A SECURE PLATFORM SUPPORT ENVIRONMENT FOR TMN APPLICATIONS

J.CELESTINO Jr. jc@masi.uvsq.fr
A.HUSSEIN hua@masi.uvsq.fr
N.MELO melon@masi.iuvsq.fr

University of Paris VI
PRISM Laboratory
45, av. des Etats Unis
78000 - Versailles
France

ABSTRACT

The Telecommunications Management Networks (TMN) have been evolving and becoming more sophisticated. Different technologies from different vendors contribute to complicate the telecommunication management so that the need for integration and interoperability is growing. A general TMN computing platform architecture was proposed by the TMN Computing Platform Special Interest Group (RACE programme) whose main objective is to provide enough support for the contrasting TMN management applications. This paper discusses the Secure Platform Support Environment in the IDEA project (PRISM Laboratory) and its relationship with this general TMN computing platform architecture. The important services suported by both the implementation developed by the IDEA project and the CP-SIG architecture are identified and discussed.

1. Introduction

The Integrated Broadband Communication Network (IBCN) is composed of a multitude of Network Elements with different technologies, from different manufacturers, which will be introduced at different rates across Europe. This complex environment needs to be managed in an efficient way, so that new applications and services are possible at a reasonable cost. The complexity increases when, beyond the technological aspects, one considers the regulatory and national environments, as well as the interconnection with existing and near future networks like ISDN.

The TMN management systems represent diverse telecommunication domains with ride ranging functionalities and requirements. On the one hand TMN management applications have different purposes, frequently employing different technologies, while on the other hand there is strong need for application integration and interoperability. In order to achieve this integration and realize open solutions an underlying TMN computing platform must offer the necessary support environments, connectivity and flexibility to allow management applications to achieve their goals[1].

2. A General Computing Platform for TMNs

A Computing Platform is a set of facilities that provides a defined infrastructure upon which management applications can be built. The access point of these facilities and the underlying systems is an interface provided by the platform that management applications should use to intercommunicate and to access the network information.

A General TMN Computing Platform architecture, capable of supporting a wide variety of TMN management applications have been developed by the CPSIG group [1,2]. It consists of five layers, figure 1. The goal of these layers is to abstract away the complexity and heterogeneity of the underlying host systems and communications protocols. The layers also provide transparencies for distribution, information access, and replication. These five layers are (i) CP Kernel, (ii) Distributed Processing Support, (iii) Computing Platform Interface, (iv) TMN Support Environment, (v) User Generic Applications. CP-Kernel, Distributed Processing Support and Computing Platform Interface layers are grouped in layer called Platform Support Environment.

The CP-kernel consists of a number of components that mask heterogeneity, and provides a common view of the world.

The Distributed Processing Support layer essentially provides abstraction from the distribution of the applications.

The Computing Platform Interface provides the link between the programs of the application developer and the underlying processing support.

The TMN Support Environment provides the platform services that are required by the TMN Management Applications and the User Generic Functions to fulfill their respective roles in managing the telecommunication network.

User Generic Functions perform key tasks which are required across a range of TMN applications.

The PRISM Laboratory has a project called IDEA which has developed its own Platform Support Environment. This project IDEA has served as support in anothers European projects as : PEMMON/ESPRIT and ADVANCE/RACE. The design and implementation of this Platform Support Environment have spent two years and have followed several phases.

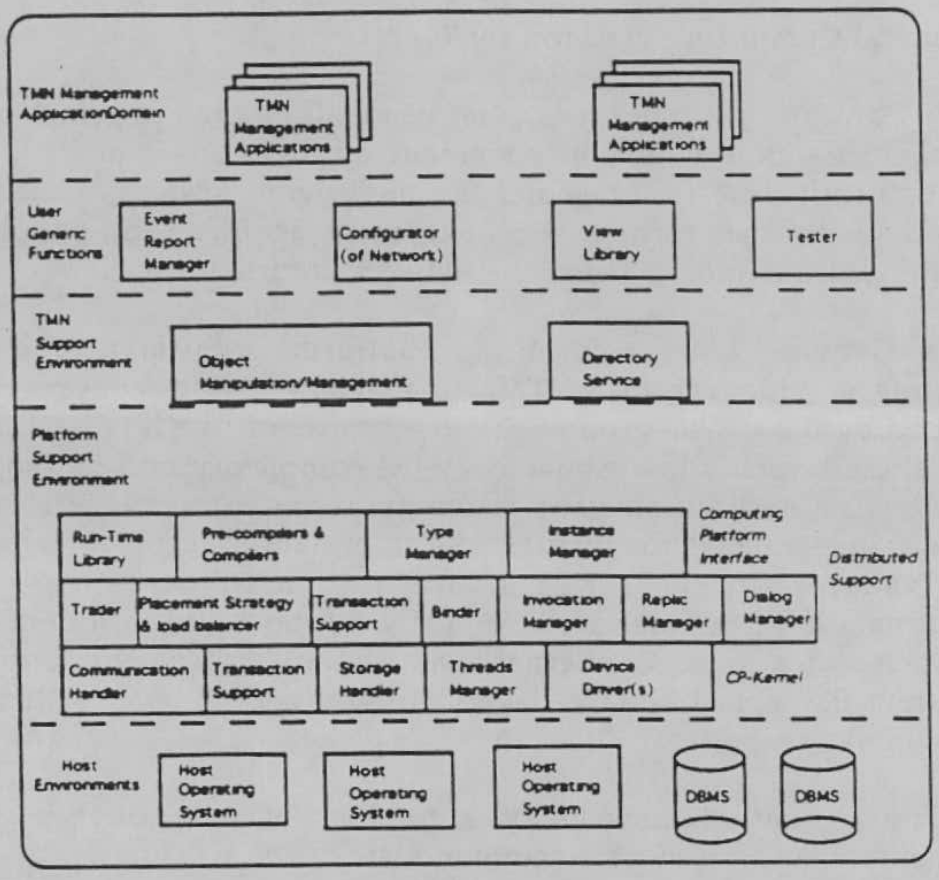

Figure 2.1 - The General Computing Platform

3. The Platform Support Environment

As shown above, the Platform Support Environment is composed by three layers: Computing Platform Interface, Distributed Processing Support and CP-Kernel. The aim in this section is to describe which components must be included in each layer.

3.1. The Computing Platform Interface

The Computing Platform Interface would include the following objects:

Pre-Compilers and Compilers : The applications within a TMN may be written in many languages. Access mechanisms to the platform facilities must be provided by the platform, together with pre-compilers and compilers.

Run-Time Libraries : The run-time libraries are those parts of the system linked together with the application code to form an atomic system component that may be added to that the system. The nature of the library may vary widely, depending on the mapping from the platform components onto the underlying host environment. The library may contain all the platform functionality, or may only contain those routines necessary to access the functionality of the platform and convert the 'native' data types and formats to the common format.

Type Management :The Type Management facilities have both an off line and on line role, and may handle two independent set of types within the system.

The first set of types is concerned with maintaning the common data types (e.g ASN.1) in multiple programming languages and a common interchange format.

The second set is the type or class lattice for computational objects within the system. These computational objects may have many roles - e.g. they may be part part of an application, an entry in an information base - and may have a multiplicity of implementations within those roles. These computational object types will be structered into a lattice to facilitate trading, biding, code re-use and polymorphism. These object types may be added, modified, versioned, or deleted.

Instance Management : The Instance Management facilities support the creation and deletion of object instances within the system. These objects are the realisation of the computational objects referred above.

3.2. Distributed Processing Support

The Distributed Processing Support components within the platform essentially provide abstraction form distribution of the applications supported by that platform.

Trading : The trader provides facilities for the identification and location of the system and application entities. The entities are identified by their properties. The trader also supports federation with other system traders or directories. The Trader functionality is similar to the X.500 Directory and the ODP Trader Function .

Binding : The Binding function provides a logical connection between two application entities taht wish to communicate via the invocation mechanisms provided by the platform.

Placement Management : It provides facilities for the initial placement of newly created object instances on machines within the system.

Transaction Management : It is concerned with transactions involving a number of potentially distributed entities. Transaction support is also provided at the lower level for those parts of a transaction within individual entities.

Invocation Management : It is concerned with ensuring that interactions between system and application entities conform to the common computation model supported by the platform.

Replication and Group Management : Replication Management is concerned with maintaning a set of instances of the same type in the same state. All the instances within the set may be visible outside the set, and invocations applied to the set are applied to each member of the set.

Group Management is concerned with maintaning a group of instances which appear as a single object to those objects outside the group. These instances may belong to different object classes and may be configured in a number of ways.

Dialogue Management : This provides a link between the platform and Human Computer Interface resources, and therefore the users, of the TMN.

3.3. The CP-Kernel

Communication Handler Objects : which present uniform interfaces to communication protocols, network data representations, and inter-process communication.
Processing Objects : which present uniform interfaces to the processing and scheduling capabilities of computer systems. Also included are objects that deal with intra-process concurrency (threads).
Storage Objects : which present uniform interfaces to file and database systems.
Device Diver Access Objects : which provides uniform interfaces to access devices on a system.

4. The IDEA Project

The MASI Laboratory of the University "Pierre et Marie Curie" (Paris VI), has worked a project called IDEA [3,4].

In this project we have proposed an architecture for network administration that takes into consideration the set of heterogeneity problems (manufacture's, architecture or data heterogeneity).

The purpose of the IDEA architecture is to provide, by a methodological approach, the design of a software environment for the development of management systems in a heterogeneous context. This environment is a set of application layer modules which can be implemented on top of several target hosts and which together form a run-time platform.

The adopted approach is based on handling the existing proprietary management systems, without modification, to provide the user with a unique view of their information. The IDEA framework is thus mainly determined by the different functionalities needed to realize the convergence of the manufactures' access points to their management applications (MAP) towards a user's unique access point (UAP). These facilities comprise: a) interfacing with the manufacturers' management systems; b) unifying management information into an integrated information base; c) management of local and remote operations and access control and user management.
To fulfill the above functionalities, the architecture defines a modular structure of co-operating functional blocks. It is the intention of the architecture that all the reference points between these blocks be identically specified. The goal of such generic interface is to allow recursivity for the sake of defining higher level functions (e.g. a meta-information base is defined as a recursion to the simple database and has the same interface specification) The main functional blocks are:
a) Manufacturer's Access Interface (MAI): provides an access to underlying system's resources through an Object-Oriented Model.

b) Integrated Network DataBase (INDB): provides an integrated data base with a unified interface to the shared network information (static and dynamic).
c) Local & Remote Operations (LRO): It controls the intra-IDEA interactions and provides an API (Application Programming Interface) which is used by an application to access the totality of the management system's information.
d) Interoperable Functional Block (IFB): provides the needed mechanisms to communicate extra-IDEA architectures i.e. between an IDEA architecture and its peers (whether not conformant to the given architecture).
e) Management Applications Entities (MAE): provides an object oriented approach to invoke common services. Applications are thus permitted to be developed by regrouping and building upon existing functions. A repository of functions implemented by applications and judged as public is kept in the platform.
f) User Access Control (UAC): provides the authentication and access rights control functions.

5. Describing the IDEA Platform Support Environment

The IDEA platform is a logical system resulting from the IDEA architecture. All the modules contained in the architecture has been mapped directly in the platform.

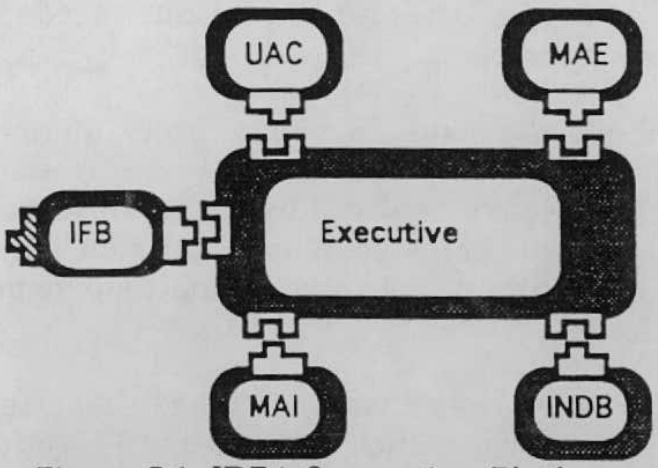

Figure 5.1: IDEA Supporting Platform

The IDEA Platform Support Environment is composed by two components: the send/receive interface and the Executive [5].

Send/Receive Interface : This interface permits the Executive to dialogue with all other components present in the network.

Executive : It is a distributed support platform prosed by IDEA project[]. The Executive possesses the following modules: Naming Server, Session Server, RPC Server and Kernel.

Naming Server : The Naming Server stores all the physical addresses that belong to each process that has manifested its intention to offer a service to the network.
Session Server : The Session Server possesses the actual state of all the messages running in the system.
RPC Server : It is responsible of the communication between Executives stored in different machines.
Kernel : The Kernel is the manager of the Executive and its main mechanism. Its task is to manage all the internal blocs, to dialogue directly with the native operating system as well as all applications that wish to use the network.

CP-Kernel : The IDEA project uses the UNIX as CP-Kernel.

5.1. The Send/Receive Interface

The API provides Send and Receive mechanisms for the two application development environments: C and C++. It is currently only intended to support remote operation primitives as described below. It does not support transactions (where a number of remote operation is grouped together so those if one operation fails, the whole group fails and the target is restored to its original state) and each message is assumed to have one, and only one target.

5.2. The Executive

The Executive provides the functionalities needed to implement a Distributed Processing Support.

Applications should not have to worry how information is stored. Transparency determines the extent that programmers need to be concerned with and have control over the distributed nature of the system. In a fully transparent system the application developer delegates all responsibility for distribution to the support environment provided (i. e Computing Platform).

The model used has been based in the client/server model. The Management Applications, clients, through the Executive Mechanism, use the service offered by other components. The Executive Mechanism has the task to locate the component searched for by the Management Applications and must permit that the communication between the two entities can be performed.

The Executive is responsible for the transfer of the information between the various engineering components, while hiding the underlying complexity of the communications network. Therefore, the design of the Executive must provide support for the following transparencies:

.Access transparency: is the property of hiding from a particular user the details of the access mechanism for a given object, including details of data representations and invocation mechanism.

. Location transparency: hides the exact location of a program component from any other.
. Failure transparency: Applications must know or be informed of failed interactions so that they can take the necessary actions. However it is advantageous to provide transparent failure recovery and provide limited feedback to the applications.
.Replication transparency: hides the effect of having multiple copies of program components.
.Migration transparency: hides the effect of a program component being moved from one location to another (not implemented).
.Concurrency transparency: the property of hiding from a particular user the existence of concurrency in the system. The concern here is to mask the details of synchronization and ordering mechanisms that ensure the distributed system can support multiple users while reaching in a consistent state (not implemented).
.Heterogeneity Insulation: The problem with the heterogeneity manifests itself when a set of networks is connected together to form an infrastructure for communications. These networks support a set of applications and are composed of components that have different design and technology. The Platform Support Environment must transform this heterogeneous environment into a consistent and uniform processing model.

5.2.1. The Naming Server
As discussed above, the distributed processing support must provide the localization transparency, as manner to abstract the network user of the physical localization of the server process.

The Naming Server is responsible for this task, storing all the server processes that are running in the machine. These server processes are called local' server processes. The Kernel asks always the Naming Server to know the physical address of a target application.

The role of the Server Name is to provide the correct destination for the Messages or Replies. This is achieved by mapping the context and target address of the message to its own directory service. Once the target server has been located, the Executive binds the client and server. This allows the transfer of the message across the platform and the return of a reply where appropriate.

5.2.2. The Session Server
The Session Server has as task to manage all the communication that circulates in the network. The message that arrives in Executive, through the Kernel, will be controlled and registered by the Session Server.

It is responsible to maintain a control about the messages that must return a response to users, signaling the Kernel if the responses have not been arrived in previous delays.

In fact the Session is reponsible of providing a reliable message service. The discussion above is attached to the problem of finding a timeout and

retransmission algorithm that takes into account the actual round-trip time that can be measured. To provide this task we are implementing to the Session Server an adaptative timeout with an exponential back-off function, known as Jacobson's algorithm.

5.2.3. The RPC Server

As discussed so far, we have treated the problems concerning local processes. It is possible that a client process asks for a service provided by a server process (called remote process) that resides in another machine.

The communication between the Executives has been done using the RPC paradigm. In this paradigm, the client applications can execute procedures on other networked computers or servers. In fact, we use a only procedure to send the messages that will be treated by the remote Executive. The services offers by each server applications are viewed as local procedures for each Executive.

5.2.4. The Kernel

The Kernel is the heart of the Executive system. It must control all the internal blocs and all the communications with the external world and native operating system.

The Kernel is responsible for managing the Executive, i. e, to initialize each process that corresponds to the internal blocs, the queues and the shared memories. Afterwards, the Executive can be accessed through the Kernel and the RPC Server.

6. Security Aspects in IDEA

In this section we present our first effort to securise the IDEA platform. As presented in the previous sections, IDEA is a distributed computing platform for the network management applications. Being distributed, it can not escape from the security threats of open systems and as a consequence it must be reinforced by security measures against the threats. In the following sections, we will see some of the threats to whom IDEA is particularly vulnerable and the integration of some security services to IDEA. The presentation will end by a short remark on further works and a conclusion.

6.1 Security Threats and Security Services

For the sake of security threat analysis, the IDEA model can be simplified as follows :

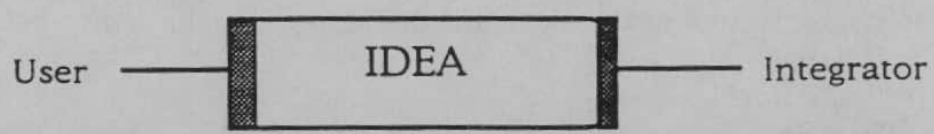

Figure 6.1

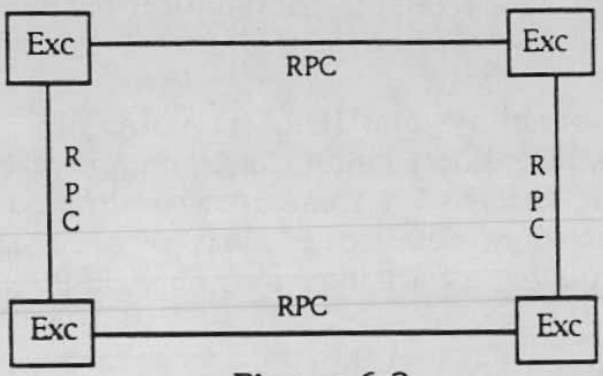

Figure 6.2

In figure 6.1, two interfaces are shown : the Visible Interface between the users and the IDEA and the "Common Service Interface" between IDEA and the Integrator. Figure 6.2 shows IDEA as a kind of network of Executives that communicates to each other based on the RPC paradigm. It is important to realise that the Executive nodes are not necessarily physically neighbours.

This simplified figures allow us to concentrate on the relevant components of the IDEA platform from the security point of view. In the present architecture, the communication between the users and IDEA and between the IDEA and the Integrator are local matters. As a matter of fact, the needed security services can be offered using the available operating system tools and/or within the application (IDEA) itself. The file protection mechanism offered by SunOS can be used to protect personal files against unauthorised access. This covers the illegal file modification threat and the file privacy violation threat. For instance, IDEA needs to protect the queues against unauthorised access. This can be achieved by creating a group (say IDEA-users) and giving the write access to the queues only to it's members. On the other hand, the usual authentication mechanism of the underlying (password) operating system can be effectively used to authenticate the IDEA users.

However the operating system doesn't cover all the threats or precisely, the resource protection model of the operating system is not sufficient for IDEA. For instance, the users' privileges with respect to the IDEA's resources can not be modeled using the operating system's protection model. This is partly because the protected resources (files, cpu, memories) in the operating system and the protected resources in IDEA (database entries,etc) are not the same objects. It is precisely for this reason that IDEA anticipates an integration of a module called User Access Control(UAC) in it's architecture. It's main function will be the authentication of users and verification of their access rights. In this paper,

we are concerned only on the security threats due to remote communication and hence the existence of the UAC module is assumed.

The only place where remote communication happened is between Executives (fig 6.2).

As any distributed system, IDEA is vulnerable to many security threats. Masquerade, illegal data modification, data privacy violation, denial of service, and traffic analysis are some of these threats. Interested readers can consult [13] for more detailed analysis of security threats in open systems. In this discussion, we are interested only on the following ones :

- Masquerade / Authentication Service

In any computer systems, principals (users, servers, etc) are identified with some sort of identification statements. Depending on the complexity of the system, the identity of principals can be a symbolic name, a number, a computer address or other statements. In general, a secret information is attached to the identification statements so that only the principal capable to supply this secret information together with the identification statement is identified by it. The masquerade threat is a threat where a principal may illegally takes the identity of an other principal for the purpose of malicious activities. The masquerading process can be as complex as the identification mechanism employed by the system. In networked systems where principals need to establish their identity with a remote system, the masquerade threat is critical as the principals' identification information traverses all along the network to the target system thus by increasing the entry points for the potential impostors.

We call an Authentication mechanism (or service), a mechanism by which a principal proves his identity to a system or an other principal. An authentication mechanism can be as trivial as the classical password insertion or as complicated as the Kerberos authentication mechanism [7] that uses cryptographic algorithms. The former is called Simple Authentication and is usually used in autonomous systems while the later is called Strong Authentication and is usually used in open systems [9].

In IDEA, the Executives are communicated remotely and in consequence Strong Authentication mechanism must be used.

- Data privacy violation /Confidentiality Service and Illegal data modification / Integrity Service

In open systems, the network is in general untrusted. This is a judicial assumption as networks are covering more and more geographically wide areas. In such situations, transit data can be illegally read or even modified in intermediate nodes. We call the former Data privacy violation while we call the later Illegal data modification threats.

Confidentiality service is a generic service that is used to counter the Data privacy violation threat. Most confidentiality services are based on cryptographic mechanisms. In this mechanism, the data is encrypted by the sender using a key pre-arranged with the receiver. That way, only the receiver who has the corresponding key is capable to recover the original data [13]

A combination of hash function and cryptographic mechanism can be used to detect whether data are modified or not in transit. The hash function is used to calculate the digest of the data to be sent while the cryptographic algorithm is used to encrypt the digest. If the data is modified in transit, it's digest would not be the same as the digest sent by the sender hence allowing the receiver to detect the modification. Integrity Service is a generic name for such services [13].

In IDEA, both these threats are present. For instance, the users' messages can be altered in transit. In some cases, confidential informations such as security management parameters can be read by unauthorised body. As a matter of fact, both the above services must be integrated in IDEA.

- The Problem of Key Management

Cryptographic algorithms are classified in two : Secret key algorithms and Public key algorithms. In the former, the same key is used both for encryption of the plaintext and for the decryption of the resulting cyphertext. In principle, the key is shared only between the communicating principals. Of course, the decrypting algorithm is the "inverse" of the encrypting algorithm. In the later case, the encrypting key is different but mathematically related to the decrypting key. One of the keys is private (not even shared) while the other is public (known by any body). In principle, it is mathematically infeasible to deduce one of the keys from the other. Data Encryption Standard (DES) is a cryptographic algorithm of the first category [3] while RSA (Rivest, Shamir, Adleman) is a public-key algorithm [11].

In the case of security services that uses secret key algorithm, a key must be shared between the communicating principals before the proper communication. If there are N potentially communicating principals in a system, it necessitates each of them to keep N-1 keys, one for each other principal. It is not hard to see that this does not work for large systems. Beside, letting each principal (specially human users) keep and use these much secret keys is not a good practice.

One common solution is to introduce a trusted component in the system. This component, we call it Authentication Server, will exchange a secret key with each of the principal in the system. In this arrangement, all a principal has to do is to keep secret the only key he shared with the Authenticating Server. The main function of the Authentication Server is the generation and distribution of short-timed keys, called session keys, among the communicating principals. The session keys are distributed

securely thanks to the shared secret keys. Once issued, a session key can be used many times until its lifetime does not expire. It is not within the scope of this paper to give a detailed presentation on this method. Interested readers can consult [6,7]. At last, we will like to mention that one of the disadvantages of this solution is the introduction of a critical component in the system. It goes with out saying that, the Authentication Server must run in physically secured room. Kerberos uses a variety of this method for authentication and key distribution.

In the public-key mechanism case, the problem is some what different. Here, each principal owns two keys; one private and the other public. For secure communication, all one needs to do is to learn the public key of his communication partner. Again, it is not wise for each and every one to keep the public keys of the potentially large number of partners. One can imagine the huge management work this would entail whenever a partner changes his public key (well, the operation is not even acceptable from a security point of view). One solution is to use a generic directory service a la X500 for the distribution of public keys. However, this necessitates as well two things : the directory must be trusted (as the Authentication Server above) and a secured communication is needed between the principals and the directory server. In particular, the Authentication service and the Integrity service must be used. This is a right solution as long as the scope of the system is limited geographically. In a more general case, the problem is more complicated as we will be forced to use a generic Directory Service (hence not necessarily trusted). This problem is treated in [6,14].

In IDEA, all secured communication is done between Executives (servers). As a matter of fact, they constitute the principals of the system. Cryptographic keys will then be associated with them.

6.2. Secured Idea

This section treats the implementation aspect of the ideas discussed in the previous sections. We recall that the security services we need to integrate are the Authentication service, the Confidentiality service and the Integrity service.

IDEA is implemented in SunOS environment [12]. In particular, Sun RPC is used for the communication of Executives. Since RPC is the only paradigm used for the communication between Executives, the integration of security services in it will be enough to realise the needed security level. SUN provides a secured version of it's RPC package that has integrated the Authentication and Confidentiality services.

The Sun Secure RPC can be divided into two components : the key management and the proper Secure RPC. In principle, one can use a different key management mechanism on the top of the Secure RPC.

The Secure RPC is based on the DES algorithm. As we mentioned above, this algorithm uses the same key both for encryption and decryption. Given that the two communicating principals are agreed on a secret shared key, the Secure RPC transmits their data securely (in encrypted form). By the mere fact that the shared key is known only to the two principals, this process assures the authenticity of the sender as well as the confidentiality of the data.

In Sun Secure RPC, each principal owns a private key and a public key. Both these keys are stored in the NIS (Network Information Service) database; the public key is stored in plaintext while the private key is stored encrypted by the principal's UNIX password. When a principal logs into his system, his record (his network name, public-key, encrypted private-key) is retrieved from the NIS database. Then his password is used to recover the private key; this key is kept in the memory of a special process called keyserver (RPC Version 4.1).

Sun RPC uses a method called exponential key exchange to let two communicating principals arrive at the same session key, without ever broadcasting that key on the network. For this to be achieved, each of them need to combine it's private key with the other's public key. Once a session key is obtained, a client (the principal that initiates a conversation) generates a random conversation key and sends it to the server, encrypted with the session key. The Secure RPC uses this key to encrypt the transmitting data.

To summarize, we observe two separate steps : the Key Initialization step and the Secured Communication step. The first step is needed to retrieve the public and the encrypted private keys of a principal from the NIS database. Sun provides mechanisms to retrieve these keys. For the sake of security, the private key is stored in the memory of the keyserver process, after having been decrypted by the principal's UNIX password. The second step is needed when a principal (client) wants to establish a secured communication with an other principal (server). In this step, the client retrieves the server's public key from the NIS, calculates the session key, and generates a random conversation key. Then, he sends the conversation key to the server encrypted by the session key, where after, the server retrieves the client's public key and calculates a session key. If the server arrives at the same session key as the client, he is able to recover the conversation key. At last, this key will be used for the proper secured communication between the client and the server.

Now that we have briefly exposed how Secure RPC works, it's use in IDEA is almost straight forward.

The first thing to do is to use the Secure RPC version instead of the originial RPC. Next the IDEA program code is to be modified so that it incorporates the calls related to the exponential key exchange method. Finally, the (Sun) key management infrastructure must be integrated to the system. Although there is a number of options for real implementation of

IDEA in UNIX, we assume that the Executive, as a server, is started as part of the bootstrap of the machine. In this assumption, the Executive is owned by the superuser and hence it uses the superuser's keys. The Key Initialization step is done by the superuser independent of the Executive server. From the Executive's point of view, it's private key is assumed to be kept by the keyserver before it initiates a secured communication.

As mentioned above, Sun Secure RPC provides the Authentication and Confidentiality service but not the Integrity service. However, this last service can be integrated with out much difficulty.

6.3 Remarks

The Sun Secure RPC is chosen to securise IDEA mainly because IDEA is developed in SunOS environment. Obviously, this will save a lot of working hours. However, this choice is not without it's price. This is specially true in the key management infrastructure. In Sun Secure RPC, the keys are associated not with servers but with UNIX users. The association with the servers is done indirectly by allowing them to use the keys of their owners. This creates two problems : first, keys are valid as long as their owners remain logged. But, in general, servers last longer than the users who started them. Second, a single user may own a number of servers (eg. the superuser) which implies that different servers use the same key at the same time. This is not acceptable from the security point of view. We believe that a different key management that takes into account these issues must be used. (In Kerberos, a single key is assigned to each server independent of it's owner).

7. Correspondence of IDEA Platform Support Environment to the General Platform Support Environment

Mapping to the CP Kernel

The Communication Handler: The IDEA CP platform supports communication over System V Messages Queues and TCP/IP networks.
Threads: They will provide in the next version. In this version, the system does not use the Threads.
Storage Handler: Which provides uniforms interfaces to file and databases. This function is provided by the INDB.
Device Drivers: It provides uniform interfaces to access devices on a system.

Mapping to the Distributed Processing Support Layer

Trader: It is provided by the Naming Server.
Binder: It is provided by the Kernel of the Executive.
Invocation Manager: is concerned with ensuring that interactions between system and application entities conform to the common computation model supported by the platform (provide by the Kernel and Session Server)

Mapping to the CPI Layer

Run-Time Library: The messages are exchanged using a Run-Time Library, that contains the following components: The send/receive interface and the parser.

Pre-compilers: It will be provided by the UAC component. The Interface Description Language needs to be defined. They will produce C code which is compiled with the appropriate libraries.

8. Conclusion

We have proposed in the CPSIG-RACE project a general architecture for TMN applications. PRISM laboratory has developed a platform called IDEA.

We are trying to securise the IDEA platform following the points discussed above. The first and theoretical study is completed. This will shortly be followed by the real practical work (hoping that we will have the exportable version of Secure RPC).

With a suitable key management infrastructure, Sun Secure RPC is a right solution for the IDEA project. We believe that this, together with the User Access Control module, will give an acceptable security level for IDEA.

Finnaly, we try to show how the IDEA platform can be mapped in the CPSIG-RACE platform.

Acknowlegments

The authors gratefully acknowledge Roke Manor Research Ltd (UK), Britsh Telecomm (UK), Trinity College (IE) and Broadcomm(IE) for their contributions. The first author has been supported by Banco do Nordeste do Brasil (BNB) and CAPES.

References

[1] Wade, V.et all.: "A Framework for TMN Computing Platforms", proceedings of the 5th RACE TMN Conference, London Nov. 1991.

[2] Wade, V., Donnely, W., Roberts, S., Harness, D., Riley, K., Celestino, J., Shomaly, R.: " Experience Designing TMN Computing Platforms for Contrasting TMN Management Applications", published in "The Management of Telecommunications Networks", Ellis Horwood Limited, 1992, ISDN 0-13-015942-5.

[3] Claudé, J-P. et al.:" IDEA : A method for the unification of heterogeneous network management", International Telegraphic Congress ITC, June 1991, Copenhagen.

[4] Claudé, J-P., "Proposition d'une spécification d'administration de réseaux hétérogènes", Thèse d'état, Université Pierre et Marie CURIE, 24 Sept 1987

[5]Celestino, J. - "A Distributed Processing Support for an Integrated Network Management, IFIP Conference'93, WG 6.4, April 1993, France.

[6] W. Lu, M.K. Sundareshan : "Secure Communication in Internet Environments : A Hierarhical Key Management Scheme for End-to-End Encryption", IEEE Trans. On Communication, Vol 37, No 10, Oct 1989.

[7] S.P. Millen, C. Norman, J.I. Schiller, and J.H. Saltzer : "Kerberos Authentication and Authorization System", Project Athena Technical Plan, Section E.2.1, MIT, July 1987.

[8] National Bureau of Standards "Data Encryption Standard", Federal Information Processing Standards Publication 46, Washington D.C, 1977.

[9] R.M. Needham and M.D. Schroeder "Using Encryption for Authentication in Large Networks of Computers" Comm ACM, Vol 21 No 12, Dec 1978.

[10] RACE Project R1009:"An Implementation Architecture for the Telecommunications Management Networks", Mar. 1991.

[11] R.L. Rivest , A. Shamir, L. Adleman : "A Methos for Obtaining Digital Signature and Public key Crypto-systems" Comm ACM, Vol 21, No 2, Feb 1978.

[12] Sun Inc : "Sun Operating System", Version 4

[13] M. Satyanarayanan : "Integrating Security in a Large Distributed System"
ACM Transactions on Computer Systems, Vol 7, No 3, August 1989.

[14] A. Tarah, C. Huitema : "Certification and Routing Protocols", IPPS 92