SDR REFERENCE SECURE ARCHITECTURE

Rafael Aguado Muñoz (Indra, Aranjuez, Madrid, Spain; ramunoz@indra.es)
Raúl Dopico López (Indra, Aranjuez, Spain; rdopico@indra.es)
Javier Fernandez Alonso (Indra, Aranjuez, Madrid, Spain; jfalonso@indra.es)
Pedro Aurelio Llamas (Indra, Aranjuez, Madrid, Spain; pallamas@indra.es)

ABSTRACT

"The good thing about standards is that there are too many to choose from"

This is not new. The quote appeared as a signature in one email too many years ago previous, even, to the existence of SCA. This affirmation becomes more truthful as you go deep into security topics, where historically, and for many reasons easily guessed, every detail has been hidden.

This situation has caused security to be discussed always as a black box, preventing the waveforms of being portable and the platforms being reusable. This paper will present some considerations on the topic, developing a candidate secure architecture, using a different set of well known state-of-the-art standards:

- **SCA Security Supplement.** Although not updated in its last version JTRS SCA v2.2.2, the security supplement is still considered the most important reference in the security definition of a SDR based equipment.
- **CICM.** MITRE cryptographic driver is increasing its importance as standard in order to offer functionality from the cryptographic side. One of the most valuable features of CICM, to be selected for this implementation, was its capacity to support high-assurance cryptographic modules in critical settings, as well as commercial modules.
- **CORBA.** CORBA connectivity mechanisms will be addressed in order to provide transparent communications between Red and Black subsystems. The Extensible Transport Framework will be implemented to customize the transport used between components allowing such communications.

The following chapters will present the integration needed among all these different areas in order to achieve the goal of having a SDR compliant cryptographic subsystem. The lessons learned during the implementation of the final solution will be presented throughout the different chapters. Lastly, the problems appeared during the last phase of the implementation and the solutions found will be presented as improvements for later consideration.

1. INTRODUCTION

The definition of a standardization process for security subsystems is far more complex, than in other SDR topics, mainly because it's a system by itself and has to cover all the topics that usually appear in a SDR environment. The areas to be discussed within the paper are covered by the following topics:

- **Connectivity model.** This topic will define how to cope with the communication model established between the different subsystems of the radio. More precisely, this chapter will investigate the requirements needed to be fulfill for the establishment of the communications between Red / Black subsystem and CS/S (crypto subsystem) and communications between Red subsystem and black subsystem.
- **Software API's.** A common set of API's will be defined to allow waveforms to be ported on different platforms. This standardization effort will be focused in two different areas: Security API and Crypto API's.
- **Hardware connectivity.** Although this topic is outside the scope of this paper, it is mandatory to identify a common standard to connect the security subsystem to the Radio.

The paper presents the work performed in the TERSO Spanish National platform, in which the main guidelines will be drawn to achieve a common standard on SDR security. This development will focus its development in two ambitious objectives portability and interoperability. The portability objective will be achieve trying to maximize its capacity of adaptation to new standards, to better adopt them, without affecting the overall architecture of the system. On the other side, the interoperability feature is defined as the final objective of the system maximizing its capacity to be interoperable among the different waveforms and platforms.

2. GENERAL ARCHITECTURE DEFINITION

This chapter will present the general guidelines on how this security model has been implemented in the TERSO national platform. This explanation will include rationales that have been used to choose among different available technologies on the selected areas.

The design of the TERSO Spanish National platform is based on the SCA [2]; following the standard defined also in the security supplement [3]. Therefore, those two documents will be the basis of the design decisions taken along the document. However, the Crypto Subsystem software will not be compliant with the SCA specifications, because the interfaces offered to the subsystems are not specified in the standard. To solve this situation The CICM Mitre [1] interfaces will be used within the Crypto boundary.

Considering the aforementioned decisions is it possible to generate a diagram that illustrates the overall architecture for the security subsystem:

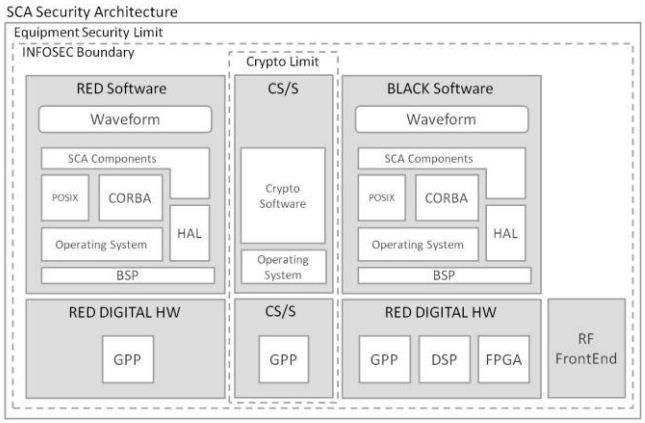

Figure 1. General design of the security architecture

The first assessment that derives from the picture is related with the security boundaries established within the radio limits. The system was divided in three different subsystems: Red, Black and Crypto.

With the general diagram of the system, the most important objectives to take into consideration in the detailed design of the TERSO platform were:

- Data and Audio Communications.
- User Authentication and basic user authorization. This operation implies the communication of the credentials through all the control elements of the platform.
- TRANSEC. The TRANSEC capabilities have been improved from the previous release of the platform, allowing the interactions between the Cryptographic subsystem and the Digital Modem.
- Communication through the Cryptographic subsystem. The implementation of the bypass communications are one of the greatest achievements of the TERSO.

These four areas will be the basis for the design of the security platform to be implemented in the TERSO architecture.

3. IMPLEMENTING THE STANDARD ARCHITECTURE

This chapter will present the decisions taken in order to finalize the implementation process.

3.1 Connectivity model. Allow me to talk, please.

Crypto modules have quite tight latency restrictions, so it is usually implemented in FPGA (Field Programmable Gate Arrays) or ASIC modules. This kind of implementation forces the SDR secure architecture to use a proprietary connectivity mechanism to communicate the different subsystems with the crypto module. As there are different set of data, there are different types of communications that have to be established:

- Control and data encryption communications. These communications are related with the normal operation of the crypto. The basic tasks related to this kind of communications are configuring the frequency hopping seed, encryption algorithms, user authentication, etc.
- Bypass communications. There are some communications that has to be established between the RED and the BLACK subsystems through the crypto subsystem. Those communications, like the starting up of a device in the black side, will use the CS/S as a bridge in order to transparently allow the connectivity.

The SCA standard encourages developers to use CORBA based communications, but as it was stated previously, the crypto subsystem can be unable of implementing an ORB in its core. The problem can be easily tailored in the case of control and data encryption communication because are directly driven by the user. The solution gets more complicated since the bypass communications has to be implemented transparent and unobstructively to all the COTS elements present in the platform.

3.1.1 Control and Data encryption communications.

Ciphering and deciphering the data flow is one of the basic operations of a secure system. In addition to these operations, the data flow coming from the crypto can be used also to generate the TRANSEC seed. Those functionalities have to be supported by the appropriate

communication mechanism, in order to achieve the objectives of latency and throughput.

The communications established by the RED and BLACK subsystems with the CS/S are driven by the use cases. The solution provided for this kind of communications will be based on the solution achieved for the bypass.

3.1.2 Bypass communications.

The objective of TERSO program was to establish the basis and implement the control bypass in order to enable transparent CORBA[4][5] communication between the RED and BLACK Subsystems. This high level goal is presented in the following picture:

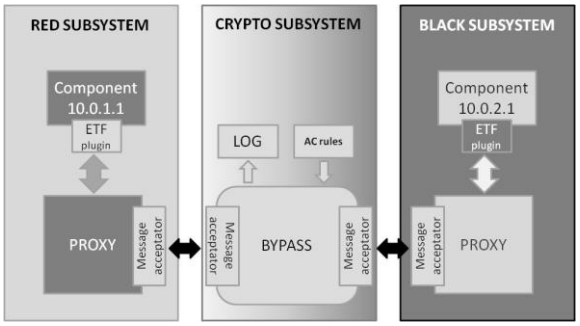

Figure 2. ETF solution to CORBA communications.

The architectural design was the starting point to have a full definition of the possible goals to be satisfied by the bypass. The goals went through a refinement process and the final list of TERSO objectives for the bypass was established as follows:

- Transparent communication between subsystems.
- Transparent monitoring of the communications between subsystems.
- Access control to the communications.
- The solution has to be independent from the middleware presented in the platform.

The figure shows that the communications architecture can be divided in three different areas of functionality, providing different capabilities, but sharing the same objective:

1. ETF Plugin. Developed for and shared by each component of the platform. This plugin will enable the communication with other components through the NON-CORBA capable Crypto.
2. Proxy Component. A proxy receiving the communications from the different components, which have to go through the CS/S.
3. Bypass System. The bypass system of the CS/S including the Log and access control capabilities.

One of the first problems faced during the design of the communication model was the impossibility of CORBA to identify the component that has started the communication.

The picture shows how, through the ETF plugin a virtual IP address is assigned to the component, identifying both the sender and the receiver components that are part of a communication. To handle the routing of these addresses some infrastructure is needed, so whenever the ETF plugin is used the communications will be redirected to a proxy. It's important to remark, that now, the CSS component in charge of taking the decisions has all the information about the participants of the communication, allowing the log capabilities or even the interruption of the communication if it's not included in the access rules.

The bypass architecture will be supported by the proxies presented in each subsystem, this way; all the SCA components can be implemented without taking into account specific details on the communication between the RED / BLACK Subsystems and CS/S.

In order to provide the achievement of transparency from the middleware, this agnosticism has to be extended to the operating environment presented in the GPP, designing the solution independently of the Core Framework, meaning that those components were not modified during the implementation. The following class diagram shows the detailed architecture:

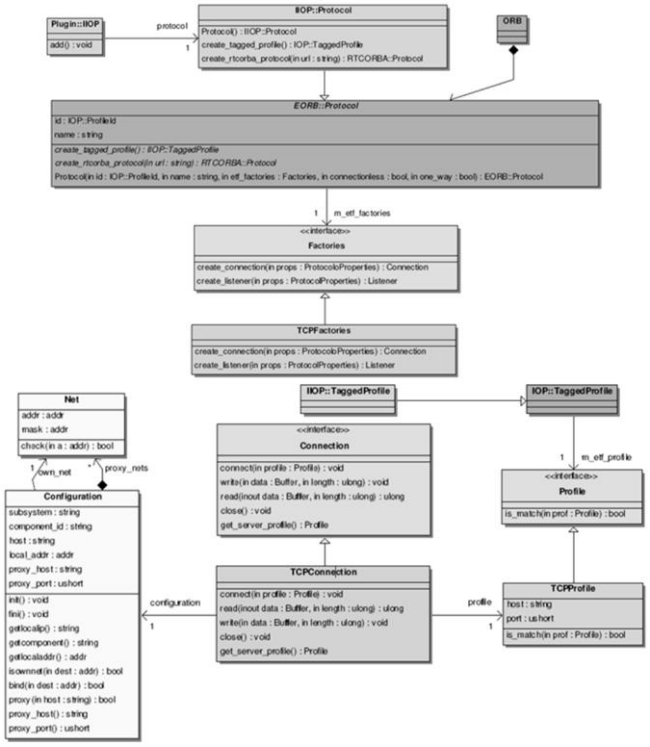

Figure 3. ETF Plugin class diagram

The diagram has been simplified in order to keep readability and comprehensibility, but it shows the basic

functionality of the solution. The class diagram shows how the ORB manages a set of communication protocols. Basically, the TCPFactories creates a TCPConnection object, which represents an IIOP peer to peer connection.

The class diagram can introduce some details on the behavior of the solution. Then, to establish a connection between subsystems, the ORB will use a TCPConnection component. The TCPConnection component has to deal with the localization of the addressee, checking if the communication is local or has to use the control bypass. The following sequence diagram shows how the establishment of the communication is performed:

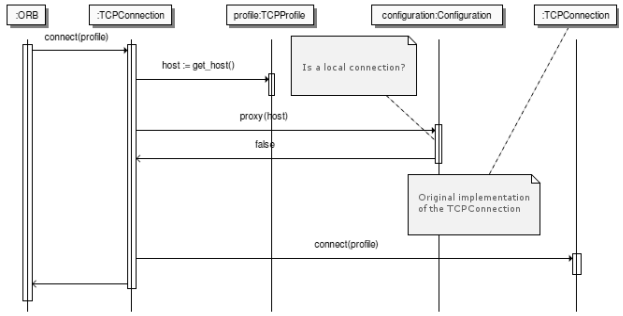

Figure 4. Sequence diagram of an ETF connection

Lastly, all the components created to implement this solution, will be deployed with the platform, being part of the SCA architecture implemented for TERSO.

3.2 Software API, let's talk the same language

The Security services described in the JTRS SCA Security supplement are defined as a group of functional packages being represented using UML notation.

The basic objectives of the Security API are to keep the assurance of the COMPUSEC boundary and to act as the interface to the Crypto Subsystem. These objectives can be achieved by securing the operating environment and implementing specific software components. This solution is based on the implementation of guards and monitors. The guards are components created to check that the information flows between authorized components, while the monitors check the correct operation of the processes within the system.

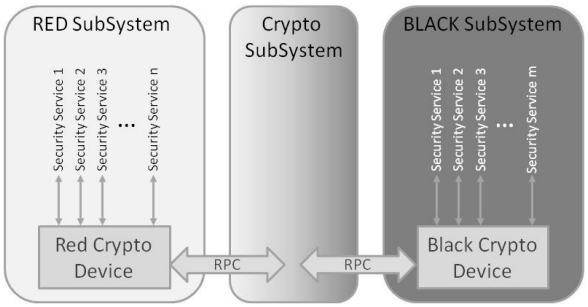

Figure 5. Security Services Implementation

In order to access all the functionality defined in the UML packages, the implementation of the Security API has been done following the basic schema depicted:

The component CryptoDevice (CF::Device) is the SCA representation of the CS/S. The software abstraction of each security functionality is made by the configuration of the ports of the CryptoDevice. Each port of this Device will have each own identifier, representing a single functionality. In detail, the access to these services is done as follow:

1. The CryptoDevice User invokes the getPort operation, using the Security Service identifier as a parameter.
2. The getPort operation will return a reference to the Security Service Provider.
3. Then, the Security Service User connects to the Security Service Provider by the use of connectPort operation.
4. Now the Security Service User is able to call the security service functionality.

Since the architecture follows the well-known red-black model, it's important to define in which subsystem the security functionality will be implemented. The table below indicates the subsystem location of each component:

Service Package	Security Service	Subsystem
Security	Management	RED
Fill	Port	RED
	Port User	RED
	Bus	RED
	Management	RED
Algorithm	Management	RED
Certificate	Management	RED
Crypto	Control	RED/BLACK
	Encrypt/Decrypt	RED/BLACK
Key	Management	RED
TRANSEC	Management	RED/BLACK
	Key Stream	BLACK
Policy	Management	RED/BLACK
Integrity and Authentication	Control	RED
	Digital Signatures	RED
Alarm	Management	RED/BLACK
Time	Management	RED/BLACK

Table 1. Security Service allocation

3.3 Crypto Subsystem, let's talk privately

The Crypto API's, are a critical problem, since they will be located in the heart of the SDR security subsystem. Trying to cover this problem, MITRE has released an *objective solution*, with the CICM (Common Interface to Cryptographic Modules). This API covers in general the

problems that appeared in the Cryptographic modules, but still lacks an SDR profile.

Basically, CICM specifies an application programming interface (API) to standardize the management of and access to cryptographic services offered by modules that contain cryptographic material and perform cryptographic operations. CICM offers application programmers a consistent interface to cryptographic functions, regardless of the underlying implementation providing the services.

3.3.1 General design of TERSO CS/S.

Although a number of commercial cryptographic interfaces have been standardized and are in use, CICM is the first generic cryptographic interface to be developed that meets the needs of a wide range of high assurance applications. Following the principles gathered in this open specification the objectives are focused on obtaining a Cryptographic Subsystem that fulfill the requirements established by MITRE, as much as possible. As it was previously stated that CICM is not tailored to SDR equipment and the development will include the identified modifications and inclusions. The following diagram shows the block structure of the solution implemented for the TERSO platform:

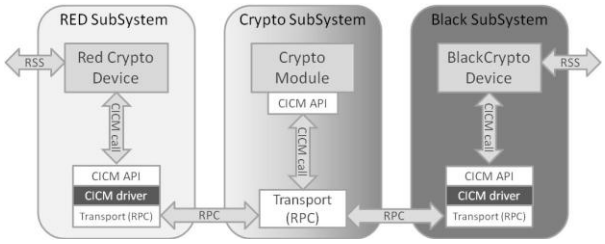

Figure 7. Communication through CS/S

The picture shows the actual solution presented in the TERSO platform in order to tackle the Crytpo access. The SCA *CryptoDevice* component shows the boundary within the SCA based platform. Just after the *CryptoDevice* is located the implementation of the CICM API specification, with the modifications needed in order to fulfill the objectives presented at the beginning of the project. The last component presented in the security architecture is the driver that connects to the actual CS/S.

The idea underlying this solution is to provide the needed abstraction of the CS/S to the platform that supports it. Then, changing the driver would allow upgrading or changing the CS/S independently from the platform.

3.3.2 Implementation of a SCA & CICM CS/S.

The functionality and design of the crypto subsystem can be categorized in the following functional groups:

- Module management.
- Algorithm management.
- Policy Management.
- Key Management.
- Channel management.
- Data management.

The different groups listed before, have been organized in a detailed design of the CS/S architecture:

This system design drove the implementation of the TERSO CS/S providing the following functionality:

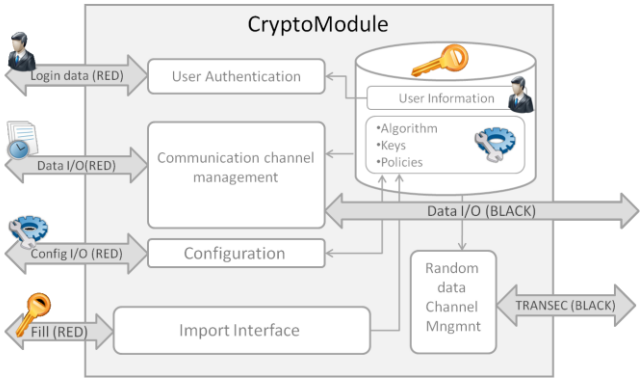

Figure 6. Detailed design of the CS/S

User Authentication. The current implementation only consider authentication against a user / password pair, being the role policies stored in the CS/S. This module has suffered some modifications from the standard in order to increase the possibilities and flexibility of the implementation. Currently, only the roles of USER and ADMINISTRATOR are considered.

Communication Channel management. This kind of channel will implement the COMSEC algorithms, receiving the data directly from a socket. The election of this kind of channels is driven by performance and channel ownership. Performance is a key feature of a crypto design. In the Spanish national platform the decision taken was to make the data flows directly through TCP sockets, avoiding the use of the RCP architecture and increasing the data throughput.

There is a problem in the CICM specification about the ownership of the channel. It is not possible to notify a subsystem the ownership of a channel when this subsystem is not the creator of the channel. The use of a direct channel only imposes the use of certain ports to perform the write / read operations.

For the purpose of the invocation of the operations related to this type of channels, the CICM standard provides the following functions:

- CICM_CM_CreateChannel_Encrypt
- CICM_CM_CreateChannel_Decrypt

Is in this step, at the creation of the channel, where all the parameters are needed. These parameters will include the

algorithm, the keys, the subsystems that are performing the communication and the associated ports.

TRANSEC Channels. The objectives of this group of functions are:

- Reliable time source. It's provided by the CICM function `CICM_MM_GetModuleDateTime`, which returns the time of a cryptographic module.
- Random number generator. In order to produce a sequence of random numbers it's necessary to use the indirect channel configured to receive synchronous data.

Import Interface. TERSO CS/S can import new cipher algorithm, security policies and keys from a RS-232 port. CICM provides different functions to import those objects:

- `CICM_AM_ImportAlgorithm`
- `CICM_KM_ImportKey`
- `CICM_PM_ImportPolicy`

keeping outside the scope the transfer process from outside the CS/S. On the other hand, the SCA does consider these possibilities, including also additional functionality like configuration, start or stop the object. As the main objective of TERSO was to be SCA compliant, some adaptation to the original API was performed:

- `CICM_INDRA_ConfigureImport`
- `CICM_INDRA_EnableImport`
- `CICM_INDRA_Import`
- `CICM_INDRA_DisableImport`

These map the SCA functionality following the CICM nomenclature. This approach forces the implementation of 'Import' to identify the specific type of the object to be imported.

Configuration. This functionality group offers the possibility of object administration (keys, algorithms, and security policies) to the user. The functionality includes object identifier, information and deletion.

Module Management. Although CICM standard is intended to work on a module-based philosophy, the implementation on TERSO platform will consider all the crypto functionality gathered on one module only.

3.3.3 Profiling model for CICM

Although CICM is an established reference for Crypto API this standard still lacks functions to cover specific SDR functionality. Indeed, the CICM standard was born with the intention of providing a general crypto API and extending the purpose of its use to a wide variety of final equipments. This means that is not specifically designed to fulfill all the SCA requirements. In order to solve this problem, and after the implementation of TERSO, a profiling based development was presented within the scope of the project.

The following figure provides details on how this profiling model can be performed:

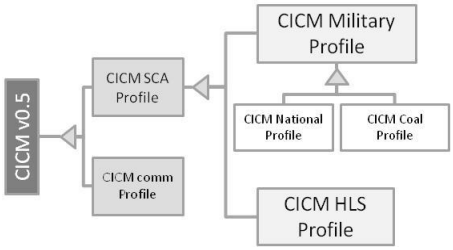

Figure 8. CICM Profiling model

Taking into account basic software concepts, this profiling hierarchy will allow improving specific areas and specific topics of different kinds of platforms and needs. Evolutions of the basic set of functionalities gathered in the CICM basic profile will be translated into the different profiles, while the evolution of the different specific profiles will run separately. This will allow also different levels of classification.

3.3.4 Improvement proposal for CICM.

Although it was not the main objective of TERSO national program, during the implementation of the CS/S some problems were identified within the CICM standard. The following improvements will be proposed:

- The import interface proposed by the CICM (specific functions for each type of object) is not compatible with the FILL interface proposed by the SCA.
- The SCA standard allows the possibility of deleting an object regardless the object is in use or not. On the other side, the CICM standard does not allow the deletion of objects that are considered in use.
- The CICM standard does not provide a way to identify the subsystem that has started the call. This possibility can be useful in order to apply security policy to the whole subsystem.
- The SCA defines in its standard the bypass policies that can be applied to a channel. This possibility is not considered in the CICM standard, making it incompatible with the SCA communications.
- The CICM user authentication process is described in terms of using SmartCard or USB devices. It's necessary to adapt the standard to recognize also the user / password authentication mechanisms.

During the TERSO program all these incompatibilities were tackled adapting the standard to specific needs, with

the intention of proposinge the changes to the CICM standardization process.

4. CONCLUSIONS

This article has presented one approach to security implementation on an SCA compatible radio. All the used components are part of the current ongoing initiatives not only in the SDR arena but in the current software efforts.

This paper intends not only to present the architecture, but it is also meant to establish an implementation reference for future developments on security SDR field. Although, all the different standards used to prepare this architecture are mature enough to be presented as a reference on this specific SDR topic, an evolution of all of them is necessary in order to establish a stable development.

Those modifications have been identified and faced during the specific implementation of this architecture, being the most remarkable one, the profiling of the Common Interface to Cryptographic Modules (CICM) for SDR. The MITRE CICM is a widely established cryptographic API for generic crypto modules, but it still lacks some specific functions for different SDR platforms. A model for future evolutions based on profiles for CICM has been presented also to be taken into consideration and for our own based development.

Besides the importance of the standardization process, the architecture has also fulfilled the main objectives that drove the initial design:

- Portability. The most important goal defined at the beginning was the increasing of the overall portability of the system. This is achieved by the use of common standards integrating them in the whole design.
- Scalability. The design of the radio architecture allows the upgrading of the different subsystems without impacting on the others. The objective takes shape in the figure of the crypto subsystem, allowing the substitution of the crypto emulation by a commercial crypto.
- Interoperability. An increase of the interoperability is achieved in means of increasing the capability of the platform to import and run different waveforms.

Lastly, the most remarkable achievement accomplished by the TERSO Spanish program is that it entails the first reference implementation of these standards in conjunction, presenting a working state-of-the-art integration of a CS/S into a SCA based platform.

5. REFERENCES

[1] D.J. Lanz et all, "Common Interface to Cryptographic Modules (CICM) Specification (DRAFT v0.5)" *MITRE product*. MP070175.

[2] JPEO JTRS SCA Standard v2.2.2

[3] JPEO JTRS SCA Security supplement 3.0

[4] OMG CORBA specification

[5] OMG CORBA ETF Communication standard