PKIX BASED CERTIFICATION INFRASTRUCTURE IMPLEMENTATION ADAPTED TO NON PERSONAL END ENTITIES

Eduardo Jacob, Fidel Liberal, Juanjo Unzilla
E-mail:, jtpjatae@bi.ehu.es, jtblimaf@aintel.bi.ehu.es, jtpungaj@bi.ehu.es

Department of Electronics and Telecommunications
University of the Basque Country
Alda. Urquijo s/n . 48013 – Bilbao Spain
Telephone: +34 94 601 42 14. Fax: + 34 94 601 42 59

Abstract

Public Key Infrastructures are considered the most suitable system to provide basic security services through digital certificates use. Nevertheless, traditional way of operation, based on web interface and asynchronous interactions, as well as the cost and difficulty of registration processes, have caused their replacement for another systems in many of the scenarios PKIs were conceived for. These more efficient solutions present generally laxer security mechanisms and require too much user knowledge. The system we propose tries to provide a bridge between both approaches by defining an automated PKI focused on specific application scopes by using on-line interaction procedures.

Keywords: PKI, Certificate, CA, RA, PKIX, cryptlib©, Public Key, LDAP, X.509, CRL.

1 Introduction

The definition of Public Key Infrastructures (PKIs) was driven by the need for setting a trust agent that should grant basic security services (integrity, condifentiality, authentication and non repudiation). This finally drew the solution into a combination of hardware, software, policies and procedures that provide mechanisms to issue, distribute, validate and revocate digital certificates.

We have found that there are several security frameworks for providing authentication and architectures as Kerberos or intermediate responses to this problem as SSH or PGP that could be sometimes replaced advantageously with a PKI based solution.

Nevertheless, traditional way of operation, based on web interface and asynchronous interactions, as well as the cost and difficulty of registration processes, has caused many times to begin searching another technologies. This has happened sometimes to ease the use of the application, or simply because there was no human to interact with.

In this article we will explain a PKI system we have developed that is oriented to provide advanced certification services. This work tries to be compliant with protocols defined in PKIX **[1]** standards.

The system architecture is composed of at least one RA, which is responsible for processing user requests and a CA that generates certificates, revocation lists and maintains a LDAP repository.

One very important point in our work is that these RAs, after performing user authentication and validation, will mark users' requests as valid and forward them to the CA, which will process them automatically, with no human agent intervention. The rationale behind this is to try to automate and speed the generation, transport and installation of the certificates in applications.

The system will also provide validation services through OCSP **[2]** protocol use. The RA will again have a proxy-capability, acting as an authorized OCSP responder. It will ask CA only when the requested information is not available locally. This schema tries to assure proper scalability, both in equipment and in geographic position, by spreading out more and more subsidiary RAs.

In addition to providing on-line validation methods, our CA will periodically issue appropriate CRLs that will be stored in the repository (a LDAP directory service) with issued certificates.

As it has been already mentioned, in contrast to traditional architectures, typically oriented to provide final services to human users (generally through a web interface) our PKI is focused on easy integration of applications or services that use the certification infrastructure. With this purpose in mind, the developed software layer will make interaction procedures easy to programmers by adapting the cryptographic library and hiding implementation details.

The software we have developed consists of RA and CA daemons, clients for RA and CA operators and administrators and a software layer to help the integration of new applications through an API. These have been written in C language with heavy use of threads for i386 Linux systems. The cryptographic support has been provided by Peter Gutmann's cryptlib© **[3]** library.

2 Background

Many times typical PKI systems land in serious security implications when we arrive to the procedures involved in many operation, specially in user registration and security of private information (PSE manipulation). This is due to the fact that they involve interaction with final users that may not be aware of the security implications and mechanics.

Traditionally, the solution to this problem has been to try to provide the users with necessary, convenient and/or frightening information to help them progress securely every step. As an example, think in the process of starting a SSL session with a site whose certificate (although perfectly valid) is not verifiable by the browser. This approach tries to actively involve the end user in security procedures and decisions. This is error prone and not by any means the more efficient way to do it.

This philosophy results on trying to provide an interface as friendly and complete as possible. Generally, that purpose is achieved through a web interface that gives requester guidance throughout the whole process (during registration, revocation…), while informing the user about the meaning of every field and the consequences of each action involved. A solution to this problem is partially addressed by "managed" certificates solutions offered by commercial all-in-one PKI vendors as Baltimore.

Using a web interfaces disallows in practice automatic operation, which doesn't automatically result on any problem, because certain human interaction has been always thought to be necessary. Furthermore, registration process implies in most cases the intervention of, at least, one human operator (CA operator).

Another aspect to take into account in classical web-based PKI approaches is that most operations are asynchronous. So, both in registration /initialisation and revocation procedures, user sends a codified request to the system (CA or RA). This request is not immediately processed but, after some time, a human agent (typically RA or CA operators) checks whether request is valid and sender is an authorized user. After this check, the request is processed and the result obtained is again asynchronously sent to user who is notified in order to pick it up. This process becomes even more difficult if checking requires requester's presence at RA to provide required administrative data to ensure user's identity.

Although this way of operation tries to guarantee registration and revocation operations, it results on a lack of agility in the service, which is specially derived from identity checking being made after request is received. All this also implies system being prone to denial of service attacks or just overloads, because it has to store all requests, as it isn't able to distinguish between correct requests and incorrect ones prior to final checking.

On the other hand, the great number of certificate extensions that can be included in a certificate makes necessary to offer some guidance to the user when requesting them to get congruent results. Depending on users needs some extensions should be used (and correctly filled) and others not. To ease this operation the user is offered several fixed pre-recorded sets of extensions some of them also pre-filled. Some classical certificates are the SSL server, S/MIME user certificate and a limited amount of additional ones. In these alternative cases it's necessary to ask users for required additional fields for each type of certificate.

The task of adding a new service becomes quite complex because of having to modify both processing and presentation aspects: web interface's appearance and CGIs, embedded code (ASP, PHP, etc …)

All these troubles are associated to the lack of requests automation mechanisms, which causes that interface has to fit every new type of certificate in order to guide user throughout the process of filling in fields.

Using this end user-oriented interface and "manual" way of operation (requiring human intervention) disallows also effective interoperability between different systems, reducing this interoperability to cross-certificates issue.

Also, the lack of a mechanism to automate interactions with PKI means that those interactions are completely segregated from the services that aim to use certificates. It must be admitted that recently the access to certificates stores by applications and services in recent versions of Windows 9X/ME/2000 operating systems and Internet suites for them has been facilitated. Nevertheless, the need for creating, importing and exporting certificates by hand persists in many applications, mostly in proprietary ones and in other operating system environments.

All these procedures have caused in practice that certificate management operations turn out to be quite obtrusive and, in most cases, slow and expensive. As a result of this and despite PKIs and certificates are considered as the more suitable foundation for providing basic security services, other systems have been preferred. These systems usually present laxer security procedures and require users' awareness about implied technologies (such as SSH and PGP).

This situation results quite paradoxical, because it causes that PKI are being replaced, in many environments they were originally conceived for, by less valid solutions whose security risks are quite bigger for users, specially for new users. As a result of this, their use has been restricted to secure web servers at middle-size companies and very rarely for S/MIME mail-client certificates, although they should be used in a wider range of applications (i.e. an alternative to SSH).

As it will be shown later, our design implies the automation of some tasks and operations, which breaks up with the way a PKI is normally supposed to run, specially in those aspects related to CA way of operation and keys protection. The decisions we have taken in our design are headed to create a system that should be able to provide agile certificate issuing and managing procedures in certain environments. This also means that our design is not designed to substitute public heavyweight PKI infrastructures.

For example we can think of an ISP providing certification services to client companies that often exchange a great amount of data. In this case the ISP acts as the central trust authority or hub. Once proper legal responsibilities derived from their use of the system are established and accepted by all parts, it would provide an efficient mechanism to provide certification services, to reduce costs and speed up involved procedures. An intensive use of the proposed PKI to centralize certification services and to provide a security framework could also be considered. Then, the system will allow the development of new applications over mentioned framework, taking advantages of its high security level and ease of use, so that traditional systems such as SSH, PGP, non-authenticated SSL connections (using dummy certificates) … would be replaced.

Many of the procedures that have been developed during this project are derived from the proposals that PKIX Working Group has made while trying to solve the troubles related with the automation of interactions. This Working Group was created by IETF in 1995 and it's focused on the development of Internet standards needed to provide a PKI based on X.509. Nowadays it goes on elaborating drafts and RFCs collecting the protocol definitions mostly in 5 areas related to:

- Definition of the profiles of X.509 v3 PKC and X.509 v2 CRL [5] standards for the Internet.
- Operational protocols to be used to distribute and obtain information from certificates and CRLs.
- Management protocols to allow on-line interactions between the entities that form the PKI.
- Certificate policies and certificate practice statements (CPS) definition and settlement.
- Time stamping and data certification and validation services.

PKIX workgroup defines a PKI structure that isn't purely hierarchical but uses intermediate links through cross-certifications. One of its main purposes is also grating the interoperability between different implementations.

But maybe its most important contribution consists in the definition of the protocols that manage the interactions with the PKI, and particularly to the fact that these management operations can be carried out over different bearers, such on-line TCP connections. In order to do so, PKIX defines operations and messages, for example for initialisation/updating tasks (CMP/CRMF protocol **[4]**) and for certificate validation through OCSP protocol.

Therefore, the dependency between the interface and processing in interactions is finally removed, because it's the same mechanism for every type of certificates. Besides, it allows the development of applications that use certification services, avoiding the need of web access in order to deal with requests.

PKIX defines in a very open way the specifications of a PKI, it limits itself to propose a general architecture model close to the classical one composed by a CA, with one or more optional RAs and a repository. Referring to operations to carry out, PKIX specifies some mandatory schemes that have to be followed during registration procedures and additional optional mechanisms.

Nevertheless, it only defines external interfaces and, as it considers the use of RAs convenient but optional, it don't specify how responsibilities are divided between these ones and the CA, neither which protocol or message format must be used in their communication.

As standards don't specify internal operations in each element but interrelations with external agents, in every implementations it must be decided which tasks and operations should be included, as well as every agent's responsibility.

For this development we decided that the attributions of subsidiary RAs were equivalent in practice to those of CA, except in the aspects related to final operation over certificates. So, CA will automatically accept any request sent by a recognized RA, and will proceed to its execution.

For internal communications between different elements we will use the same protocols as in the external interfaces, so that RAs will act as "special clients" toward CA.

Requests sent by RA will be specially secured by digitally firming their content with RA's certificate, instead of user/password schema used by normal end-entities.

3 Design

The general architecture of the system consists of using of one or more subsidiary RA around a central CA, and a directory system, responsible for CRLs and issued certification storing.

3.1. General architecture and operation

To decentralize the task of registration (and to help and reduce the cost of identity checking during this process) some RAs are articulated around of a CA, which will be only in charge of automatically certifying the requests sent by RAs in a proxy agent way. So, in the CA all identity verification operations are finally reduce to check whether received requests have been generated by a trusted RA. That really means delegating identity checking to the RA. This schema would also allow company that runs the PKI to use RAs that were managed by another companies.

A typical example of previous case would again consist on a ISP that provides certification services to big companies that require a great volume of certificates. These companies would find it interesting to have their own subsidiary RA, so that they could save costs of registration procedures and at the same time keep their employees' administrative data secret, since registration processes is done within the company itself.

Therefore, CA's role will be limited to ensuring its private key's integrity, which the whole system is supported over, and to executing final certificate-issuing operations, by automatically accepting requests from RAs.

Obviously, certain condition must be matched so that previous model could be valid: First of all, responsibilities of every agent (both CA and RA) must be explicitly set and then which requisites must be demanded by subsidiary RA for user authentication. The trust that is put in RA is also granted by the fact that any bug or fraud attempt would only have effects on those certificates which were handled by the RA itself and so, that belonged to the company that held the RA.

Regardless of whether they are managed by the same company that runs the CA or another one, RAs are responsible for fetch users' requests so that they form the external interface for the system. Registration procedure will also take place in RAs, both initial identity checking and later certification request.

With the purpose of speed up and make easy registration process for users in mind, our design provide the following mechanism for users initialisations:

First, requesting user or entity gets in contact with the closest RA in order to carry out the only human interaction that will be required during the whole process. The rest of the operations to be performed in order to issue certificates will be completely automated and won't need any human operation's intervention.

The initialisation process works as follows:

Once the RA operator has checked requester's identity following security policy described in CPS, he would fill in all fields of the "future" certificate with user's data and store them in an entry of the DB in RA. During this process the RA server would create an ID and a password that the operator would give to user out-of-band.

These data are the only thing the client needs in order to later on send a certification request (using CMP/CRMF protocol) to the RA. The RA server, after having checked user's ID/password, will load from the database a previously generated pre-request with all fields already filled in. Then, public key will be extracted from user's request and incorporated into pre-formatted one. Finally the newly created request will be forwarded to CA, which will be responsible for issuing the certificate, bringing it back to RA and publishing it in the repository.

This schema frees final users from any question related to which fields should present the request, which information should be included in DN and other certificate format problems. While web interface in traditional approaches imposed critical restrictions while providing different certificate types, in proposed system requesting one type of certificate or another is completely equivalent. So, registration process is therefore easier than in traditional approaches since user only needs to know ID/password to get into the system.

User must be able to choose between one certificate type and another, as well as to fill in those fields that are needed in each case. In our system, the RA operator carries out all these operations during the registration process.

To achieve this, a table with certificates types is stored in RA's database. In this table there are two columns for each field/extension described by X.509 standard. In the first column a value codifies whether corresponding field/extension is optional or mandatory and whether it must be filled in by RA operator or automatically by RA itself. The second column contains the value to be filled in automatically by RA (i.e. in keyusage extension, Netscape and Microsoft proprietary extension) that will be fixed for a certain type of certificates.

So, during the registration procedure, once RA server knows which certificate type is needed, it will read from the database all possible fields/extension associated with this certificate type. If the RA operator must fill in certain fields, server will ask him/her for that information. If it must be filled in automatically by RA server, proper value will be read from DB and incorporated to the pre-request. Once this process is finished pre-request will be stored in the DB.

With this architecture, adding a new certificate type issuing capability to a system is as easy as adding a new entry to certificate types table in DB, with all suitable fields and extensions and their values. So, proposed system is able to provide detailed and complex certification services easily, since only RA operator has to know each field's meaning and which values could it take.

The rest of operations over certificates in which PKI is involved are much easier than registration, since users already have the certificate that authenticates them against the system, so that revocation and actualisation procedures are carried out automatically.

PKI also provides validation services, once again through RAs acting as authorized OCSP responders.

The way of operation followed by RAs in validation services looks like the way DNS servers reply to request in domains where they are not authorities. So, EEs will ask already-known RA for information about certain certificate.

The RA will search through its memory-cache for requested information. If the information is found and it is not expired (a configurable lifetime value exists), the RA would return associated state (valid or not) to the requester. This answer would include elapsed time since certificate was really checked. If any of these two previous conditions are not matched, RA server will make a query to internal DB and, in last instance to CA. Since CA has information continuously updated it always gives reliable answers. In both previous cases memory-cache system will be updated with the obtained information.

As we have already mentioned, an adaptation layer is provided, in order to help developers to add these services to their application hiding the complexity of many operations. It consists of a programming API developed over the cryptlib© [3] crypto library.

3.2. Implementation

As already is said, proposed architecture involves the implementation of a CA, at least one RA, the repository and an API.

The RA and CA servers are daemons written in C language on i386 Linux systems.

The cryptographic library that provides management protocols defined by PKIX is cryptlib©. The use of this library instead of an Open Source one (like OpenSSL) is justified by the fact that it includes many of the procedures that a PKIX based PKI requires. Although it is not supported by a big user community, it is an active project since 1996. The library itself offers a robust environment designed around a security kernel based architecture. In this way every function is checked for the needed attributes before any action is taken. Also, the API is widely documented, including a 300-page manual and a great amount of sample programs.

The general structure of **irad** is shown in *Figure 1*.

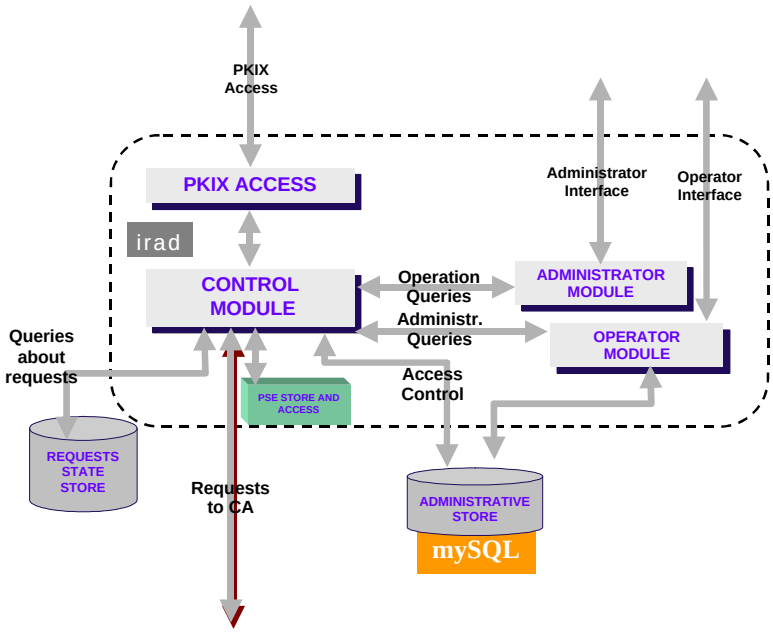

Figure 1. Block diagram for **irad** server

With regard to the servers (**irad** and **icad**), their way of operation is very similar: after having read their respective configuration files each server will unfold into a series of threads responsible for CMP and OCSP request attending. They are also responsible for management tasks.

Both RA and CA daemons are multithreaded servers that adapt to requests load. To achieve this, a series of CMP and OCSP serving threads are pre-launched, so that a minimum number of threads is always maintained. These threads are in "sleep" or "idle" mode, awaiting connections. As more connections arrive new threads will be generated on demand, to enhance requests attending. Those newly created threads would disappear automatically after certain inactivity period provided that new request didn't arrive. All this mechanism tries to improve response times while trying not to waste memory (by thread self-destruction).

Both servers offer two interfaces to interact with two different types of users or "roles":

- The first role, the Operator, is responsible for operations related to users and certificates, without too much technical knowledge about the technical internal running of the server.

- The second role, the Administrator, has to ensure that server is working properly but can't either register users in the system nor access any information stored in DB. In fact,. RA/CA Administrators can only access those variables that affect internal server operation.

The servers don't provide these interfaces through console, but through two SSL servers that listen in certain ports and interact with the users using a simple command/response protocol.

This approach allows that any possible software client could use whatever type of front-end over SSL connection based protocol, varying in a wide range from text clients to graphical applications.

Firstly a simple text based client was implemented for testing and debugging purposes. A Java-based graphical client was developed later to make Operator's work much easier. Providing a friendly interface to the Operators is almost mandatory, since these employees tend to not having technical skills.

The DBMS we have chosen to store certificates, requests and administrative information is MySQL.

Cryptlib© library already provides functions to run a simple CA, including a certificate store and capabilities to launch CMP/OCSP listening processes. However, since it only comprises an unique CA, the whole requests attending mechanism almost has had to be rewritten to bypass library's security constraints and forward requests to CA. Even more library hacking was needed while implementing pre-requests creating and reloading functions.

Adding prefabricated requests on demand also meant adding new data structures and tables to the database trying to be compatible with the way cryptlib© accessed DB.

The internal design of the Certification Authority (*Figure 2*) is almost the same, but it requires even less human intervention, since CA Operator only takes part in RAs' registration process and in exceptional manual certificate revocations.

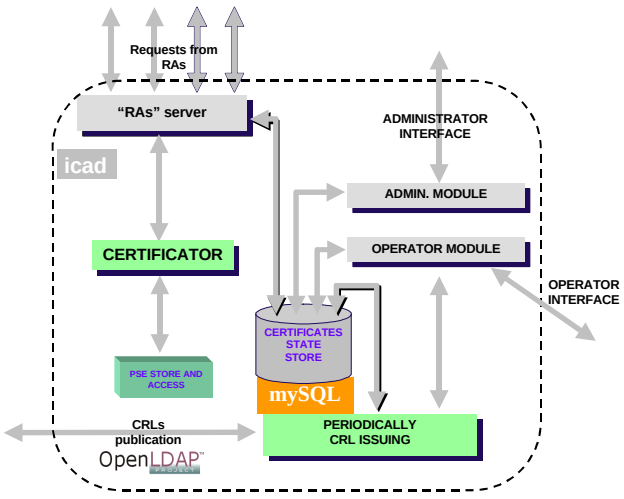

Figure 2. **icad** server's block diagram

Repository system consists of a LDAP server based on an OpenLDAP distribution. CA will store issued certificates and CRLs in this repository, so that traditional operation way is somehow maintained, which also makes migration easier.

Finally, adaptation layer consists of a series of software modules within a static and dynamic library, which structure is shown in *Figure 3*. This library offers an API to application developers to design applications that use this infrastructure. It also offers access to the actual software based user's PSE.

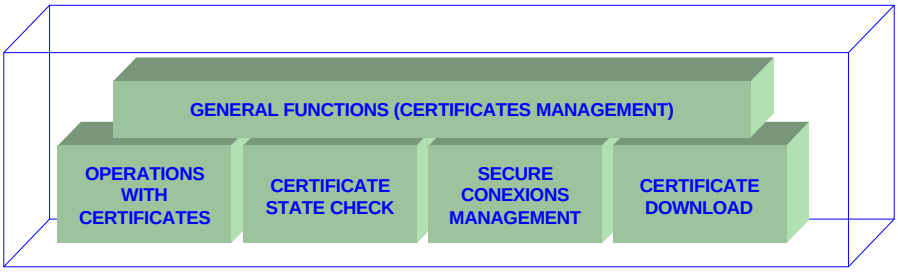

Figure 3. Adaptation lawyer's block diagram

At this time, the project is about 10.000 lines of code, including the crypto library hacking, totally developed by Fidel Liberal Malaina.

4 Conclusions

Our project tries to go one step further on certificate service providing through automation of most interaction procedures with 3 main purposes:

- Speed up all procedures and reduce operations that needed human operators.
- Make certificate services more flexible, as it provides an unique access mechanism, so that adding new types of certificates is much easier.
- Ease as much as possible users' access but maintaining services complexity and power.

These objectives are achieved using protocols proposed by PKIX standards. Is also important to note that the responsibility for filling requested certificates fields falls on the RA Operator, while the EE later, only has to know given ID/Password in order to get or manage the certificate.

The system is primarily focused on providing application certification services. This is the reason for the inclusion of a software layer, instead of a graphical user-oriented interface. Besides, its applications scope doesn't originally fit PKI public environments, but particular environments that can take advantages from system flexibility and speed in order to use intensive certification services.

In terms of implementation, it was mainly focused on speeding up certificate management operations, through load-adapting multithreaded servers use and making administration interfaces more flexible.

The use of OCSP to provide validation services allows on line certificate state checking, a mechanism much more reliable than traditional one of CRLs.

5 Future lines of work.

At the present state, we can group the future lines in two main directions. In the first one we can propose the development of the PKI itself:

- Exhaustive interoperation tests against commercial tools that follow PKIX standards.
- Adapt PSE access modules to hardware devices, such as smartcards, crypto-tokens…

- Integration with other certifications systems like PGP.
- Inclusion of attribute certificates.
- Development of Window family client libraries.

In the second, we can deal with the design and deployment of services that use this PKI.

- Integration of certificate services in cooperative e-learning tools used in distant education.
- Integration of certificate services in network management tools.

6 Acknowledgements

This work has been developed in the Department of Electronics and Telecommunications of the Faculty of Engineering of Bilbao (University of the Basque Country) and partially funded with a Research Project granted by the Department of Industry of the Basque Country Government and Sarenet S.A. an Internet Service Provider.

7 References

[1] IETF PKIX WORK-GROUP, http://ww.ietf.org/html.charters/pkix-charter.html.

[2] IETF, *RFC2560* Internet X.509 Public Key Infrastructure. Online Certificate Status Protocol – OCSP. *June 1999.*

[3] cryptlib© Library. http://www.cryptlib.com

[4] IETF, *RFC2510* Internet X.509 Public Key Infraestructure. Certificate Management Protocols. *March 1999.*

[5] IETF, *RFC2459* Internet X.509 Public Key Infrastructure. Certificate and CRL Profile. *January 1999.*

8 Vitae

Dr. Eduardo Jacob Taquet (1963) works as lecturer in the Department of Electronics and Telecommunications of the University of the Basque Country at the School of Engineering of Bilbao where he teaches Operating Systems and it's associated laboratory and Mobile Services and Networks. His main interests are Distributed Systems Security where he has directed several R&D publicly funded projects, and IP/GSM/GPRS mobility. He is co-founder of the GIT (Grupo de Ingeniería Telemática) that is a Group for Research in Telematics of the University of the Basque Country which is currently involved in several projects related to Internet Security, PKI and Intrusion Detection.

Fidel Liberal Malaina (1978) is a Doctoral Candidate in Information Technologies in the Electronics and Telecommunications Department at the School of Engineering of Bilbao. He received his BS degree and later his MS degree in Telecommunications Engineering in 2001 from the School of Engineering of Bilbao, University of the Basque Country where he currently cooperates in R&D projects. His research interests include Electronic Commerce, Security (PKI) and Internet applications, specially related to mobile environments.

Dr. Juanjo Unzilla Galán (1966) received his BS and MS degrees in Electrical Engineering from The University of the Basque Country (UPV/EHU), Spain, in 1990 and Ph.D. degree from the same University in 1999. He works as lecturer in the Electronics and Telecommunications Department at the School of Engineering of Bilbao, where he teaches Telecommunication Services and Networks and Open Systems Interconnection. He is also co-founder of the GIT (Grupo de Ingeniería Telemática), a Group for Research in Telematics of the University of the Basque Country. His research interests include computer networks, Electronic Commerce, Security and Internet applications.