integrated Services and Applications*. In Proceedings of the IEEE, Special Issue on the Global Information Infrastructure, December 1997, pages 1887-1914.

[2] Ford, W., *Computer and Communications Security*. Prentice Hall, Englewood Cliffs, 1994.

[3] Freier, A. O., Karlton, P., and Kocher, P. C., *The SSL Protocol*. Version 3.0., Internet Draft, http://home.netscape.com/eng/ssl3/

[4] Hubaux, J-P., Gbaguidi, C., Koppenhoefer, S., and Le Boudec, J-Y., *The Impact of the Internet on Telecommunication Architectures*. Computer Networks and ISDN Systems 31(3), Special Issue on Internet Telephony, February 1999.

[5] Schoenbauer, H., Zoernack, A., Hussmann, H., Sevcik, M., DeZen, G., Marsiglia, M., Ricagni, G., and Vezzoli, L., *Value-added Internet: a pragmatic TINA-based path to the Internet and PSTN Integration*. In Proceedings of the TINA'97 Conference, Santiago, Chile, November 1997.

[6] Smith, C., *Applying TINA-C Service Architecture to the Internet and Intranets*. In Proceedings of the TINA'97 Conference, Santiago, Chile, November 1997.

[7] Staamann, S., Buttyán, L., and Wilhelm, U., *Security in TINA*. In Proceedings of the IFIP/SEC'98 14th International Information Security Conference, Vienna/Budapest, August-September 1998, (also available from http://lsewww.epfl.ch/projects/crystina/)

[8] Woo Sun, H., Eun Chul, K., and Hee Kyung, J., *Realization of TINA Service Architecture on the Internet*. In Proceedings of the TINA'96 Conference, Heidelberg, Germany, September 1996, pages 235-243.

[9] ISO/IEC 9594-8, *Information Technology - Open Systems Interconnections - The Directory: Authentication Framework*. (Also ITU-T Recommendation X.509).

[10] Telecommunications Information Networking Architecture Consortium, http://www.tinac.com/

[11] Telecommunications Information Networking Architecture Consortium, *TINA-C Service Architecture 6.0*. http://www.tinac.com/

[12] Object Management Group, *The Common Object Request Broker*. Architecture and Specification, Revision 2.2., http://www.omg.org/

[13] IONA Technologies, *OrbixWeb online product information*. http://www.iona.com/products/internet/orbixweb/

[14] Object Oriented Concepts Inc., *ORBacus for C++ and Java*. http://www.ooc.com/ob/

[15] Amazon on-line book-shop, http://www.amazon.com/

[16] PointCast Network, http://www.pointcast.com/

Biography

Sebastian Staamann is a researcher in the Operating Systems Laboratory (LSE) in the Computer Science Department of the Swiss Federal Institute of Technology in Lausanne (EPFL). From 1992 until 1997 he taught information and communication security at the University of Technology Berlin. At the EPFL he is conducting research projects concerned with security in middleware based service architectures. His research interests are security and privacy issues in computer and telecommunications networks, cryptography, and electronic commerce.

Levente Buttyán is a research and teaching assistant, and PhD student in the Institute for computer Communications and Applications (ICA) in the Computer Science Department of the EPFL. He received the M.Sc. degree in computer science from the Technical University of Budapest (TUB) in 1995. Before joining the EPFL, he spent one and a half year as a postgraduate student at the Department of Telecommunications at the TUB, where he was concerned with the design issues of conventional block ciphers. His main areas of interest are cryptography and security in distributed systems, and in particular the use of formal methods in the design of authentication protocols.

Allan Coignet is a research and teaching assistant, and PhD student at the LSE. He received the M.Sc. degree in computer science from the EPFL in 1998. His main areas of interest are middleware and security in distributed systems.

Ernesto Ruggiano is a software engineer working for Swisscom Corporate Technology. His current activities include middleware technology integration, distributed systems, electronic commerce, and Internet technology. He has experiences in the development of object oriented client/server applications and intelligent agent frameworks for network management.

Uwe Wilhelm is an assistant and PhD student in the LSE. He received a diploma in computer science from the University of Kaiserslautern, Germany, in 1992, after which he spent a year as researcher at Cornell University, USA. Before joining the EPFL he spent several months as a private consultant for IDS. His main areas of interest are privacy, data protection, and security in computer and telecommunications networks as well as reliability in distributed systems. His current work is centred on security issues in the mobile agent paradigm.

Marc Zweiacker is a program manager at Swisscom Corporate Technology. During the last seven years he worked in the areas of fault tolerant distributed systems, middleware technologies, electronic commerce, and interoperability standards (e.g., CORBA and RM-ODP). His program, called *Business Services*, is the leading source of the know-how for advanced services and applications that support a range of business branches dominated by the small and medium-sized enterprises (SMEs) in Switzerland. The program assists the development of complete solutions for selected business sectors such as software industry, transportation, and healthcare.