Cryptographic Protection of Connection Integrity with Interruption Detection in TINA

S. Staamann, U. Wilhelm
Swiss Federal Institute of Technology - Lausanne
EPFL-DI-LSE, CH-1015 Lausanne, Switzerland,
fon: +41 21 693 5267, fax: +41 21 693 6770,
email: {staa, wilhelm} @lse.epfl.ch

Abstract

The Telecommunications Information Networking Architecture (TINA) provides the architectural basis for the realization of telecommunications services as distributed applications. In particular application of distributed systems, new security problems arise. We discuss threats that are caused by the interruption of service control connections. A method to detect interruptions is presented and the handling of interruptions is described. The detection method is based on a cryptographic protocol using sequence numbers and probe messages. TINA and its security approaches are introduced. The threats by interruption are illustrated using a toll fraud example.

Keywords

TINA, Connection Integrity, Service Control Interruption, Security States, Sequence Numbers, Probe Messages

1 INTRODUCTION

Computers are increasingly used for the control of telecommunications systems. The use of computers enables more flexibility in control as well as the fast and cheap introduction of new telecommunications services. The basic ideas for this approach come 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. A consistent effort to specify an architecture that supports the implementation of telecommunications services as distributed applications is the Telecommunications Information Networking Architecture (TINA) (Barr *et al.* 1993). Roughly speaking, TINA is a modern approach to telecommunications since it looks at this field from a distributed systems point of view. In TINA, services are composed of service components. These components are built of distributed interacting software units, called computational objects (COs). COs reside and interact in a Distributed Processing Environment (DPE). The DPE is realized using middleware, namely CORBA (OMG 1995_2) implementations.

An essential requirement of distributed applications is security. Thus, the CORBA architecture includes a security service (OMG 1995_1) that provides basic security features, such as authentication of users and data origin, access control, integrity and confidentiality of transmitted data, and non-repudiation. The TINA architecture supports a large variety of services, multiparty communications, flexible accounting and billing schemes, and enables the competition of providers within one network. The security problem domain in TINA is not completely covered by the CORBA security architecture. For instance, the complexity of service control enables inside and outside attackers to take advantage of deliberately caused interruptions of message flows in service control interactions. Particularly, the flexibility of accounting and billing schemes or multiparty communications may attract such attacks on service control that cannot easily be prevented. In this article, we analyze threats to user security that may be caused by the interruption of service control connections. We introduce the notion of states regarding security as proposed in (Staamann 1997) and present a cryptographic protocol for the protection of service control connections. Section 2 introduces the TINA architecture and its relevant concepts. * Acquaintance with TINA is not assumed, but some familiarity with the basic concepts of CORBA is expected. Section 3 presents the service control concepts in TINA. Section 4 gives an overview of the current state of TINA security. Subsequently, we demonstrate threats that are caused by service control interruptions using a toll fraud example. In Section 6 we introduce a method for preventing this kind of attack and Section 7 presents the corresponding cryptographic protocol.

2 THE TINA ARCHITECTURE

The Telecommunications Information Networking Architecture (TINA) (Barr *et al.* 1993) is an effort to define an open architecture for telecommunications

*The introduction represents the state at the time of writing (May 1997). Since the TINA specifications are still under development, there may be changes until the time of publication.

services 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 telecommunications equipment and computer system suppliers. The architecture is based on distributed computing and object orientation. It comprises concepts from ODP, IN, TMN, and CORBA. TINA is applicable to broadband and narrowband networks. As already mentioned, in TINA services consist of service components. Each service component is realized as one or a number of computational objects. Service components and COs interact with each other via the distributed processing environment. The DPE is a software sub-layer that operates above a native computing and communications environment (NCCE). While the NCCE is technology dependent, the DPE offers a uniform interface to the distributed environment. It is widely expected that the DPE will consist of CORBA implementations as the DPE kernel and additional TINA specific services, such as traders and name servers.

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 usage and 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, NCCE, and computing resources as well as the structuring in component categories. More detailed descriptions of the TINA overall architecture (TINA-C 1995) can be found in (Dupuy *et al.* 1995) and (Nilsson *et al.* 1995).

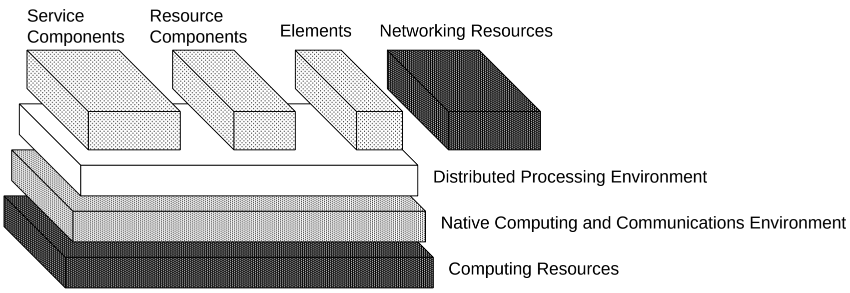

Figure 1 TINA overall architecture

The TINA Computing Architecture provides the concepts for how service components and computational objects are specified and how they interact. It defines the DPE as the computer and communications platform support and provides the computational modelling concepts, such as the Object Description Language (ODL) that is a superset of CORBA's IDL. The main enhancements are the introduction of multiple interfaces for a single object, stream interfaces, and the possible grouping of objects. Thus, there are two different kinds of interfaces: operational interfaces that are comparable to object interfaces in CORBA and stream interfaces that do not offer operations, but are designed to convey an arbitrary sequence of bytes between two objects, e.g., audio or video bit streams.

The business model of TINA is 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 other 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. An example may illustrate this: Consumers A, B, and C are involved in a video conference. They are all customers of Retailer R and get the video conference service from R. Retailer R does not provide the service itself, but has a wholesale agreement with a Third Party Service Provider P that provides the actual service. In this example, there are four user-provider relationships: A-R, B-R, C-R, and R-P. A user-provider relationship contains three types of interaction: access, primary usage, and ancillary usage. The access part is concerned with the establishment of a trusted and reliable temporary relationship between the user's current* domain and the provider domain. Such a relationship is a prerequisite for usage interactions. Primary usage is related to the actual service(s) the user has subscribed to. Ancillary usage assists the user in the contract with the provider, e.g., online subscription of primary services. Interoperability between domains is guaranteed by the definition of interdomain reference points.

3 SERVICE CONTROL IN TINA

The traditional call concept of telecommunications is substituted by the more flexible session concept. There are different types of sessions, according to the aspect under consideration, i.e. access and usage, service and communication links, or the allocation of functionality to administrative domains. At the service level there are access sessions and service sessions. A service session is

*TINA inherently supports personal mobility, i.e. the user can act in various domains under the same identity.

an instance of a service type. Exactly one provider is involved in a service session. One or more users participate in the service session. The service session consists of the provider service session with the global service logic in the provider domain and the user service sessions. For each user, there is one user service session that is concerned with the user specific part of the service session. The user service session consists of two parts, one in the provider domain and one in the user domain. A service session or a user service session, i.e. a user's participation in a service session, can be suspended and resumed later on, potentially from another user domain (session mobility). 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. Two persistent sessions in their domains are involved in the access session: the user domain access session and the provider domain access session. The access session is established by interaction of these two sessions. Content information of services is passed in the form of streams. The communications session establishes and maintains the necessary stream connections for a service. It is controlled by the service session.

Sessions and other information objects are mapped onto service components. Service control is achieved by the interaction of service components in the users' and provider's administrative domains. Figure 2 shows the service components in their administrative domains and their relation to different sessions using the example of a conference with two participants. We will briefly explain the service components and their interactions. For the sake of brevity, some aspects are omitted or simplified.

The access related User Application (AUAP), the Provider Agent (PA), the Initial Agent (IA), and the User Agent (UA) belong to the access session. AUAP models applications in the user domain that support the access session. It provides the user interface for this purpose. The PA is the essential embodiment of the user domain access session, the provider domain access session is mapped onto IA and UA. AUAP, PA, IA, and UA are service independent. The IA is the initial contact point of the provider for the users; it is also user independent. For establishing an access session, the PA in the user domain contacts the IA in the provider domain. After the authentication, a reference to the UA is delivered to the PA. The UA represents the user in the provider domain. It is the user's contact point to start or resume a service session or the user's participation in a service session. The lifespan of the UA is the time of subscription of the user at the provider, the IA exists as long as the provider is active in his business.

The service session is mapped onto the service related User Application (SUAP), Service Factory (SF), Service Session Manager (SSM), and User Session Manager (USM). The SUAP models applications in the user domain that support the service. It embodies the user service session in the user domain. The SUAP is the endpoint of content information flows (streams) and it facilitates service control in the user domain. To achieve the latter, it inter-

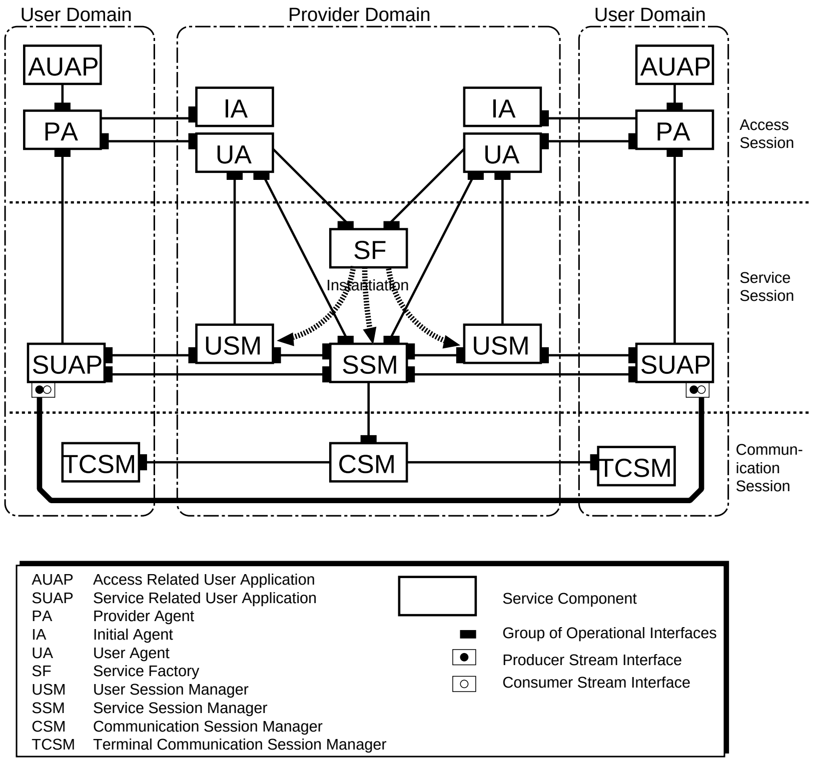

Figure 2 TINA Service Components

acts with the USM and the SSM in the provider domain. The SSM contains the global service logic; it embodies the provider service session. The USM represents the user related aspects of the service logic in the provider domain; it embodies the provider domain part of the user service session. Both are service specific, i.e. they support service specific and generic control functions. Generic functions are start, suspend, resume, and end of a service session or the user's participation in a service session. An example for a service specific function is the rewind function of a video-on-demand service. Obviously, all the control functions of a service, both generic and service specific ones, must be supported by the SUAP for the service. Thus, the SUAP is also service specific. In contrast to the UA, the SSM and the USM have a short lifespan. The SSM exists for the duration of the service session, the USM for the duration of the user service session, i.e. as long as the user participates in the service session. Both are created by the SF on request from the UA. For each new service session, the SF creates the SSM and the USM for the initiating user. For each user who joins an existing service session, an additional USM for the joining user is created by the SF. The Communication Session Manager (CSM) and the Terminal Communication Session Manager (TCSM) are concerned with the establishment of streams between service components, such as the SUAP in the user domain and the SSM and the USM in the provider domain. The CSM is controlled by the SSM. The detailed specification of service related aspects in TINA, including service control, is given in (TINA-C 1996). A good, but now slightly outdated, description can be found in (Yagi *et al.* 1995).

4 TINA SECURITY PROBLEM DOMAIN

From the consumer's point of view, security is mainly concerned with preserving integrity and confidentiality of transmitted information. The means for protecting transmissions over untrusted communication links are provided by cryptography. In TINA, all contents information is transferred in the form of streams between producers and consumers of this information. Authenticity, integrity, and confidentiality of the information streams can be achieved as end-to-end-security between the communicating parties trough using standard cryptographic mechanisms, such as message authentication codes (MACs) and encipherment. The generation of a MAC is shown in Figure 3a. For performance reasons, symmetric ciphers, such as DES (DES 1977) or IDEA (Lai 1992) in appropriate modes or specific stream ciphers*, should be used for encipherment. The remaining problem is the negotiation of keys. TINA systems must enable secure communications in case of first contact. In such cases of spontaneous communication, the communicating parties do not have a common key. Thus, distribution of symmetric keys for end-to-end security should be supported by TINA.

From the system perspective, the most important aspects of security are the protection of the single administrative domains against intrusion, the authentication of users and providers, the correctness of service executions, and the prevention of unauthorized service usage. Two-party-authentication is done during the establishment of the access session. The authentication scheme and the trusted third party (or certification authority for schemes based on asymmetric cryptography) are negotiated and the authentication protocol is executed. As a possible and desirable result, both sides should obtain a session key that can be used for all interactions between user and provider during the access session.

Unauthorized service usage is prevented at the service level. Each request for a new user service session (a user wants to join an existing service session) or for a new service session (start of a service) is checked before actions take place. The UA checks the subscription information, i.e. whether the user has subscribed to this service and under which conditions the user may use the

*There may be legal constraints to the choice of cryptosystems and keys, e.g., regulations may prescribe the use of escrowed keys.

service. If the session already exists, the SSM checks additionally, whether the state of the service logic allows the user to join the service session. Similar authorization checks have to be executed for suspending and resuming service sessions and user service sessions. Protection against illegitimate access to single service components or computational objects is achieved by access control at the DPE level. A more detailed analysis of the security problem domain in TINA, including the architectural embedding of access control, is given in (Staamann and Wilhelm 1997).

The correctness of service execution requires the authenticity and integrity of the exchanged service control information, i.e. the information passed via operational interfaces. For user privacy, confidentiality of this information should also be provided. The above requirements can be fulfilled by the use of security services available in the DPE. CORBA implementations that comply with the CORBA security specifications (OMG 1995_1) will provide the means to guarantee authenticity, confidentiality, and message integrity using the session key of the access session. The CORBA security specification allows various underlying security technologies, such as Kerberos (Neuman and Ts'o 1994) and SESAME (Kaijser *et al.* 1994). However, it is still an open question how CORBA systems with different security technologies will be able to securely interoperate. This is crucial for TINA, since it is widely expected that the DPE kernel will be provided by at least one ORB system in each TINA administrative domain. In TINA, the use of security mechanisms and formats should be negotiated as part of access session establishment. Provided that the respective mechanisms are available as part of the security technology in both domains, interoperability can be achieved in this way.

5 THREATS TO SERVICE CONTROL BY INTERRUPTION

Obvious threats to service control are the masquerade of attackers as subscribed users or the modification of service control messages. Masquerading is prevented by authentication as described above. The modification of single service control messages is prevented by security services that detect the modification of a message. This is achieved by using cryptographic mechanisms, such as message authentication codes (Figure 3a). The granularity of integrity is neither mentioned in the TINA nor in the CORBA specifications. It can be assumed that parameters, results, and exceptions of operation invocations are the units for which integrity is guaranteed. This means that complete messages at this level (invocation with parameters, results, exceptions) can be subject to interception, replay, and reordering. Attackers can thus deliberately disturb service control to gain an advantage. An obvious instance of such an exploitation is the illegitimate use of services at the cost of another user. The following, rather unconventional example may illustrate this.

Home entertainment is expected to be an important business area for the provision of services in TINA networks. Many Retailers will offer video games as services to their Consumers. We choose a multi user game service as our example. For simplicity, we assume that it is provided by the Retailer itself, i.e. no additional Third Party Service Provider is involved. One of the main goals in TINA is the support of complete flexibility for accounting and billing. In our example, a sensible accounting scheme could be that the user who creates the service session (starts the game) has the control over the global aspects of the service (chairing) and is billed for the session as long as he chairs it. After starting the game, other players can join the session. A service specific operation allows to hand over accountability and chairing to another player. In case, the chair wants to resign and no other player takes over, the chair may break off the game.

Since the communication links between user domain and provider domain are not physically secure, other players may intercept or replay messages of a player. To improve his own chances of winning, a player could impede another one by intercepting messages. This kind of attack, although in the case of our example not a real concern, can be prevented by the use of sequence numbers or cryptographic chaining in the generation of message authentication codes. The use of sequence numbers is shown in Figure 3b. These mechanisms detect the loss, replay, and reordering of messages within a sequence, but they do not detect complete interruption. In our example, interruption can be used for accounting fraud. Assume the following scenario: The chair announces that he wants to resign. No other player takes over chair and accountability. Instead, one of the other players interrupts the link between chair and provider. As result, the chair remains accountable for the service session, although he cannot control it anymore. The other players can continue to use the service at his cost and against his will. Note, that physical protection of subscriber lines is practically impossible and complete interruption of a communication link is physically even easier than replay or reordering. In the case of a later legal dispute, the usufructuaries of the attack can hardly be made responsible.

One could imagine similar attacks on many other multi-user services, such as conferences, joint editing, etc. The general problem is that users remain in an accountable state, even if the control connection is interrupted. From a more general perspective, the distinction can be made between secure and critical states of a service participation. An accountable state is just one instance of a critical state. Another instance is the exposure of data to access by other users for a limited time interval in the course of a service. A critical state is any state of a user service session or provider service session in which the user has a further disadvantage other than simply the unavailability of the service if the connection between user domain and provider domain is interrupted.

6 HANDLING OF CRITICAL STATES

The threats that result from critical states can be minimized if the respective session passes from the critical state into a secure state when the service control connection is interrupted. Which state is considered secure and which critical is service dependent. It is the responsibility of the service designer to categorize states of sessions as secure or critical. Both parts of a service session are relevant for the analysis: the provider service session (between the SUAP and the SSM) and the user service session (between the SUAP and the USM).

Each session must be analyzed by the service designer whether the session can get stuck in a critical state when interdomain control connections are interrupted. The result of this assessment is a list of critical states of this session. Each of these states is dependent on one or more relevant interdomain control connections. For each pair of critical state and relevant connection, the designer specifies a secure state into which the session should pass if the relevant connection is interrupted. Critical states of user service sessions are only dependent on the USM-SUAP control connection, but the provider service session may be dependent on various SSM-SUAP connections. For the provider service session, there may be transitions to different states from the same critical state, since the participating users may act in different roles (in the multi user game example: chair and ordinary players) and the relevance of their control connections may differ.

To initiate the specified state transition for a pair of critical state and control connection, the logic in the respective service component (USM or SSM) must detect the interruption of the control connection. A prerequisite for this is the employment of a security mechanism that allows to detect the connection interruption. The following section presents a cryptographic protocol for the protection of overall connection integrity, including the protection against interruption. Since the use of this protocol causes additional traffic between user and provider domain, the "protection against interruption" feature of the security service should not always be activated. It is switched on by the using service component (USM or SSM) before passing into a critical state and switched off after leaving this state. When the feature is activated and the interruption of the relevant connection is detected by the service, it triggers the specified state transition.

7 INTEGRITY OF SERVICE CONTROL CONNECTIONS

The only way to check whether a connection still exists is the exchange of probe messages. We demonstrated that it is important for service components in the provider domain to detect an interruption. We could not identify any threat, except the unavailability of service control functionality, that would result if the user does not detect the interruption of the service control connection.* Thus, in our protocol, probe messages are only sent from the user to the provider domain. This means that if the service is interrupted, the user is not automatically aware of this. Besides security considerations, this might be handled at the application level using the traditional timeout concept. For performance reasons, the protection of the connection itself is combined with message and sequence integrity in the same protocol. Sequence integrity of messages can be achieved by the use of sequence numbers or by cryptographic chaining of consecutive messages. In contrast to the cryptographic chaining approach, the use of sequence numbers allows the receiver to tolerate the loss of one or more messages without loosing the chance to verify the subsequent messages. Thus, we decided in favour of the sequence number approach. Nevertheless, the protocol can be modified to use cryptographic chaining.

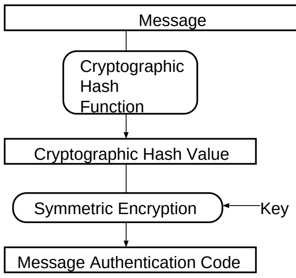

Figure 4a
A Method for the Generation
of Message Authentication Codes

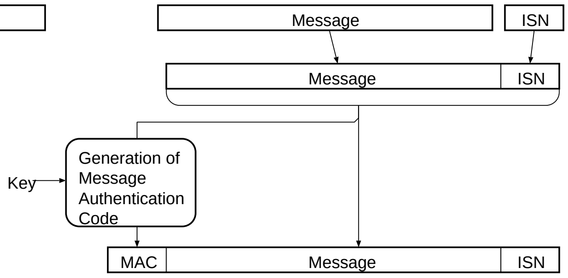

Figure 4b
Sequence Integrity by Use of
Integrity Sequence Numbers

The protocol is simple and the sequence aspect is based on well known cryptographic techniques. As a prerequisite, it requires that both parties already have a common session key for use in the service session. It seems reasonable that the same session key that was distributed to both parties as the result of the authentication during establishment of the access session is used for the protocol. There are no additional (stronger) requirements on the security and usage time of the session key than for simple message integrity protection. To protect sequence integrity, both parties use sequence numbers. To each message, the sender adds a sequence number and generates a message authentication code over both using the session key (Figure 3b). The receiver checks the MAC and compares the current sequence number with the one of the previous message. If the difference is not equal to 1, the receiver knows that the sequence integrity was violated.

*In other distributed applications, the requirement for both sides to detect the interruption may appear.

When the "protection against interruption" feature is activated by the service component in the provider domain, a control message at the security service level containing a timeout_time as a parameter is sent to the respective security service entity of the SUAP in the user domain. At the same time, the security service entity for the connection in the provider domain starts its clock and waits for a message from the user's side. Every time when a message from the user domain has arrived, the clock is reset. If no message arrives within the timeout since the last clock reset, the state transition is triggered. This means, in case the path between user and provider is immediately interrupted and the activation message does not arrive at the user's side, the critical state is left after the timeout at the provider side. In the user's domain, after receiving the activation message, a clock is started. If no service control message is sent within the timeout, a probe message is sent. Figure 4 illustrates this.

Probe messages contain only the sequence number and padding bits, but are also integrity protected by adding a MAC. When a message is sent, regardless whether service control or probe, the user domain clock is reset. Service control messages and probe messages are numbered in the same sequence. When the "protection against interruption" feature is deactivated by the service component in the provider domain (the session has left the critical state), a control message at the security service level is sent to the user domain to inform it and the clock in the provider domain is halted. On receipt of the deactivation message in the user domain, the sending of probe messages is stopped. The cryptographic strength of the protocol is the same as that of the established pure sequence number approach for the protection of the sequence of messages. Note that the protocol does not depend on absolute system times but on time periods that do not differ significantly between systems. For the SUAP-USM connection, Figure 4 shows the passing of messages at the security service level for an example period without irregularities.

The timeout in the user domain t_u is chosen smaller than the one in the provider domain t_p in order to take the maximum transmission time for control messages t_t and the processing time t_g for the generation and sending of the probe message tolerated into account. It is determined at the user's side at runtime, so that $t_p \geq t_u + 2 * t_t + t_g$, where t_t and t_g are assumed to be fixed values. The timeout is a variable parameter. It is determined by the service designer and the service provider. Reasonable values depend on the time sensitivity of the critical state, the acceptable traffic overhead, and the maximum transmission time of a message in the control network. It has to be chosen at least so that: $t_p \geq 2 * t_t + t_g$. In cases where interruption of a short duration may be sufficient to cause serious damage, the scheme could be varied such that the provider does not pass into the critical state, before the "reply message" to the activation message has arrived. This would prevent that the critical state is entered at all if the path between user and provider has already been cut off. In this variation, the "reply message" is sent immediately

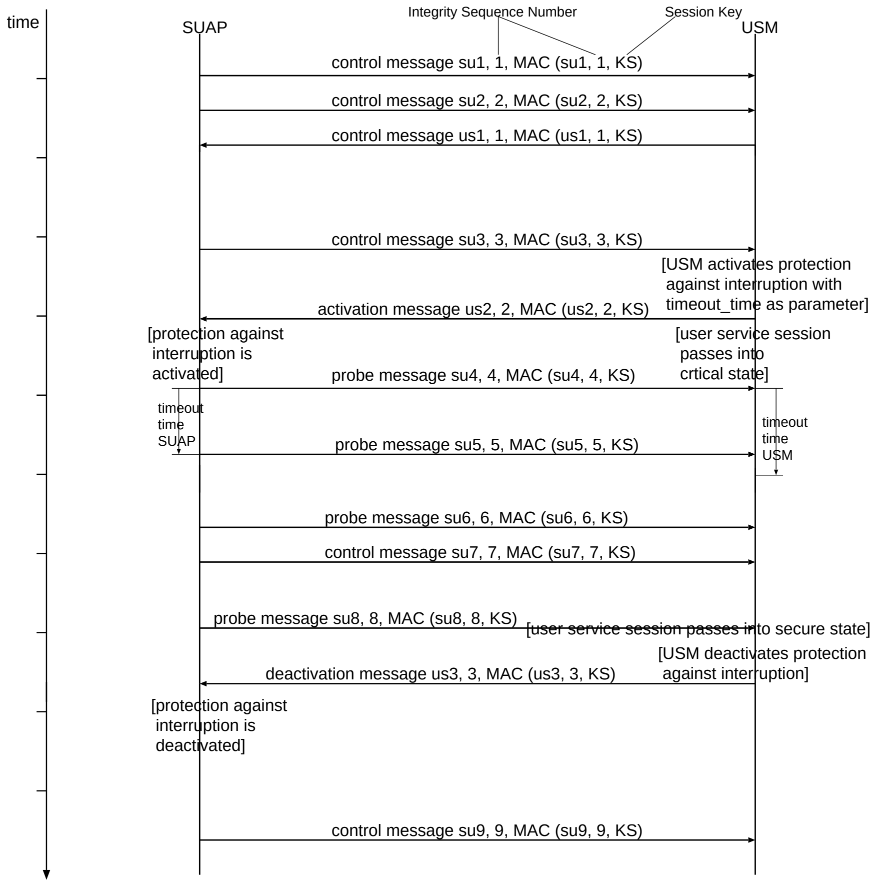

Figure 4 Sample Message Transmission Event Trace

at the user's side. The protocol can be used in conjunction with the "integrity with recovery" feature implementations of security services that use the pure retransmission approach. In this case, each retransmitted message obtains a new sequence number. This way, retransmission messages are also used as message events for the protection against interruption and probe messages are saved.

8 CONCLUSION

The Telecommunications Information Networking Architecture is a flexible and promising architecture for future telecommunications networks. Nevertheless, the considerable flexibility of service control causes new threats to security that cannot be prevented by the use of established security mechanisms. It was demonstrated that the interruption of service control connections may cause damage, such as toll fraud. We proposed a method to prevent this kind of attack. A cryptographic protocol for this method was introduced. Future work will be concerned with the realization of the cryptographic protocol as part of a secure and reliable DPE. Furthermore, it should be investigated whether the notion of critical states is also reasonable for distributed applications other than telecommunications and whether it is required that both sides are able to detect the interruptions.

ACKNOWLEDGMENTS

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. The authors also wish to thank the reviewers for helpful comments.

REFERENCES

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

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.

Kaijser, P. Parker, T. and Pinkas, D. (1994) SESAME: The Solution to Security for Open Distributed Systems. *Journal of Computer Communications*, July 1994, 501-518.

Lai, X. (1992) On the Design and Security of Block Ciphers. *ETH Series in Information Processing*, vol. 1, Hartung-Gorre-Verlag, Konstanz, 1992.

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

National Bureau of Standards (1977) Data Encryption Standard. *Federal Information Processing Standards Publication* 46.

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

Nilsson, G. Dupuy, F. and Chapman, M. (1995) An Overview of the Telecommunications Networking Architecture. *TINA Conference 1995*, 1-12.

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

Object Management Group (1995) CORBA Security.

Staamann, S. (1997) Overall Integrity of Service Control in TINA Networks. *Communications and Multimedia Security Conference '97*, Athens, Greece, September 1997.

Staamann, S. and Wilhelm, U. (1997) CORBA as the Core of the TINA-DPE: A View from the Security Perspective. *Distributed Object Computing in Telecommunications 1997*, Frankfurt a.M., Germany, October 1997.

TINA-C Document TB_MDC.018_1.0_94 (1995) Overall Concepts and Principles of TINA.

TINA-C Document TB_RM.001_4.0_96 (1996) Service Architecture, Version 4.0.

Yagi, H. Ohtsu, K. Kazuyuki, W. Wakano, M. Kobayashi, H. and Minerva, R. (1995) TINA-C Service Components. *TINA Conference 1995*, 113-123.

BIOGRAPHIES

Sebastian Staamann is a researcher at the Operating Systems Laboratory (LSE) in the Computer Science Department (DI) of the Swiss Federal Institute of Technology in Lausanne (EPFL). He received a diploma in electrical engineering from the Technical University Berlin (TUB) and is currently finishing his doctoral dissertation in computer science. From 1992 until spring 1997 he tought information and communication security at TUB. At EPFL he is conducting a research project aiming at a secure and reliable TINA-DPE. His research interests are security and privacy issues in computer and telecommunications networks, cryptography, and reliability in distributed systems.

Uwe Wilhelm is an assistant and PhD student at the Operating Systems Laboratory (LSE) in the Computer Science Department (DI) of the Swiss Federal Institute of Technology (EPFL). He received a diploma in computer science from the University of Kaiserslautern in 1992, after which he spent a year as researcher in the USA at Cornell University. 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.