Development and implementation of security modules in the RINA prototype

Berta Delgado Ventosa

Director: Davide Careglio
Codirector: Jordi Perelló Muntan
GEP Tutor: Jasmina Berbegal Mirabent

Bachelor Degree in Informatics Engineering
Specialization in Computer Engineering
June, 2016
Abstract

Recursive InterNetwork Architecture (RINA) prototype is an experimental and innovative model of network architecture, based on inter process communications. The objective of RINA is to substitute the current redundant and complex TCP/IP Internet architecture providing a simplified network model. Originally proposed few years ago, RINA is currently investigated by many research, industrial and academic institutions and funded by some international projects. As security is one of the ongoing research topic in RINA, this project makes a specification, deployment and verification, of a certificate-based authentication module that guarantees security in RINA computer systems. The module is based on the TLS protocol, specifically this project will develop the TLS handshake part of the protocol, using OpenSSL cryptographic libraries. Furthermore, the performance of the authentication module is verified, tested, and at the end, compared with the use of no-authentication policies. It is worth mentioning that this project has been developed in collaboration with i2cat foundation in the framework of the European project PRISTINE. The implementation of this security module will be integrated in the RINA toolset and will be part of the outcomes of the PRISTINE project.

Resum

Recursive InterNetwork Architecture (RINA) és un prototip experimental i innovador que proporciona un nou model en l’arquitectura de xarxes. El objectiu de RINA és substituir al model redundant i complex de l’arquitectura TCP/IP del Internet actual i, proporcionar un nou model de xarxes simplificat. El model es va proposar ja fa uns anys, actualment està sent investigat per centres de investigació, industrials i acadèmics, finançats per projectes internacionals. Ja que la seguretat és un tema de investigació en curs a RINA, aquest projecte especifica, desenvolupa i valida un mòdul d’autenticació basat en certificats, que garanteix la seguretat d’aquells sistemes informàtics que utilitzin RINA. El mòdul està basat en el protocol TLS, aquest projecte desenvolupa específicament la part del protocol corresponent al TLS handshake, fent servir les llibreries criptogràfiques d’OpenSSL. Per acabar, es verifica i s’analitza el rendiment del nou mòdul i, es compara amb el rendiment obtingut quant no s’utilitza cap política d’autenticació. Aquest projecte ha estat desenvolupat en col·laboració amb la fundació i2CAT en el marc del projecte europeu PRISTINE. La implementació d’aquest mòdul de seguretat s’integrarà en el conjunt d'eines de RINA i serà part dels resultats obtinguts pel projecte PRISTINE.
Resumen

Recursive Internetwork Architecture (RINA) es un prototipo experimental e innovador que proporciona un nuevo modelo en la arquitectura de redes. El objetivo de RINA es sustituir al modelo redundante y complejo de la arquitectura TCP / IP del Internet actual, y proporcionar un nuevo modelo de redes simplificado. El modelo se propuso hace ya unos cuantos años, actualmente está siendo investigado por centros de investigación, industriales y académicos, financiados por proyectos internacionales. Ya que la seguridad es un tema de investigación en curso en RINA, este proyecto especifica, desarrolla y valida un módulo de autenticación basado en certificados, que garantiza la seguridad de aquellos sistemas informáticos que utilicen RINA. El módulo está basado en el protocolo TLS, este proyecto desarrolla específicamente la parte del protocolo correspondiente al TLS handshake, utilizando las librerías criptográficas de OpenSSL. Por último, se verifica y analiza el rendimiento del nuevo módulo y se compara con el rendimiento obtenido cuando no se utiliza ninguna política de autenticación. Este proyecto ha sido desarrollado en colaboración con la fundación i2CAT en el marco del proyecto europeo Pristine. La implementación de este módulo de seguridad se integrará en el conjunto de herramientas de RINA y será parte de los resultados obtenidos por el proyecto Pristine.

Acknowledgments

I wish to express my sincere gratitude to Davide Careglio and Jordi Perelló Muntan for his guidance and help throughout the development of the project. I would also like to thank Eduard Grasa and the rest of i2cat members for their assistance, advice and time, without whose help, the project would not have been successful.

I am also sincerely grateful to my parents who support me emotionally and economically, during all these years. Thanks to my family and friends who has always been there. Finally, I appreciate the learning experience I gained from studying at FIB, and especial mention to my fellow classmates, for making my stay here a pleasant and memorable one.
ACRONYMS

AP Application Process
CACEP Common Application Connection Establishment Protocol
CDAP Common Distributed Application Protocol
DAF Distributed Application Facility
DAP Distributed Application Process
DIF Distributed IPC Facility
FP7 Seventh Framework Programme for Research and Technological Development
IP Internet Protocol
IPC InterProcess Communication
MAC Message Authentication Code
OS Operating System
PDU Protocol Data Unit
QoS Quality of Service
RIB Resource Information Base
RINA Recursive InterNetwork Architecture
RSA algorithm (Rivest-Shamir-Adleman)
SDK Software Development Kit
SDU Service Data Unit
TCP Transmission Control Protocol
TLS Transport Layer Security
VLAN Virtual Local Area Network
VPN Virtual Private Network
Contents

1. Introduction....................................................................................................................................... 9
   1.1 Context.......................................................................................................................................... 9
   1.2 Stakeholders.................................................................................................................................. 12
       1.2.1 Developer ................................................................................................................................. 12
       1.2.2 Project director, codirector and support.................................................................................. 12
       1.2.3 Target audience ...................................................................................................................... 12
   1.3 State of the art............................................................................................................................... 13
       1.3.1 History and current research activities in RINA ................................................................. 13
       1.3.2 RINA .................................................................................................................................... 14
       1.3.3 Security .................................................................................................................................. 15
       1.3.4 Previous work on security in RINA ....................................................................................... 17

2. Scope of the project ....................................................................................................................... 18
   2.1 Objectives ................................................................................................................................. 18
   2.2 Technical competences ............................................................................................................ 19
   2.3 Requirements ........................................................................................................................... 19
   2.4 Risks management .................................................................................................................... 20

3. Methodology .................................................................................................................................... 21
   3.1 Monitoring tools ....................................................................................................................... 21
   3.2 Validation method ..................................................................................................................... 22

4. Planning ........................................................................................................................................... 23
   4.1 GEP tasks description ............................................................................................................... 23
   4.2 Tasks description ....................................................................................................................... 23
       4.2.1 Documentation ......................................................................................................................... 23
       4.2.2 Design ..................................................................................................................................... 24
       4.2.3 Setting work environment .................................................................................................... 24
       4.2.4 Implementation ...................................................................................................................... 24
       4.2.5 Analysis ................................................................................................................................ 24
       4.2.6 Final documentation ............................................................................................................. 25
   4.3 Time estimation .......................................................................................................................... 25
   4.4 Alternatives and action plan ...................................................................................................... 26
   4.5 Gantt diagram ........................................................................................................................... 27
4.6 Corrections to the initial planning ................................................................. 28
4.7 Resources ...................................................................................................... 28
4.8 Applicable regulations .................................................................................. 29
4.8.1 Licenses ..................................................................................................... 29

5. Budget and sustainability ................................................................................. 31
5.1 Budget ........................................................................................................... 31
5.1.1 Human Resources ..................................................................................... 31
5.1.2 Material Resources .................................................................................. 32
5.1.3 General expenses ..................................................................................... 32
5.1.4 Total Budget ............................................................................................. 33
5.1.5 Control management ............................................................................... 33
5.1.6 Corrections to the initial budget ............................................................... 34
5.2 Sustainability ................................................................................................. 34
5.2.1 Economic dimension .............................................................................. 34
5.2.2 Social dimension ...................................................................................... 35
5.2.3 Environmental dimension ...................................................................... 35

6. Authentication modules in the RINA prototype .............................................. 37

7. TLS authentication policy ................................................................................. 39
7.1 New authentication policy: AuthTLSHandshake .......................................... 40
7.2 Specification .................................................................................................. 41
7.3 Implementation ............................................................................................... 46

8. Validation and performance analysis ............................................................ 49
8.1 Authentication policy verification scenario .................................................. 49
8.2 Configuration .................................................................................................. 50
8.3 Certificate generation .................................................................................... 51
8.4 Traces of the verification experiments .......................................................... 54
8.5 Performance analysis ...................................................................................... 55

9. Conclusions and future work .......................................................................... 58

10. Bibliography ..................................................................................................... 60

APPENDIX

A. Configuration of the system ............................................................................ 64
B. User’s guide ........................................................................................................................................ 66

C. Performance analyses ......................................................................................................................... 70
   C.1 Round Trip Time ................................................................................................................................ 70
   C.2 Throughput ....................................................................................................................................... 70

D. Traces .................................................................................................................................................. 72
1. Introduction

1.1 Context

The internet is a global network that connects millions of computers, probably one of the most revolutionary creation of the last century. But, driven by the emerging applications, networks and the growth in use, the Internet has become an architectural patchwork of growing complexity, which strains to cope with new technological requirements. The Internet has a lack of security and scalability; it needs to move beyond to prosper in the long term.

Nowadays, packet switching technology that allows electric data exchange globally over the Internet is TCP/IP [1]. Even its great popularity, is it true that TCP/IP has a number of initial flaws design [2], that complicates its adaptability to the current and future needs. Is not a surprisingly fact, since this technology was designed at the beginning of the 70s of last century, when Internet (ARPANET in those years) was only a small group of geographically distributed computers over the United States. But, in the last decades, the size of the Internet has grown exponentially and also its expected benefits. Although TCP/IP technology has evolved these years, it has been decided to maintain its initial design, introducing progressively extensions focused on solving the needs of the moment. The final result of this evolution has been a TCP/IP technology increasingly complex due to its initial flaws design [3].

For example, two initial big design flaws of the Internet based on TCP / IP are: 1) the public global address space, where coexist with all devices connected over public address of the world; 2) the decision to identify a system with the address of their network interface and not with a unique address per node. The first of these design flaws, has generated serious problems of scalability in Internet addressing, causing depletion of the current public IPv4 directions. Additionally, this giant public space address also prevents any technological update, because the number of affected devices would be enormous. In other words, it would be the need to stop all the Internet to perform updates, something unthinkable today, due to the large business activities that it performs.

The second design flaw mentioned, also erodes the scalability of the Internet, preventing proper management of multi-homing scenarios, for example, when a single computer
connects to multiple network providers’ services for robustness reasons, a very common fact today. Something similar happens in mobility scenarios (crucial 5G functionality) when a mobile terminal changes the access point to which it connects to the network from one domain to another (and therefore changing IP address). One part of the scientific community sees the solution to the Internet problems in IPv6 addressing. Many others, however, identify IPv6 as just another temporal patch, since with this solution, routers must be able to store and manage many more entries and even longer.

The Recursive InterNetwork Architecture (RINA) prototype [4], is a new architecture model that simplifies the current Internet, and tries to build a new paradigm in order to solve the current shortcomings. Its fundamental idea is that networking essentially is inter-process communication (IPC).

RINA rebuilds the outline of the Internet layers and creates a new concept. It consists in forming a model that comprises a single repeating layer named DIF (Distributed IPC Facility). The DIF is the minimal set of component to allow distributed IPC between different application processes. There are several benefits resulting from the use of these new architecture; it supports multi-homing, mobility, Quality of Service (QoS), improves security and has a configurable environment.

The RINA network architecture, appears as a clean-state solution to the current TCP/IP Internet, based on the basis of the interconnection process network. It is important to stress that RINA is not only an extra layer for the current Internet, trying to solve its limitations and associated problems, but a new network architecture designed from zero, but without committing the early design flaws of TCP/IP.

In essence, the RINA architecture [5] reproduce the full structure of the Internet, describing a model where a single layer (DIF), is repeated indefinitely and recursively. The DIF implements the minimum set of mechanisms that enable basic communication between processes, while all specific configurations of this communication remain open as a programmable policy (e.g. those policies routing, quality of service or security that implements a specific DIF). Thus, RINA includes programmability in design directly. In addition, as the DIFs are totally independent from each other (we could understand them as virtual containers), it also incorporates virtualization. All network systems are added to a DIF through a specific instance of an IPC Process (IPCP). Another remarkable feature is that in RINA, there is a unique global address space as in the case of TCP/IP but one
for each independent DIF, thus eliminating the limitations of the current Internet addressing.

Figure 1, shows a scenario of a global RINA network example, where an end user is connected (via VPN) with its laptop to a cloud services, running on a virtual machine from a data centre, through 2 different internet service providers. As shown in the figure, a RINA network scenario comprises multiple DIFs, stacked on each other recursively, each of which covers a different network segment. Then, each of these DIFs would be programmed with specific policies that allow proper communications operation in the specific area that it covers. For example, the communication via a wireless network referred as Home DIF in the figure, would require political broadcast package, security, quality of service, etc, different than the ones required for the Backbone DIF.

Security is one of the ongoing branches in RINA experimental field. Without security policies, our confidential information or private messages could be viewed, changed or stolen. Protection against this type of crime and fraud is extremely important, as everyone is a potential target, not only individuals are in danger, but also corporations and governments have risks. We must enhance protection in our data, since in the modern world information is one of the most powerful currency.

The aim of this degree final project is to collaborate with i2cat [6] in order to implement an operative security module for RINA prototype. I2cat is a Catalan research centre actively working on the RINA prototype. They have different fields of study as QoS support or congestion control.
The security module implemented in this project will authenticate two IPC processes in order to establish a secure connection between them. With this project we will be one step closer to achieve a fully functional RINA prototype.

1.2 Stakeholders

There are several stakeholders involved in the development of this project, either directly or indirectly. In this section, all the people who made it possible are described, from the early days until it has reached the final goal.

1.2.1 Developer

The developer is a key component since it is the one who will carry out the project and test its performance. The developer function is more than just a technical job, she will write the documentation, prepare the presentations and manage the project.

1.2.2 Project director, codirector and support

The project director is Davide Careglio and the codirector is Jordi Perelló Muntan. They will supervise the technical aspects and control the correct achievement of the objectives. They have had periodical meetings with the developer so they can check the correct progress of the project.

Eduard Grasa is the support person within i2cat, who has helped with the better understanding of RINA and also with the new security module design and implementation.

1.2.3 Target audience

Implementing security modules in RINA prototype is directly targeting RINA research community. As it is a research topic, it does not have an immediate effect to the industry or the end-user.

Trying to stretch the capabilities of the current architecture, does not seem to provide an answer to the current Internet problems, so RINA also targets networks of different physical media, and distributed applications that demands for performance, stability and efficiency. In a long-term best case, RINA will be used in the everyday life and indirectly the security modules that it contains.
1.3 State of the art

1.3.1 History and current research activities in RINA

In the early days, RINA was just a theoretical model architecture presented by John Day in his book “Patterns in network architecture: A return to fundamentals” [2]. He took into account the current network problems and learn from the failure of OSI model [7] in order to build a new concept of network modelling.

In 2010 a project named IRATI [8] started the deployment of an open source prototype for Linux. With their specification and implementations, RINA was closer to be used in the real world. Nowadays the prototype is partially developed, there is still ongoing activities to implement all its capabilities and enhance its performance. Different research groups are working in these goals.

PRISTINE [9] is a FP7 (Seventh Framework Programme for Research and Technological Development) European project with 15 partners: WIT-TSSG, i2CAT, Predictable Network Solutions, Telefónica I+D, Thales, Nexedi, B-ISDN, Atos, University of Oslo, , Brno University, IMT-TSP, CREATE-NET, iMinds and UPC.

The aim is to develop a Software Kit (SDK) for RINA, under Linux operating systems as its initial baseline (IRATI). It enables the possibility to customize the behaviour of the DIFs, it also targets congestion control, resource allocation, routing, security and network management. PRISTINE is described in this section, as is the project that englobes this degree final work.

PRISTINE will end in less than 5 months. Currently there is a new project named ARCFIRE [10] (being i2cat also one of its partners): the aim is to experimentally demonstrate at large scale the key benefits of RINA, produce robust RINA software suite for large-scale and long-lived experiments and raise the number of organization involved in RINA research, development and innovation activities. ARCFIRE will allow the IRATI implementation to be used both for rich experimental research activities and for internal trial deployments by network operators.
1.3.2 RINA

rina is the outcome of the effort to solve the current problem in computer networking. It uses some particular terminology for its components. In this section we will explain the basic concepts to understand how it works.

Distributed Application Process (DAP) is the name given to the instance of a program being executed on the system. A DAP can have one or more tasks, it also contains functions for managing its resources (processor, IPC, storage). Two or more cooperating DAPS, which exchange information using IPC, are named Distributed Application Facility (DAF). They collaborate to perform a defined task. In Figure 2, it is shown the basic elements of the architecture.

These DAPs communicate using a single application protocol called CDAP (Common Distributed Application Protocol), which enables two DAPs to exchange structured data in the form of objects. CDAP provides an approach to share data over a network without having to create specialized protocols. It enables distributed application to deal with communications at an object level, rather than forcing applications to explicitly deal with serialization and input/output operations.

All of the DAP’s externally visible information is represented by objects and structured data in a Resource Information Base (RIB), which provides a naming schema and a logical organization to the objects known by the DAP.
In order to exchange information, DAPs need an underlying facility that provides communication services to them. This facility is another DAF whose task is to provide and manage Inter Process Communication services over a certain scope; hence this DAF is called DIF: Distributed IPC Facility.

DIFs can be thought of as a layer. A DIF enables a DAP to allocate flows to one or more DAPs, by just providing the names of the targeted DAPs and the characteristics required for the flow. Each DIF implements the same functions and uses the same protocols, but is configured with different policies to fulfil the particular requirements of the layer. DIFs can also be the users of other underlying DIFs, creating in this way the recursive structure of the RINA architecture.

DAPs may not trust the DIF they are using, therefore may decide to protect their data before writing it to the flow - for example using encryption - via the SDU (Service Data Unit) Protection module.

1.3.3 Security

In this subsection we focus in the security investigation branch as is the main topic of the project. In a RINA network, security functions sit at two levels: the IPC (or DIF) level and the domain level.

The IPC manages aspects of security that are specific to the DIF, while the domain manages aspects that apply to multiple DIFs or the interactions between them. The IPC process is responsible for authentication and access control as well as protecting application, control and management data. It monitors and logs events that occur within the IPC or DIF. The IPC detects intrusions and failures and is responsible for implementing solutions. The domain determines the level of trust that an IPC has in other IPCs and DIFs. It is responsible for managing the keys needed by the DIF for authentication, access control and protecting data. The domain also manages network flow protection as well as redundancy functions.

The recursive model of the RINA architecture provides a clear security model, Figure 3, in which the trust relationships between layers (DAFs or DIFs) and between members of a single layer are well identified. Once the flow through the underlying DIF has been created, the new IPC member must create an application connection with the DIF member. The new IPC member and the DIF member may authenticate each other
according to the DIF’s policy (we focus in authentication as is the topic of the project). The DIF member may also make an access control decision whether the new IPC process can join the DIF. If the new IPC member is allowed to join the DIF, it is then initialised with the current information in the DIF, e.g. addressing, policies, keys, etc. Once the IPC has enrolled, it can offer IPC services to the layer above and exchange data with other members of the DIF using CDAP.

Another area of RINA where access control may be used is in protecting remote operations on the RIB. Strict access control is possible by authorising access to each individual object in the RIB for each of the six CDAP operations (create, delete, read, write, start, stop).

The network’s primary objective is the transmission of data as a service to applications. From this point of view, its security concerns are related to maintaining the availability and reliability of this service. For these reason, the priority security requirements in RINA are resiliency and high availability.

Within a RINA network, the responsibility for detecting and recovering from and locating failures lies firstly with the DIF in which the failure occurs. If the DIF is unable to sufficiently recover from a failure, a higher level DIF should take over responsibility for recovery and further propagate the recovery if it cannot resolve the failure. The failure of an IPC process should be resolved within the DIF. The RIB daemon will have to converge the state of the DIF related to forwarding (i.e. the forwarding tables) to cope with the failure. If this is insufficient, the recovery action should propagate to the higher level DIF which has sufficient capabilities.
Another security improvement on RINA prototype is that the node addresses are not exposed to applications, they are placed internally in a DIF. This decoupling of transport port allocation and access control cause resiliency to data transport attacks, such as port-scanning. As a DIFs per se, are securable containers there is no need to use firewalls, since to join a DIF different policies as authentication, authorization, confidentiality and integrity are used. RINA is a new concept build from zero, so it has identified the proper placement of all the components on its architecture it has no need to patch the old model, this means less overhead and more effective solutions, offering less complexity than in the current Internet.

1.3.4 Previous work on security in RINA

There is a previous study in the field [11] related to this project. The main objective was to describe the initial specifications of authentication policies in RINA that we explain in chapter 6. This study shows, at an abstract level, the main application components that are related to the authentication procedures and the main interactions amongst them.

It also provides a theoretical specification on how the authentication processes in RINA should behave, and implement simple policies to test them. The result of this project was satisfactory. It was found that RINA architecture provides a clear security model, in which the trust relationships between layers (DIFs), and between members of a single layer are well identified. One step further is to implement complex authentication policies more secure and look-alike as the ones we use nowadays. That will be the aim of this project.
2. Scope of the project

This project has been developed in collaboration with i2cat (one of the research group of PRISTINE project). The task of i2cat and of the senior research Eduard Grasa in particular, has been to give the technical advice on the development of this project. The goal is to design an authentication protocol to enable more secure communication between IPCs.

The policy implemented will be based on the TLS (Transport Layer Security) [12] protocol. TLS is a secure communication protocol composed by two layers. The first one is the record protocol, it is the lowest layer and is used to encapsulate and protect higher-level protocols. It provides integrity and protection of the data. The record protocol divides the outgoing messages into manageable blocks and applies Message Authentication Codes (MAC) to secure the application data, and reassembles the incoming messages verifying them using also the MAC. It is also in charge of encrypting and decrypting the incoming and outgoing messages.

The second layer of TLS protocol corresponds to the handshake protocol, it is used to set up the secure channel and negotiate the parameters of a session using certificates. This project will develop a TLS handshake module, based on the cryptographic procedures of the TLS protocol, adapted for a RINA prototype.

TLS is the chosen security protocol as is widely adopted across the Internet, it is used in web sites, web browsers, e-commerce, e-mails and much more everyday applications. If RINA wants to be functional and used world-wide, it must offer security in this kind of services.

2.1 Objectives

The main objective is to design and implement an adaptation of the TLS handshake protocol, as said before, that allows client/server application to communicate without eavesdropping, tampering, or message forgery. RINA will work over Ethernet as it needs to be compatible with the existing mechanisms. To obtain this result we need to:

- Design a TLS handshake protocol compatible to RINA prototype.
- Implement the protocol using OpenSSL cryptographic libraries [13].
- Enable an IPC process to join a DIF using authentication.
• At the end of the implementation, an IPC process will be able to join a DIF in a secure way.
• TLS handshake module will be joint with the TLS record module (made by third parties), so the two layers of the TLS protocol can operate together.
  o Obtain a full TLS protocol for RINA.
• Study the throughput and the round trip time (RTT) of new policy and compare it to the use of no authentication.
• Additionally, there has been added a validation section to prove that the project meets the reliability, repeatability and reproducibility conditions.

2.2 Technical competences

In this subsection, technical competences for computer engineering specialization achieved in the project, are described and justified. The competences are the following:

• **CEC4.1** To design, deploy, administrate and manage computer networks.

In order to accomplish the objectives, a security module for a special kind of computer network has been specified, deployed and tested, so CEC4.1 requirement is satisfied.

• **CEC4.2** To demonstrate comprehension, to apply and manage the guarantee and security of computer systems.

This project is specifically focused on one of the key aspects of security in computer network, i.e. the authentication process. In particular, it deals with the specification, deployment and verification, by demonstrations of the handshake protocol of the authentication process that guarantees security in RINA computer systems.

For the development of the project, the most important related subject done during the degree are XC and XC2, they are necessary to understand the basic concepts of RINA architecture and its features. Nevertheless, subjects as PROP has been also important since there has been the need to manage a big C++ project.

2.3 Requirements

To make possible the communication between two IPCs, is needed two computers running under Debian operating system, as is the Linux OS distribution chosen for the
project. To accomplish such requirements, there has been used VirtualBox in order to emulate two Debian virtual machines, each machine performing as an IPC process. The use of emulation makes easier to work with two computers simultaneously.

Both virtual machines, are configured with RINA modules and provided with the OpenSSL libraries. They are both extremely important, since RINA modules enables a RINA environment over Ethernet, and OpenSSL libraries offers cryptographic APIs needed in the implementation of the authentication policy. Furthermore, OpenSSL libraries are required to create the corresponding certificates used in the policy. Additionally, RINA libraries are needed to be able to work with the RINA environment.

2.4 Risks management

As all projects, this one could have had some setbacks as well. The risks could have been the following:

- First of all, one of the authentication message could be poorly designed and there would be the need to change some of its parameters, in order to accomplish the authentication process. This problem would represent a loss of time since the policy would need to be redesigned and this affect also to the implementation process.

- There could be also some difficulties finding the right APIs in OpenSSL libraries that suit this project. In this case it won’t be a huge problem, just a delay in the project that has been taken into account.

- Also it could be problems with extern dependencies of this project that cannot be controlled. It may happen that the TLS record policy module is not finished by the time the handshake module must be tested. In this case, external modules that simply check the parameters will be implemented for testing the project.
3. Methodology

The project methodology proceeds to follow the waterfall method [14], as shown in Figure 4. This model is defined as a sequence of well-determinate actions. Its phases are: requirements analysis, design, development, test and release. It is used the iterative circle that joins test and development, since every time a new feature was implemented was also tested.

![Waterfall Method](Source: Self-devised)

The requirements analysis corresponds to the study of RINA special needs in order to adapt the TLS handshake protocol. Before this phase, there has been a documentation period to be able to understand the topic. The design phase is clearly defined; it corresponds to the design of the messages that the protocol needs. As is observed in Figure 4, next step is the implementation itself, it will be an iterative process of testing and coding. The release phase corresponds to the final deliverable of this final degree project. Some waterfall models have a maintenance phase, as this is a degree final project and has a limited period of time, maintenance phase will not apply.

An important part of the methodology of this project, corresponds to monthly meetings with the director and codirector of the project. It has been a fundamental part, in order to check the correct progress and the achievement of the objectives.

The project has been partially developed at i2cat research centre, so the questions and problem that have occurred during the process have been solved rapidly.

3.1 Monitoring tools

To proceed with the correct development and supervision of the project, a wiki entry has been created. There can be found the specification adaptation of the TLS handshake
protocol. The research group that I have collaborated with, has access to it so they have checked the correct design of the policy.

A software for controlling versions and manage the repositories called GitHub has been used. It allows to control distributed revisions and to manage the source code. With this tool the access to the code, as well as the recuperation of old versions have been guaranteed.

To communicate with the director and the codirector of the project regular communication tools have been used, as e-mail or in special cases Skype.

3.2 Validation method

Periodical and frequent meetings with the director and codirector have allowed the review of the project requirements, the correctness and the quality. The partial development of the project in i2cat, as said, have ensured constant communication that has facilitated the quick resolution of doubts and problems that might have arisen through the project. At the end, the correct performance of the authentication protocol has been tested with the i2cat support person.
4. Planning

This project has last about 5 months. The initial date of the project was February 8th (one week before the beginning of spring semester), and has continued until the end of June when the final presentation must be done.

In this section the tasks that have been carried out are explained, as well as the resources needed at each step. As this project has 2 well differentiated parts, firstly the tasks corresponding to GEP will be explained, and afterwards the project parts itself.

At the end, a Gantt diagram will be shown to expose the time dedication planned for each activity over the total time. GEP have been developed simultaneously with other project tasks. The meetings with the tutors are not shown, as they have not had a fixed time slots and have not impacted significantly on the time estimation.

4.1 GEP tasks description

GEP corresponds to the management part of the project. It is an extensive task carried out the first month, which has 6 deliverables during this period. In total, GEP has around 75 hours of work. In the following list we can see the quantity of hours needed to achieve each subtask of GEP.

- Context scope: 24.5h.
- Planning: 8.25h.
- Budget and sustainability: 9.25h.
- First oral presentation: 6.25h.
- Speciality specification’s: 8h.
- Final document and oral presentation: 18.25h.

4.2 Tasks description

4.2.1 Documentation

The very first task of the project was to document about RINA. The directors of the project provide several information about the topic. It was quite long period of autonomous learning, documentation about RINA and study about TLS protocol.
4.2.2 Design

The second task corresponds to the design process. Once RINA and TLS is understood, there has been analysed the requirements needed for the protocol adaptation. After that, the specifications of the messages have been done in the wiki entry. The wiki is available for all the people involved in the project. The directors have taken also an important part in this step, they have checked that the protocol design has no mistakes. This phase precedes the implementation task.

4.2.3 Setting work environment

After the design phase and before the implementation the development environment must be prepared. First of all, Debian OS has been downloaded, to be able to create a virtual machine (VM) that emulates this system. Next, all RINA modules have been installed, this include kernel and user space modules. As is a new environment, text editors and some other minor tools has been set. Afterwards, this virtual machine has been cloned to obtain another identic VM, and enabled a connection between them.

Once setting the environment is done, it has been proceeded to verify that two IPCs process can connect to each other without authentication. At the end of this phase, some extra time was scheduled, since time to adapt and control this new work environment was needed.

4.2.4 Implementation

This was an iterative phase, it consisted in coding and testing every step was made in the implementation of the protocol. The supervision of the directors and support person was important in this process, in order to verify the correctness of the implementation. This task also coped with the interoperability of the TLS record module and the TLS handshake module. At the end, correct behaviour between them was tested.

4.2.5 Analysis

This is the task that proceeds the implementation phase. This task consisted on the evaluation of the policy. We have evaluated the throughput and the round trip time (RTT) obtained and have compared between the use of no authentication and TLS authentication.
4.2.6 Final documentation

This is the final task of this project. It will englobe the writing of the final paper and the preparation of the slides needed in the final presentation.

4.3 Time estimation

The following Table 1 shows the time estimation for each task. The tasks follow a temporal dependency; one task must be accomplished before the next task can start (except GEP and testing that will be done simultaneously with other tasks, as explained before).

<table>
<thead>
<tr>
<th>Tasks</th>
<th>Estimated time in hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td></td>
</tr>
<tr>
<td>Autonomous learning</td>
<td>40</td>
</tr>
<tr>
<td>GEP</td>
<td></td>
</tr>
<tr>
<td>Context Scope</td>
<td>24.5</td>
</tr>
<tr>
<td>Planning</td>
<td>8.25</td>
</tr>
<tr>
<td>Budget and sustainability</td>
<td>9.25</td>
</tr>
<tr>
<td>First Oral Presentation</td>
<td>6.25</td>
</tr>
<tr>
<td>Speciality specification</td>
<td>8.5</td>
</tr>
<tr>
<td>Final document and oral presentation</td>
<td>18.25</td>
</tr>
<tr>
<td>Design</td>
<td></td>
</tr>
<tr>
<td>Analyses requirements</td>
<td>30</td>
</tr>
<tr>
<td>Design and specifications</td>
<td>50</td>
</tr>
<tr>
<td>Setting work environment</td>
<td></td>
</tr>
<tr>
<td>Install RINA modules</td>
<td>5</td>
</tr>
<tr>
<td>Install text editor and minor tools</td>
<td>2</td>
</tr>
<tr>
<td>Check correct installation</td>
<td>3</td>
</tr>
<tr>
<td>Implementation</td>
<td></td>
</tr>
<tr>
<td>Code</td>
<td>190</td>
</tr>
<tr>
<td>Test</td>
<td>70</td>
</tr>
<tr>
<td>Modules adaption</td>
<td>30</td>
</tr>
<tr>
<td>Analyses</td>
<td></td>
</tr>
<tr>
<td>Throughput evaluation</td>
<td>20</td>
</tr>
<tr>
<td>Comparison</td>
<td>30</td>
</tr>
<tr>
<td>Final documentation</td>
<td></td>
</tr>
<tr>
<td>Final paper</td>
<td>50</td>
</tr>
<tr>
<td>Oral presentation</td>
<td>25</td>
</tr>
<tr>
<td>Total</td>
<td>620h</td>
</tr>
</tbody>
</table>

*Table 1 Time estimation in hours (Source: Self-devised)*
4.4 Alternatives and action plan

Probably there will be some deviations in the initial plan, as is difficult to foresee the exact amount of hours needed for each task. It could happen that the task, last less than planned. In this case it will not be a problem, we will start the next phase. Otherwise, if they last longer than expected, we will need to add more work hours a day.

In order to prevent that one task, overtake another, there has been some extra time scheduled. For example, the implementation phase last almost two months in the Gantt diagram, but it will be surely less than one month and a half. Then, if we have problems, we will have time to solve them. In the module adaptation subtask, there has been added 20 extra hours. This time will be used in case there is need to implement by ourselves the record modules (explained in risk management section).

In the worst case, if all the previous tasks have been delayed, the comparison between the use of no-authentication and authentication policy will be cancelled. We will just study the throughput and RTT of the new policy and we will have the proper time to document the project and prepare the final presentation.
4.5 Gantt diagram
4.6 Corrections to the initial planning

The deadlines made during GEP course fits perfectly to reality. Although, the extra hours scheduled were too high and we have reduced them. Initially they were 693h planned and now 620h of working hours have been estimated. Even though, the planning does not get affected as the previous scheduled hours represented too much work hours per day.

4.7 Resources

For the correct accomplishment of the project we have need quite a lot of resources, they can be human or material. The human resources correspond basically to the developer, but the support person and the directors of the project are also implied. They are needed to supervise and verify the fulfilment of the all the tasks. The developer has been also involved in all the activities carried out during this period.

There has been needed the following material resources (hardware and software) listed in Table 2:

<table>
<thead>
<tr>
<th>Development tools</th>
<th>Tasks involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lenovo i5, 8GB RAM, windows 10</td>
<td>To develop the security module and write the project memory</td>
</tr>
<tr>
<td></td>
<td>Documentation, GEP, design, Setting work environment, implementation, analyses and final documentation</td>
</tr>
<tr>
<td>Microsoft Office</td>
<td>To write the project memory</td>
</tr>
<tr>
<td></td>
<td>GEP and final documentation</td>
</tr>
<tr>
<td>Ganter generator</td>
<td>To generate a Gantt diagram</td>
</tr>
<tr>
<td></td>
<td>GEP</td>
</tr>
<tr>
<td>Eclipse</td>
<td>IDE to develop the project</td>
</tr>
<tr>
<td></td>
<td>Implementation</td>
</tr>
<tr>
<td>Microsoft Office</td>
<td>To do the memory and presentations of the project</td>
</tr>
<tr>
<td></td>
<td>GEP and final documentation</td>
</tr>
<tr>
<td>VirtualBox</td>
<td>To emulate Debian OS</td>
</tr>
<tr>
<td></td>
<td>Setting work environment, analyses and implementation.</td>
</tr>
<tr>
<td>Debian OS</td>
<td>To install RINA modules</td>
</tr>
<tr>
<td></td>
<td>Setting work environment, analyses and implementation.</td>
</tr>
<tr>
<td>GIMP2</td>
<td>To create and edit pictures</td>
</tr>
<tr>
<td></td>
<td>GEP and final documentation</td>
</tr>
<tr>
<td>Adobe Reader</td>
<td>To visualize documents</td>
</tr>
<tr>
<td></td>
<td>Documentation, GEP, design and final documentation</td>
</tr>
<tr>
<td>OpenSSL library</td>
<td>To be able to use APIs</td>
</tr>
<tr>
<td></td>
<td>Implementation and analyses</td>
</tr>
<tr>
<td>Camera</td>
<td>To record oral presentation</td>
</tr>
<tr>
<td></td>
<td>GEP</td>
</tr>
</tbody>
</table>
### 4.8 Applicable regulations

This final degree project does not have any governmental law applicable in its development, but it needs to follow the licences of the programs/libraries which uses.

#### 4.8.1 Licenses

In the following section we can see the licenses required, we have respect all of them during the project.

- OpenSSL license: [https://www.openssl.org/source/license.html](https://www.openssl.org/source/license.html)

- Librina: GNU Lesser general public license, Version 2.1
  


- Linux kernel: GNU general public license, Version 2
  
  [https://github.com/IRATI/stack/blob/master/linux/COPYING](https://github.com/IRATI/stack/blob/master/linux/COPYING)
- Rina-tool: GNU general public license, Version 2
  https://github.com/IRATI/stack/blob/master/rina-tools/LICENSE

- Jsoncpp: MIT license
  https://github.com/IRATI/stack/blob/master/rinad/LICENSE-jsoncpp

- Rinad: GNU general public license, Version 2
  https://github.com/IRATI/stack/blob/master/rinad/LICENSE

- TLS handshake module: GNU general public license.
  https://github.com/edugrasa/stack/blob/pristine-1.5-authtls/librina/src/tlshand-authp.cc

  In the link below can be found the description of the license, as well as, the implementation of the TLS authentication policy that will be used for i2cat.
5. Budget and sustainability

5.1 Budget

In this subchapter we will see the analyses of the estimated budget for this project. The expenses have been divided in three groups: human resources, material resources (hardware and software) and indirect costs. All the estimated hours correspond to the planning made in the Gantt diagram.

5.1.1 Human Resources

This project is fully developed by a single computer engineer student. Even the project has a director, codirector and an adviser, is consider them as a one single project manager as they assume the same role in the project. In Table 3 is shown the total amount of human resources and the estimated price each role gains per hour:

<table>
<thead>
<tr>
<th>Role</th>
<th>Estimated hours (h)</th>
<th>Price per hour (€/h)</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computer engineer</td>
<td>620</td>
<td>14</td>
<td>8.680 €</td>
</tr>
<tr>
<td>Project manager</td>
<td>146</td>
<td>35</td>
<td>5.110 €</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td>13.790 €</td>
</tr>
</tbody>
</table>

*Table 3 Total estimation of human resources (Source: Self-devised)*

In table 4 below is illustrated the contribution for each specific role in the budget depending or the tasks of the project. For the sake of brevity and clarity, there has been computed hours in task, subtasks are not included.

<table>
<thead>
<tr>
<th>Task (including subtasks)</th>
<th>Project Manager</th>
<th>Computer engineer</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Documentation</td>
<td>10 h</td>
<td>40 h</td>
<td>910 €</td>
</tr>
<tr>
<td>GEP</td>
<td>0 h</td>
<td>75 h</td>
<td>1.050 €</td>
</tr>
<tr>
<td>Design</td>
<td>5 h</td>
<td>70 h</td>
<td>1.150 €</td>
</tr>
<tr>
<td>Setting work environment</td>
<td>1 h</td>
<td>10 h</td>
<td>217 €</td>
</tr>
<tr>
<td>Implementation</td>
<td>100 h</td>
<td>290 h</td>
<td>7.560 €</td>
</tr>
<tr>
<td>Analyses</td>
<td>20 h</td>
<td>50 h</td>
<td>1.400 €</td>
</tr>
<tr>
<td>Final documentation</td>
<td>10 h</td>
<td>75 h</td>
<td>1.400 €</td>
</tr>
<tr>
<td>Total</td>
<td>10 h</td>
<td>75 h</td>
<td>13.687 €</td>
</tr>
</tbody>
</table>

*Table 4 Itemized expenses of human resources (Source: Self-devised)*
5.1.2 Material Resources

In this subsection have been included both hardware and software resources. The delivery tools, monitoring and communication are not specified in the following figure, since they are all free of cost and they suppose a huge overhead in the table. In table 5, there is just specified the development resources.

<table>
<thead>
<tr>
<th>Product</th>
<th>Price</th>
<th>Service life</th>
<th>Amortized time</th>
<th>Approx. Amortized cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop</td>
<td>899 €</td>
<td>5 years</td>
<td>5 months</td>
<td>75,51 €</td>
</tr>
<tr>
<td>Windows 10</td>
<td>119 €</td>
<td>5 years</td>
<td>5 months</td>
<td>9,9 €</td>
</tr>
<tr>
<td>Microsoft Office</td>
<td>50 €</td>
<td>6 years</td>
<td>5 months</td>
<td>3,45 €</td>
</tr>
<tr>
<td>Gantter generator</td>
<td>0 €</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Eclipse</td>
<td>0 €</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>OpenSSL library</td>
<td>0 €</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VirtualBox</td>
<td>0 €</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Debian OS</td>
<td>0 €</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>GIMP2</td>
<td>0 €</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Camera</td>
<td>250 €</td>
<td>6 years</td>
<td>1 month</td>
<td>3,25 €</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>1.318 €</strong></td>
<td></td>
<td></td>
<td><strong>92,11 €</strong></td>
</tr>
</tbody>
</table>

*Table 5 Estimated cost of development tools (Source: Self-devised)*

5.1.3 General expenses

All projects have expenses not directly involved in the project, they are very important as they can suppose a huge increase in the final budget. In this subsection, is shown the indirect costs associated to this project, Table 6.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Laptop electricity (0.1 kW during 600h at 0.15€)</td>
<td>9€</td>
</tr>
<tr>
<td>Internet access (5months, 30€/month)</td>
<td>150€</td>
</tr>
<tr>
<td>Printing and binding (various copies)</td>
<td>25€</td>
</tr>
<tr>
<td>Power consumption (without laptop)</td>
<td>19.10 €</td>
</tr>
<tr>
<td>Transport (2 trimestral travel card under 25s discount, for 3 zones)</td>
<td>398.40 €</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>601.50€</strong></td>
</tr>
</tbody>
</table>

*Table 6 Estimated costs of general expenses (Source: Self-devised)*
5.1.4 Total Budget

To summarize Budget section, we conclude with a final table, Table 7, which shows the total necessary budgeting to accomplish the project. The contingency has been calculated as the 5% of the total costs (human resources, material resources and indirect costs). There has been chosen a low percentage for contingency, since all tasks are well detailed and the majority of resources needed are free. At the end, a generic tax of 21% has been added to the cumulative cost of the budget.

<table>
<thead>
<tr>
<th>Concept</th>
<th>Costs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Human resources</td>
<td>13,687 €</td>
</tr>
<tr>
<td>Material resources</td>
<td>92,11 €</td>
</tr>
<tr>
<td>Indirect costs</td>
<td>601,50 €</td>
</tr>
<tr>
<td>Contingency (5%)</td>
<td>719,1 €</td>
</tr>
<tr>
<td>Taxes (21% IVA)</td>
<td>3,170,93 €</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>18,270,64 €</strong></td>
</tr>
</tbody>
</table>

*Table 7 Estimated total budget (Source: Self-devised)*

5.1.5 Control management

During the project it is possible that some deviations occurs. The best action plan is to calculate the real deviation at the end of each task. With this control mechanism, we will be able to identify the actual budget spent for each task. At the end, we will calculate the real cost in hours, and compare them with the estimated hours. In case the deviation is quite big, we will need to make a study to understand why there has been so many changes, and how we can prevent them in future occasions.

Although we study the deviation, we do not expect that they are very large, as tasks are really well defined; That’s why we have just planned 5% of contingency. We don’t need a high budget on material resources, almost all part of the budget goes to human resources and 5% of contingency represents approx. 50h of extra working (for the computer engineer). In case there is a problem with the laptop, it will be replaced by a 3 years old PC that has similar specifications. Since that PC is quite old, the amortization costs won’t suppose a significant increase.
5.1.6 Corrections to the initial budget

Almost all the budget estimation corresponds to the initial budget calculated in GEP, but as the hours of working for the developer has been reduced, the budget has been updated. This update consists in recalculating the human resource budget, and consequently the final budget. This also implies to update the contingency and the taxes, which are calculated upon this final budget. Total budget has been reduced from 19,699,76 € to 18,270,64 €, making it even more competitive.

5.2 Sustainability

In this chapter the scores in Table 8, are explained, they correspond to the sustainability study made for this project. As we can see, the final score is 73,5 out of 90, a high mark that shows a good sustainability rate for this final degree project.

<table>
<thead>
<tr>
<th>Dimension</th>
<th>PPP</th>
<th>Useful life</th>
<th>Risks</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Economic</td>
<td>9/10</td>
<td>18/20</td>
<td>-4</td>
<td>23/30</td>
</tr>
<tr>
<td>Social</td>
<td>8,5/10</td>
<td>16/20</td>
<td>0</td>
<td>24,5/3</td>
</tr>
<tr>
<td>Environmental</td>
<td>9/10</td>
<td>18/20</td>
<td>-1</td>
<td>26/30</td>
</tr>
</tbody>
</table>

**Table 8 Sustainability matrix (Source: Self-devised)**

5.2.1 Economic dimension

This is a research oriented project and hence, does not represent a product to be sold directly. It is integrated in a bigger project still under development, so there is no need to evaluate the potential gains from the product. It will not be commercialized and the security policy obtained will be open source, it has scored 18/20 for its viability and just a -4 in its risk section for being a research project developed by a student.

Furthermore, the expenses estimated in the budget are competitive, that is why it has scored a 9 out of 10. We have to keep in mind that the project could be implemented in less time than planned, but just if the developer has had more experience and knowledge
about OpenSSL libraries. We have spent a lot of time on learning about this topic, if the 
developer has had more experience, his salary would be increased to be in concordance 
with his aptitudes and therefore, the human resources costs would be higher.

5.2.2 Social dimension

This project has been developed in Spain. Currently is a first world country but it has 
severe economic crisis. Research groups often do not have enough resources for its 
research topic. Luckily PRISTINE is funded by the EU so it makes it viable.

RINA does not impact directly to society but proposes an alternative to the current 
network problems. It helps to continue with the current growth in networking and 
provides more control and security. This project gives a secure module to RINA 
prototype, potential end-user would benefit from this feature. It does not harm any 
collective but if one day there is the need to change the current networking model, the 
industry in this field will need to adapt to this new paradigm. That is why it has scored 
16/20, not for its current use but for the big potential that has this innovative solution.

Personal impact has score 8.5, since it represents a big challenge for the developer and 
Improves her knowledge about the topic. It does not have risks in social dimension as it 
cannot harm any collective working on the prototype and even less to people who is not 
related to it.

5.2.3 Environmental dimension

This project does not suppose a big impact on the environmental field. Basically, the only 
hardware used is a Lenovo laptop. It has a special energy system program called Energy 
Star [15]. This program is designed to save money and protect the environment by 
implementing best practices and efficient products. Furthermore, the laptop will be used 
after the project and, at the end of the hardware’s lifespan, the components will be brought 
to a recycling centre so the harmful components can be treated appropriately.

According to official data [16], the estimated emission per Kw/h of greenhouse gas is 
about 248g of CO₂. Therefore, this project has an ecological footprint of ≈ 16,5Kg of 
CO₂. In the calculation process, it has been taking into account, the power spent during 
the implementation and documentation of the project. This dimensions scores 9 points as 
this project does not use extra tools and does not cause a big impact on the environment, 
and a 18/20 for its low footprint impact.
Its risk dimension has scored a -1/-20, as it is possible that there is the need to use another computer to develop or write the documentation of the project. In this case, the other laptop consumes a little bit more than the current laptop and the power consumption in the project would be increased. Event though, this increase would be almost negligible thus the risks score is almost 0.
6. Authentication modules in the RINA prototype

One of the first measures to implement for securing a distributed system is authentication. DIFs are securable containers, therefore in order to verify the identity of IPC Processes that want to join a DIF, proper authentication policies must be put in place. Such policies can range from no authentication to complex policies that exploit cryptographic techniques for more hostile environments. Even within a single DIF, different regions of the DIF may use distinct authentication policies depending on the properties.

Authentication is part of the Common Application Connection Establishment Phase (CACEP) that takes place between two IPCPs (and application processes in general) as illustrated in Figure 5. All the messages required for authentication are exchanged after the M_CONNECT message (which initiates the application connection setup procedure) and before the M_CONNECT_R message (which completes the application connection setup procedure). Google protocol buffers are used during CACEP and the exchange of the policy messages in order to serialize the data that must be sent.

In Figure 6, is shown at an abstract level the main application components that are related to the authentication procedures and the main interactions amongst them.
There are three main components that are relevant to an application’s authentication: The Security Manager, the RIB Daemon and the SDU Protection module.

- **SDU Protection module**: First of all, we define Service Data Unit (SDU) as the amount of data passed across the (N)-DIF interface to be transferred to the destination application process. The integrity of an SDU is maintained by the (N)-DIF. To protect/unprotect the data coming in/out an N-1 flow we have the SDU Protection module. It must be configured with the right policies and policy parameters (encryption algorithm, encryption key, etc.). The SDU Protection module configuration can be different for each different N-1 flow, and is owned by the Security Manager. The SDU Protection module can query a security profile to learn the operations that must be applied to incoming and outgoing SDUs.

- **RIB Daemon**: Receives incoming SDUs from SDU protection, which are CDAP messages targeting one or more RIB objects. The RIB Daemon is also the responsible for establishing an application connection to a remote. Before starting the application connection request, the RIB Daemon must query the Security Manager to obtain support of the relevant authentication policy module associated to the application connection. Any authentication-related messages received between M_CONNECT and M_CONNECT_R will be delivered to the authentication policy for its processing.

- **Security Manager**: Hosts all the authentication policy instances supported by the application, as well as the current security contexts (for each allocated N-1 flow). The authentication policy is in charge of initializing and populating the security profile associated with a particular N-1 flow with the relevant data (algorithms, key material,
The authentication policy interacts with the RIB Daemon to send/receive authentication-related messages.

7. TLS authentication policy

This project has been focused on certificate-based authentication. Certificate-based authentication relies on digital certificates, i.e. an electronic document that binds an identity with a public key. Regardless of its limitations, the use of certificates helps to prevent the use of fake public keys for impersonation. Digital certificates are issued by trusted third parties called Certificate Authorities (CAs), which sign all the certificates they issue with their private key. A Certificate Authority may be a Root CA - the root of trust for a particular chain - or may rely on an upstream CA which has previously delegated the rights to issue certificates to it. If so, validating a digital certificate requires validating the whole chain of certificates until the certificate of the root CA is reached (the root CA signs its certificate with its own private key, instead of having an upstream CA signing it). Expiration dates built-in the digital certificates and lists of certificates that have been revoked (certificate revocation lists) maintained by CAs, provide tools to minimize the lifetime of bogus or misused certificates.

The most popular application of digital certificate authentication is the Transport Layer Security (TLS) framework, heavily used by HTTPS to enable secure transactions over the web such as online banking. TLS is in fact two protocols, as said in section 2: the TLS Handshake protocol [12], which performs authentication and negotiates the algorithms and keys to be used by the TLS Record protocol [12] to perform encryption, compression and message integrity validation functions. In this project we have adapted the cryptographic operations and information exchange performed by the TLS Handshake protocol as an authentication policy for CACEP, thus allowing the use of a well-known certificate-based authentication procedure in the DAF/DIF environments. Cryptographic mechanisms in the TLS Record protocol have also been mined by a third party and adapted as SDU protection policies. There has been followed the schema of authentication modules in RINA, described in the previous chapter.
7.1 New authentication policy: AuthTLSHandshake

The handshake protocol consists of a sequence of messages sent between IPC processes. This protocol is used to negotiate the set of algorithms for authentication, confidentiality, compression and integrity; generate shared secrets, crypto keys and negotiate other parameters for the session. It allows peers to authenticate each other. At the end, these security parameters must be provided to the record layer, which is configured with the algorithms and keys negotiated by the handshake protocol. The differences of the current version of the CACEP authentication policy with respect to the normal TLS Handshake protocol are the following:

- No support of session resuming (useful feature foreseen for future extensions of the policy).
- No support for HelloRequest by "server".
- No support for extensions.
- Client is always obliged to provide its certificate for authentication.
- Limited availability of cipher suites. Cipher suites are the combination of authentication, encryption, message authentication code (MAC) and key exchange algorithms used to negotiate the security parameters for a connection using TLS protocol. In the first version there will be only support for one cipher suite to demonstrate the correct operation of the policy.
  - RSA (Rivest, Shamir and Adleman) [17] as key exchange algorithm (implies no need for ServerKeyExchange message). RSA is a public-key cryptography widely used for secure data transmission, it has two different but mathematically linked keys, one public and one private, the last one must be kept secretly. In RSA cryptography, both the public and the private keys can encrypt a message; the opposite key from the one used to encrypt the message is used to decrypt it.
  - RSA as a client certificate signing algorithm (no need for CertificateRequest message)
  - AES128 [18] or AES256 [18] as encryption algorithms (always used in hard cbc mode)
  - MD5 [19] or SHA256 [20] as message integrity verification algorithms (always used in HMAC [21] mode)
  - Deflate as the compression algorithm.
7.2 Specification

In this section is explained the specification of the TLS authentication policy and how he needed message have been designed.

When IPCP A first connects to IPCP B, it is required to send M_CONNECT (Client Hello) as its first message. With this message starts the application connection setup procedure, it contains the list of configuration options that IPCP A can support. It looks as seen below:

**IPCP A to IPCP B, M_CONNECT** (equivalent to Client Hello)

- **Name**: PSOC_authentication-tlshandshake
- **Versions**: 1 (only supported version as of now).
- **Options**:
  - *ap_con_id*: If not zero, ID of the application connection to be resumed.
  - *Ciphersuites*: The list of cipher suites supported by the requesting IPCP, sorted by preference.
  - *Compression*: The list of compression methods supported by the requesting IPCP, sorted by preference.
  - *UTCUnixTime*: The current time and date in standard UNIX 32-bit format.
  - *RandomBytes*: 28 bytes generated by a secure random number generator.

IPCP B receives the M_CONNECT message and decides if the authentication policy is correct. If it is, it selects a single cipher suite and compression method from the list provided by IPCP A. If none are supported or the application connection is rejected, IPCP
B sends an M_CONNECT_R indicating error. Otherwise it sends the following Server Hello message back to IPCP A:

**IPCP B to IPCP A, M_WRITE (Server Hello)**

- **Opcode**: M_WRITE
- **Object class**: Server Hello.
- **Object value**:
  - *Version*: The version of the policy chosen for the application connection
  - *ap_con_id*: The id of the application connection (for resuming the application connection)
  - *Ciphersuite*: The cipher suite chosen for the application connection
  - *Compression*: The compression method chosen for the application connection
  - *UTCUnixTime*: The current time and date in standard UNIX 32-bit format
  - *RandomBytes*: 28 bytes generated by a secure random number generator (independent from the ones generated by the client)

Once the negotiating process has ended, IPCP B needs to show its credentials to IPCP A, so that IPCP A can authenticate him. To do so, it sends its certificate chain. Certificate must be of type X.509v3, IPCP B’s certificate public key must be compatible with the key exchange algorithm. Normally there are multiple options available in TLS, we choose to stick to the use of RSA as the key exchange algorithm (the certificate must allow the key to be used for encryption). Its Certificate chain is sent back to the "client" using the following message:

**IPCP B to IPCP A, M_WRITE (Server certificate)**

- **Opcode**: M_WRITE
- **Object class**: Server Certificate.
- **Object value:**
  - *Certificate chain:* The certificate chain of IPCP B. First is the certificate of IPCP B, then all the certificates that certify each other until one that provides a root of trust within the DIF.

Since the only supported key agreement algorithm is RSA and the client always needs to provide a certificate, there is no need for the "ServerHelloDone" message. When IPCP A receives the "ServerCertificate" message, it can already send a "Certificate" message to IPCP B, since no more messages from IPCP B are expected. The certificate must be of type X.509v3. The public key of IPCP A’s certificate must be allowed to be used for signing with the signature scheme and hash algorithm that will be employed in the certificate verify message. The following diagram shows the next three messages that IPCP A needs to send to IPCP B.

![Diagram of next three messages](source: Self-devised)

---

**IPCP A to IPCP B, M_WRITE (Client Certificate)**

- **Opcode:** M_WRITE
- **Object class:** Certificate.
- **Object value:**
  - *Certificate chain:* The certificate chain of IPCP A. First is the certificate of IPCP A, then all the certificates that certify each other until one that provides a root of trust within the DIF.

After sending this message, IPCP A sends a ClientKeyExchange message. To do so, IPCP A generates 48 random bytes and encrypts them using the RSA public key, extracted from Server certificate message. Once IPCP B has received them, it must decrypt the message.
with its RSA private key. The message contains the RSA-encrypted pre-master secret and looks as follows:

**IPCP A to IPCP B, M_WRITE (ClientKeyExchange)**

- **Opcode**: M_WRITE
- **Object class**: Client Key Exchange.
- **Object value**:
  - *Encrypted pre-master secret*: The pre-master secret generated by IPCP A, encrypted with the public key of IPCP B’s certificate.

After this message, both parties have agreed upon the same pre-master secret. Then, both need to generate the master secret. The master secret is computed by using a PRF (pseudo random function) defined in RFC5246 [12], Chapter 5. It is based on HMAC, and always uses the SHA-256 hash algorithm. TLS’s PRF is created by applying P_HASH to the pre_master_secret as:

\[
\text{master\_secret} = \text{PRF}(\text{pre\_master\_secret}, \text{"master secret"}, \text{ClientHello.random + ServerHello.random})
\]

= P_hash(secret, label + seed)

The final formula is (where + indicates concatenation):

\[
P_{\text{hash}}(\text{secret}, \text{seed}) = \text{HMAC}\_\text{hash}(\text{secret}, A(1) + \text{seed}) +
\text{HMAC}\_\text{hash}(\text{secret}, A(2) + \text{seed}) +
\text{HMAC}\_\text{hash}(\text{secret}, A(3) + \text{seed}) + ...
\]

A() is defined as:

\[
A(0) = \text{seed}
A(i) = \text{HMAC}\_\text{hash}(\text{secret}, A(i-1))
\]

P_hash can be iterated as many times as necessary to produce the required quantity of data. For example, if P_SHA256 is being used to create 80 bytes of data, it will have to be iterated three times (through A(3)), creating 96 bytes of output data; the last 16 bytes of the final iteration will then be discarded, leaving 80 bytes of output data. In our case we iterate 2 times in order to obtain 64 bytes and we just store the first 48 bytes, since they correspond to our master secret length.

Both parties will compute the same master secret if the pre-master secret has been well decrypted. At this point IPCP B and IPCP A are able to compute the encryption keys. In the standard protocol, they should calculate two keys for encryption (Rx and Tx) and two for the HMAC hash (Rx and Tx) operations. But, by the end of this project, TLS record
module was not finished, so we proceed to implement the policy with just one encryption key to be able to test the policy with the existing record module. To calculate this key, we can use SHA256 or MD5 algorithm, as wished.

The next message is used to verify the certificate of IPCP A. All the ObjectValue field of all the messages exchanged between IPCP A and IPCP B (up to and without including this message), need to be hashed with the SHA-256 algorithm. Then, they will be signed with IPCP A’s private key and send it to IPCP B with the next message:

**IPCP A to IPCP B, M_WRITE (CertificateVerify)**

- ** Opcode: M_WRITE**
- **Object class: Certificate Verify**
- **Object value:**
  - Signed structure: Hashes of the ObjectValue field of all messages exchanged between both parties, signed with the private key of IPCP A.

IPCP B also needs to compute the hash of the ObjectValue field of all messages exchanged between IPCP A and IPCP B (up to and without including this message) with the SHA-256 algorithm. Once its computed and IPCP B has received the CertificateVerify message, it must decrypt it using IPCP A’s public key. If the authentication is done correctly, the computed and received hash should be the same. If not, an error message is generated (M_CONNECT_R with error code).

Once the security parameters are in place, IPCP A can send the ChangeCipherSpec message to signal IPCP B that this is the last message before starting to use the new cipher suite. This message consists on a single byte of value 1. It is used to notify the other party (IPCP B) to start using the policies agreed during the handshake. The reception of this message causes IPCP B to enable the new SDU protection policies on the reception path over the N-1 flow. Once this is done IPCP B sends the ChangeCipherSpec message to IPCP A. IPCP A, receives this message and enables the new SDU protection policies on both, the reception and transmission paths, over the N-1 flow. All subsequent messages are protected with the newly negotiated security parameters. IPCP B will enable the new policies on the transmission path when it receives the Finished message from IPCP A.

A Finished message is always sent after a ChangeCipherSpec message to verify that all SDU Protection policies are correctly in place. It contains the new configuration, and it
is the first one protected with the just negotiated algorithms, keys, and secrets. To generate the finish message, IPCP A uses the same PRF function as explained before with the following parameters:

**IPCP A to IPCP B, M_WRITE (FINISHED)**

- **Opcode**: M_WRITE
- **Object class**: Client Finish
- **Object value**:
  - **Verify data**: PRF(master_secret, "finish label", hash(object_value field of all handshake_messages)).

IPCP B also computes the PRF function with its parameters. Once IPCP B receives the verify data compares it with its generated verify data, they should be equal if the content is correct. If not, an error message is generated. If verify data is the same in both parties, IPCP B can send its *Finished* message to IPCP A following the same procedure.

At the end of the authentication process a M_CONNECT_R message must be send. This message indicates that the authentication process has ended successfully.

### 7.3 Implementation

In this subsection is described briefly how the previous specification has been implemented. The TLS authentication policy has been deployed in a common library, so that the policy can be used by an IPC Process but also by other application processes that
follow the DAF model. In this implementation, the IPC Process’s SDU Protection module is located at the kernel, while the RIB Daemon and the Security Manager are at user-space. Configuration of the SDU Protection module requires asynchronous messaging (via Netlink sockets, a socket addressed by process identifiers). This authentication policy uses OpenSSL’s libcrypto library as it needs to perform reliable cryptographic operations. The libcrypto facilities used in this policy are: HMAC, RAND_BYTES and SHA256 functions, load RSA keys from PEM files, RSA public key encryption and private key decryption and RSA private key encryption and public key decryption. When the communication is initiated, both client and server create a security context, it enables them to capture all relevant data and stores their certificates.

TLS Handshake authentication policy inherit from an abstract class named IAuthPolicySet. This class contains the name of the messages and the necessary function to allow authentication. As is an abstract class, it overwrites its pure virtual methods. One of the most important methods is get_auth_policy, is the first operation implemented and invoked by the Enrolment Task when it has to initiate the application connection. In this operation we obtain the values for the AuthPolicy field of the CDAP M_CONNECT message. It returns an AuthPolicy object.

Afterwards, the initiate_authentication policy checks for the correct policy names and version, selects the algorithms to be used for encryption, and sends the SERVER_HELLO and SERVER_CERTIFICATE messages with their corresponding fields.

Once this is done, is called the processIncoming_message_function whenever an authentication message is received. This function processes the different messages involved in this policy and does the following actions:

- **Server Hello message**: IPCP A stores cipher suites, random bytes, compression algorithms and version in its security context.
- **Server certificate message**: IPCP A stores IPCP B’s certificates for later use. Once it has received this message, IPCP A can proceed to send its credentials.
- **Client certificate message**: IPCP B stores IPCP A’s certificate.
- **Client key exchange**: After IPCP A generates the master secret and encrypts it using IPCP B’s RSA public key, IPCP B receives it and decrypts the pre-master secret with its private RSA key. Both generate the master secret using the PRF function explained in the policy design section, in this operation they also compute the encryption keys.
• **Client certificate verify message**: During all previous message exchanges both IPCPs have hashed the *ObjectValue* field of sent/received messages using the SHA256 algorithm. IPCP A has sent its verification hash signed with its RSA private key. In this operation IPCP B decrypts it using IPCP A’s public key, and compares the verification hash received with its own computation. If the authentication has been correct the verification hash must be equal.

• **Client Change Cipher Spec message**: When this message is received IPCP B requests to enable the negotiated SDU protection policies on the receive path for the related N-1 port in the kernel.

• **Server Change Cipher Spec message**: When this message is received, IPCP A requests to enable the negotiated SDU protection policies on the receive and transmit paths for the related N-1 port in the kernel.

• **Client Finished message**: IPCP B calculates the verification data and compares it with the received one. If they are equal, the procedure has been successful. Then, IPCP B request to enable the negotiated SDU protection policies on the transmit path for the related N-1 port in the kernel.

• **Server Finished message**: IPCP A compares the received verification data with its own computation. If the result is equal, the operation returns SUCCESS, at this point, encrypted data can be sent over the connection.
8. Validation and performance analysis

In this section we present the validation process of the implementation of the handshake protocol of the TLS authentication module. First property is reliability, which means that the components of the experiment remain functional during all the elaboration of the project and the results and outcomes are reliable. The next property is repeatability, meaning that different runs of the experiment yield the same outcome and results. The last one is reproducibility, namely, the same experiment can be reproduced in different platforms.

To achieve repeatability there has been run several times the same experiment in the virtual machines, observing with tcpdump Linux utility the authentication packets send and received, and realizing, after the procedure is done, that all the following packets are encrypted (log of an IPCP shown in appendix D). Afterwards, to guarantee the reliability property, there has been generated traffic between IPCP (explained in section 8.5) confirming that all data sent or received in the established connection remains encrypted. To accomplish reproducibility property, the experiment has been repeated in two different machines that supports RINA placed in a data centre, and the authentication process obtained was the same as if we have run the same module in the virtual machines.

8.1 Authentication policy verification scenario

The experimental scenario used to verify the correct operation of the AuthTLSHandshake policy is shown in the verification scenario shown in Figure 11. A normal DIF consisting of two IPCPs operates over one shim DIF over Ethernet. IPCP test1.IRATI and IPCP
test2.IRATI are configured to use the AuthTLSHandshake policy by default, with encryption, Error Check (CRC) and TTL policies.

8.2 Configuration

The following code snippet shows an example of the AuthTLSHandshake policy configuration, as well as of the associated encryption policy that must be activated for N-1 port. The authentication policy needs to be populated with the location of the certificates files and the RSA private key.

```json
"securityManagerConfiguration" : {
  "policySet" : {
    "name" : "default",
    "version" : "1"
  },
  "authSDUProtProfiles" : {
    "default" : {
      "authPolicy" : {
        "name" : "PSOC_authentication-tlshandshake",
        "version" : "1",
        "parameters" : [ {
          "name" : "keystore",
          "value" : "/usr/local/irati/etc/creds-tls"
        }, {
          "name" : "keystorePass",
          "value" : "none"
        } ]
      },
      "encryptPolicy" : {
        "name" : "default",
        "version" : "1",
        "parameters" : [ {
          "name" : "encryptAlg",
          "value" : "AES128"
        }, {
          "name" : "macAlg",
          "value" : "SHA256"
        }, {
          "name" : "compressAlg",
          "value" : "deflate"
        } ]
      },
      "TTLPolicy" : {
        "name" : "default",
        "version" : "1",
        "parameters" : [ {
          "name" : "initialValue",
          "value" : "50"
        } ]
      },
      "ErrorCheckPolicy" : {
        "name" : "CRC32",
        "version" : "1"
      }
    }
  }
}
```
The files contained by the folder provided as the value as the "keystore" parameter have to be arranged in the following way:

- **key**: contains the RSA private key of the IPCP, in PEM format.
- **cert.pem**: contains the certificate of the IPCP, in PEM format.
- **<cert_name>.pem**: one PEM file for every certificate required to reach the (or one of the) root of trust of the DIF (root CA).

### 8.3 Certificate generation

One of the first steps to run TLS authentication is to create the certificates needed for the policy. Although there can be used other trustier certificates we explain one way to generate them for testing using self-signed certificates (this should never be done in a production scenario). This method consists on creating a ROOT Certificate Authority (CA) from an OpenSSL’s default script and two self-signed certificates from OpenSSL commands.

We will work from the directory: `/etc/ssl` We move the CA.pl to the current directory and execute `./CA.pl -newca` in order to start creating the root certificate. After this command, we have generated a public-private key pair. We can find the public key at `/etc/ssl/ca/cacert.pem` and the private key at `/etc/ssl/ca/private/cakey.pem`. To obtain this CA certificate is necessary to set a password, as it is needed for signing the end-user certificate.

```
root@debian:/etc/ssl# ./CA.pl -newca
CA certificate filename (or enter to create)

Making CA certificate ...
Generating a 2048 bit RSA private key
........+++ ..................................+++
writing new private key to './demoCA/private/cakey.pem'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
```
What you are about to enter is what is called a Distinguished Name or a DN.

There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.

-----
Country Name (2 letter code) [AU]: SP
State or Province Name (full name) [Some-State]: Barcelona
Locality Name (eg, city) []: Barcelona
Organization Name (eg, company) [Internet Widgits Pty Ltd]: i2cat
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []: CAROOT
Email Address []:

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:
An optional company name []:
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok

Certificate Details:
  Serial Number: 17273132791151573382 (0xefb67ec1aebdb986)
  Validity
    Not Before: Jun 1 11:33:08 2016 GMT
    Not After : Jun 1 11:33:08 2019 GMT
  Subject:
    countryName               = SP
    stateOrProvinceName       = Barcelona
    organizationName          = i2cat
    commonName                = CAROOT
  X509v3 extensions:
    X509v3 Subject Key Identifier:
    X509v3 Authority Key Identifier:
  X509v3 Basic Constraints:
    CA:TRUE

Certificate is to be certified until Jun 1 11:33:08 2019 GMT (1095 days)

Write out database with 1 new entries
Data Base Updated

CA will use a CSR (certificate signing request) to create our SSL certificate. It contains information that will be included in the end-user certificate such as organization name, location and country. It also contains the public key that will be included in the end-user certificate. At the same time, we create the private key of the end-user certificate, we need to keep the privat key "secret". To do so we have used the following command, the key generated is a RSA key of 2048 bits:
root@debian:/etc/ssl# openssl req -nodes -newkey rsa:2048 -keyout demoCA/private/cert1key.pem -out demoCA/certs/cert1csr.pem
Generating a 2048 bit RSA private key
......................................+++ ......+++
writing new private key to 'demoCA/private/cert1key.pem'
-----
You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:SP
State or Province Name (full name) [Some-State]:Barcelona
Locality Name (eg, city) []:Barcelona
Organization Name (eg, company) [Internet Widgits Pty Ltd]:i2cat
Organizational Unit Name (eg, section) []:
Common Name (e.g. server FQDN or YOUR name) []:test1.IRATI
Email Address []:

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []:
An optional company name []:

This creates a new CSR (in etc/ssl/demoCA/certs/cert1csr.pem) which must then be signed using the CA’s private key, and a private key (in /etc/ssl/ca/private/cert1key.pem).
Last step is to create and sign the end-user certificate. We will take the input CSR (demoCA/certs/cert1csr.pem) and create the end user certificate (in demoCA/certs/cert1.pem). The command line used is:

root@debian:/etc/ssl# openssl ca -policy policy_anything -in demoCA/certs/cert1csr.pem -out demoCA/certs/cert1.pem
Using configuration from /usr/lib/ssl/openssl.cnf
Enter pass phrase for ./demoCA/private/cakey.pem:
Check that the request matches the signature
Signature ok
Certificate Details:
Serial Number: 17273132791151573383 (0xefb67e1ae6db987)
Validity
Not Before: Jun 1 11:37:46 2016 GMT
Not After : Jun 1 11:37:46 2017 GMT
Subject:
countryName = SP
stateOrProvinceName = Barcelona
localityName = Barcelona
organizationName = i2cat
commonName = test1.IRATI

X509v3 extensions:
X509v3 Basic Constraints:
CA:FALSE
Netscape Comment:
This resulting certificate will be the one used for the policy. Then we will need a second certificate for IPCP B. We will follow the same procedure, creating a new CSR and a new end-user certificate to obtain the second certificate.

8.4 Traces of the verification experiments

The traces attached in Appendix D are the output of capturing the Ethernet packets at the eth1.110 interfaces in system 1, with the Linux utility tcpdump. The first one correspond to the ARP request and reply issued by the shim DIF when the IPC Process test1.IRATI request a flow allocation to the IPC Process test2.IRATI.

The following trace reflects test1.IRATI sending an M_CONNECT messages to test2.IRATI, requesting a new connection to be opened using the "PSOC_authentication-tls handshake" authentication policy.

IPCP test2.IRATI replies with the Server Hello and Server Certificate message. Now, test1.IRATI sends its client certificate message, client key exchange message and client certificate verify message. The last non-encrypted message is client change cipher spec message, and its corresponding response, server change cipher spec as we can see in appendix D.

From now on, all messages are encrypted, as shown by the trace of the Finished messages. Since the communication is encrypted, showing the log of tcpdump of the following packets is not very illustrative. The last trace, shows the log of IPCP test1.IRATI (the one that initiated the application connection). The sequence of messages shows how
test1.IRATI sent and received the messages, until it receives an M_CONNECT_R message indicating that the application connection has been successfully established.

8.5 Performance analysis

In this section we will analyse the performance of this new policy and compare it with the use of no authentication. The results obtained in the measurement of the enrolment time are very variable, we have computed the average with 10 samples. It takes about **16.4 ms** for TLS handshake authentication policy to enrol, a not very significant increase over the **8 ms** of the use of no authentication - which involves no cryptographic operations and 8 messages less to exchange. The reason is that both machines are virtual machines co-located in the same physical machine, therefore since the RTT is very low the cost of exchanging a message is also low. With longer RTTs the cost of message exchanges will dominate over the cost of cryptographic operations, increasing the enrolment time in relation to that of a policy with no authentication.

The second set of measurements we have taken consist on calculating the round trip time (RTT) of an application over an encrypted N-1 flow. To do so, we have used a script (configuration explained in appendix C.1) called rina-echo-time that pings the other IPCP when echoed back to the source, computes the minimum, maximum and average time, as well as the standard deviation, we show the results in the Figure 12 and 13:

<table>
<thead>
<tr>
<th>Packets size</th>
<th>Minimum RTT</th>
<th>Maximum RTT</th>
<th>Standard Deviation</th>
<th>Average RTT</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.001361 ms</td>
<td>12.517 ms</td>
<td>0.53555 ms</td>
<td><strong>0.51517 ms</strong></td>
</tr>
<tr>
<td>10</td>
<td>0.00056 ms</td>
<td>36.805 ms</td>
<td>0.74741 ms</td>
<td><strong>0.52448 ms</strong></td>
</tr>
<tr>
<td>100</td>
<td>0.001521 ms</td>
<td>13.892 ms</td>
<td>0.47448 ms</td>
<td><strong>0.5837 ms</strong></td>
</tr>
<tr>
<td>1000</td>
<td>0.000934 ms</td>
<td>46.191 ms</td>
<td>1.0616 ms</td>
<td><strong>0.62628 ms</strong></td>
</tr>
<tr>
<td>1450</td>
<td>0.000214 ms</td>
<td>51.892 ms</td>
<td>1.1071 ms</td>
<td><strong>0.68921 ms</strong></td>
</tr>
</tbody>
</table>

*Figure 12 RTT of no authentication policy (Source: Self-devised)*
Figures above are computed sending each time 10000 packets of variable size. The maximum size established has been 1450 bytes for packet since it is the maximum size to avoid fragmentation; a feature not supported yet for RINA prototype.

We can observe that the RTT is just moderately impacted by the use of the TLS handshake and associated SDU protection policies. It has always a higher RTT due to the cryptographic operations, with more data to encrypt, more operations must be done. We can see the progression in the graphic below, Figure 14:

To conclude this section, we will show the throughput study results in Figure 15 and 16. We have used the traffic generator application to obtain them (configuration in appendix C.2). This application generates traffic during a specific period of time with a predefined packets size. Our period of time will be 20 second and the same sizes as in the RTT analysis. As we increase the size of the packets we observe that the overhead of performing cryptographic operations per packet compromises significantly the throughput.
This result has been obtained in different machines, in this way the performance is not limited by the use of virtual machines. As we have said before, the use of encryption compromises the throughput significantly, to clarify and see it in a more visual way we attached the graphic showing the result obtained when traffic is generated.
9. Conclusions and future work

In this final section is explained what this project has brought, not just the results obtained, but the experience gained from it.

Working in RINA experimental field has been an enriching experience. At the beginning was difficult to familiarize and understand the model. It is a prototype with lots of new concept never done before, with a large own vocabulary. Afterwards, there was a lot of documentation phase about how authentication protocols and specifically TLS works. It was a challenging task that was worthy to experiment. Another uphill was to set all the RINA environment and get to know the code project. Even though, this has set another perspective of how to face a problem; instead of set a fixed point of view and work from it, there is the possibility of change the perspective and try to solve the problem from its basis.

The results obtained from this project are satisfactory. The initial baseline of creating an authentication policy based on certificate based protocol has been achieved. An IPCP is able to enrol to another IPCP using the TLS handshake policy in a correct way. The downside of using this policy is that the performance is reduced, although this is not a problem of this policy, since all security protocols cause an overhead in the communication.

In a future extension of this project the first thing to do is to set the corresponding encryption keys. Currently we pass a single encryption key to the record protocol (located in the kernel), calculated using the PRF function with the master secret. The TLS record module, as is not finished yet (made by other party), uses this key to encrypt for both receiving and sending (Rx, Tx) channels. In the project we have adapted the PRF function to be able to calculate the future keys. The extension would consist on calculate one key for the received data, one for the sending data and two more for the HMAC field that TLS record module has. Once they are generated they should be sent to the record layer. This other module, would operate with them in order to achieve a TLS policy as much similar as the one that the Internet uses.

Another improvement would be to support session resuming and Hello requests from the server side. Furthermore, more extensions, cipher suites and key exchange algorithm could be supported. This features would be implemented when needed, for a first
approach on certificate based authentication policy it has not been necessary to implement them.

Deciding to use more or less secure authentication policies depends on the scenario we are targeting. Depending on the performance we want to obtain, for example a really high throughput for a determined experiment, we should not use TLS authentication. Nonetheless if the data that we are transferring can be sensible, we should use it, never forgetting that, nowadays, personal data is a powerful currency in the world.
10. Bibliography


[16] “Greenhouse gas emissions calculator”  


[22] “Wikipedia: RINA”.  


APPENDIX
A. Configuration of the system

To be able to run RINA prototype we need to install this following packages in a GNU/Linux OS platform. Our project runs under Debian OS.

Kernel packages:

- kernel-package
- libncurses5-dev (needed for make menuconfig)

For the user-space parts, the following packages from the default repository are required:

(we can get them with the command: sudo apt-get install name_of_the_package)

- autoconf
- automake
- libtool
- pkg-config
- git
- g++
- openjdk-6-jdk
- maven
- libssl-dev
- openSSL

We also need additional packages from the testing repository. This repository contains all RINA prototype and its features, not only the authentication policy made in this project.

- Add the testing repository
  - Add deb http://ftp.de.debian.org/debian jessie main in /etc/apt/sources.list
  - Run apt-get update
- protobuf-compiler, libprotobuf-dev (version >= 2.5.0 required)
- libnl-genl-3-dev and libnl-3-dev (version >= 3.2.14 required):

After this installation we need to build the following packages from source:

- SWIG (version >= 2.0.8 required, 2.0.12 is known to be working fine)
  - From http://swig.org
Once all packages are ready we can proceed to install RINA prototype. First step is cloning its repository:

- `git clone https://github.com/IRATI/stack.git`

Now we need to make sure that we have the latest tags of the repository by typing `git pull --tags`.

Then we need to run `make menuconfig` in the Linux directory. The RINA parts can be found at:

- Networking support
  - Rina support

To install both the kernel and user space, you can use the `install-from-scratch` script. The script takes one parameter, the location where to install the user-space parts; if left empty, they will be installed in the root of the filesystem. Installation will take a while, since a Linux kernel has to be compiled and the user-space parts have to be installed as well. After RINA prototype has been installed, reboot the machine. You can use `uname -a` to check if the new kernel was loaded, and you can browse to the folder where you installed the user-space parts to verify they are there.

Alternatively, you can install the kernel manually. To compile and install the kernel (and its headers), do the following:

- `make headers_install`
- `make bzImage modules`
- `make modules_install install`

Afterwards, run the `install-user-from-scratch` script to install the user-space parts. If you have already compiled and installed the user-space ones and you want to be able to just build/install librina, rinad or rina-tools separately, you can do the following:

- From the main folder of your local repository, type `cd build`
- Then `cd librina` or `cd rinad` or `cd rina-tools`
- Then `make install`
B. User’s guide

As explained in subsection 2.3 we need to have two computers (or virtual machines) that runs under Linux OS, once we have provided them with proper packages and configurations we can test our policy.

First of all, we need to add the necessary modules in our systems to be able to init RINA daemon. We have created a script that runs every time we turn on our virtual machines. It contains the following commands:

```bash
modprobe 8021q
ifconfig eth1 up
vconfig add eth1 110
ifconfig eth1.110 up
modprobe shim-eth-vlan
modprobe shim-tcp-udp
modprobe rina-default-plugin
modprobe normal-ipcp
```

Once we have run the script we are able to initiate RINA daemon. To do it, we need to run the following command in the location /IRATI/bin as following:

```bash
root@debian:/home/berta/IRATI/bin# ./ipcm -c ../etc/ipcm.conf -l DEBUG
```

The ipcm.conf file is where the configurations of the IPC process is defined. In the following figure we can see how it the IPC manager looks like:
As we can see in the last lines it creates a normal DIF that use the tls.dif template. This template is used to tell the IPC which authentication policy needs to use, in this case TLS authentication policy. This code snippet shows an example configuration of the security manager:

```json
root@debian:/home/berta/IRATI/etc# more ipcm.conf
{
    "configFileVersion" : "1.4.1",
    "localConfiguration" : {
        "installationPath" : "/home/berta/IRATI/bin",
        "libraryPath" : "/home/berta/IRATI/lib",
        "logPath" : "/home/berta/IRATI/var/log",
        "consoleSocket" : "/home/berta/IRATI/var/run/ipcm-console.sock",
        "pluginsPaths" : ["/home/berta/IRATI/lib/rinad/ipcp"]
    },
    "ipcProcessesToCreate" : [ {
        "type" : "shim-eth-vlan",
        "apName" : "test-eth-vlan",
        "apInstance" : "1",
        "difName" : "110"
    }, {
        "type" : "normal-ipc",
        "apName" : "test2.IRATI",
        "apInstance" : "1",
        "difName" : "normal.DIF",
        "difsToRegisterAt" : ["110"]
    } ],
    "difConfigurations" : [ {
        "name" : "110",
        "template" : "shim-eth-vlan.dif"
    }, {
        "name" : "normal.DIF",
        "template" : "tls.dif"
    } ]
}
```

*Figure 18 IPC manager configuration*
"securityManagerConfiguration": {
  "policySet": {
    "name": "default",
    "version": "1"
  },
  "authSDUProtProfiles": {
    "default": {
      "authPolicy": {
        "name": "PSOC_authentication-tlsHandshake",
        "version": "1",
        "parameters": [
          {
            "name": "cipherSuite",
            "value": "TODO"
          },
          {
            "name": "compressionMethod",
            "value": "TODO"
          },
          {
            "name": "keyStore",
            "value": "jore"
          },
          {
            "name": "myCredentials",
            "value": "/home/berta/Escritorio/Certificats_publics3"
          },
          {
            "name": "myPrivKey",
            "value": "/home/berta/Escritorio/Certificats_publics3/certkey.pem"
          },
          {
            "name": "keyStorePass",
            "value": "nore"
          }
        ]
      },
      "encryptPolicy": {
        "name": "default",
        "version": "1",
        "parameters": [
          {
            "name": "encryptAlg",
            "value": "AES128"
          },
          {
            "name": "macAlg",
            "value": "SHA1"
          },
          {
            "name": "compressAlg",
            "value": "default"
          }
        ]
      },
      "TTLPolicy": {
        "name": "default",
        "version": "1",
        "parameters": [
          {
            "name": "initialValue",
            "value": "50"
          }
        ]
      },
      "ErrorCheckPolicy": {
        "name": "CRC32"
      }
    }
  }
}
"version": "1"
}
}

Figure 19 TLS.dif configuration

After that we will have the Rina daemon working on background. Now, we will open a new terminal and writes the command similar to:

```
socat-UNIX:/home/berta/IRATI/var/run/ipcm-connsole.sock
```

This will create a socket, we will use it in order to enrol IPCP B in IPCP ‘s A DIF. Now we can proceed to enrol IPCP B using this command:

```
USAGE: enroll-to-dif <ipcp-id> <dif-name> <supporting-dif-name> 
<neighbor-process-name><neighbor-process-instance>
```

In a correct enrolment we can see in the terminal something like this example:

```
IPCM >>> enroll-to-dif 2 normal.DIF test1.IRATI 1
DIF enrollment succesfully completed
```

Afterwards we have already verify the identity using TLS handshake policy of an IPC process that want to join a DIF.
C. Performance analyses

C.1 Round Trip Time

In order to get the RTT we make a ping to the other IPCP, there is an existing script in RINA repository (located in IRA/II/bin) where it send a controlled amount of packets and computes the average, minimum and maximum RTT. To do so, we need to configure an IPCP as a server(listener) using:

```
./rina-echo-time -l
```

The other IPC is the one making the PING and calculating the times. We use the same script as the the “server”, but with other option, we can use -h in order to obtain help.

One good example of use is:

```
./rina-echo-time -w 0 -s 1450 -c 10000
```

Where -c represent time to wait between packages, -s the size of the packages and -c being the number of packets to send. The output generated looks like:

- SDUs sent: 10000; SDUs received: 10000; 0% SDU loss
- Minimum RTT: 0.002613 ms; Maximum RTT: 12.194 ms;

C.2 Throughput

To analyse the throughput, we need to run another script. Unfortunately, this script is not included in RINA repository and we need to get it from https://github.com/IRATI/traffic-generator To install it we need to do some procedures showed below:

```
cd traffic-generator
./bootstrap
apt-get install libboost-all-dev
export LD_LIBRARY_PATH=/home/berta/IRATI/lib/
./configure --prefix=/home/berta/IRATI
make install
```

As in the other script we need to execute one IPC process using:

```
./rina-tgen -l
```
The other IPC acts sends packet during a period of time of a defined size. We can check more options in -h parameters. In the following line we have an example:

```
./rina-tgen --duration 20 -s
```
D. Traces

ARP request and response

09:48:33.467682 08:00:27:4d:bf:88 (oui Unknown) > Broadcast, ethertype 802.1Q (0x8100), length 68:

0x0000: 0001 d1f0 060f 0001 0800 274d bf88 7465 ..........'M..te
0x0010: 7374 312e 4952 4154 492f 312f 02ff 0fff st1.IRATI/1/..
0x0020: 02ff ff74 6573 7432 2e49 5241 5449 2f31 ...test2.IRATI/1
0x0030: 2f2f //

09:48:33.468124 08:00:27:48:d1:bf (oui Unknown) > 08:00:27:4d:bf:88 (oui Unknown), ethertype 802.1Q (0x8100), length 68:

0x0000: 0001 d1f0 060f 0002 0800 2748 d1bf 7465 ..........'H..te
0x0010: 7374 322e 4952 4154 492f 312f 0827 4dbf st2.IRATI/1/..
0x0020: 8874 6573 7431 2e49 5241 5449 2f31 M..test1.IRATI/1
0x0030: 2f2f //

M_CONNECT message

09:48:33.468707 08:00:27:4d:bf:88 (oui Unknown) > 08:00:27:48:d1:bf (oui Unknown), ethertype 802.1Q (0x8100), length 237:

0x0000: 3200 0000 0000 0000 0100 0002 0001 0000 2...............m
0x0010: 0000 0040 00cf 0000 0000 0008 7310 0108 ...@........s...
0x0020: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x0030: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x0040: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x0050: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x0060: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x0070: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x0080: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x0090: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x00a0: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x00b0: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x00c0: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x00d0: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x00e0: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
0x00f0: 012a 0032 0038 0048 0050 0092 015f 0a20 .*.2.8.H.P..._..
Server Hello and Server certificate messages

<table>
<thead>
<tr>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:48:33.468707</td>
<td>08:00:27:4d:bf:88</td>
<td>08:00:27:48:d1:bf</td>
<td>oui Unknown</td>
<td>237</td>
<td>Management...</td>
</tr>
<tr>
<td>09:48:33.475161</td>
<td>08:00:27:4d:bf:88</td>
<td>08:00:27:48:d1:bf</td>
<td>oui Unknown</td>
<td>1121</td>
<td>Management...</td>
</tr>
</tbody>
</table>

Client certificate message

<table>
<thead>
<tr>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Payload</th>
</tr>
</thead>
<tbody>
<tr>
<td>09:48:33.475161</td>
<td>08:00:27:4d:bf:88</td>
<td>08:00:27:48:d1:bf</td>
<td>oui Unknown</td>
<td>1121</td>
<td>Management...</td>
</tr>
</tbody>
</table>
Client key exchange message

09:48:33.476260  08:00:27:4d:bf:88 (oui Unknown) > 08:00:27:48:d1:bf (oui Unknown), ethertype 802.1Q (0x8100), length 403:

0x0000:  3200 0000 0000 0000 0100 0002 0001 0000  2............
0x0010:  0000 0040 0075 0100 0000 0008 0010 0c18  ...@.u........
0x0020:  002a 1343 6c69 656e 7420 6b65 7920 6578  .*.Client.key.ex
0x0030:  6368 616e 6765 3213 436c 6965 6e74 206b  change2.Client.k
0x0040:  6f6e 3a00 206f 0040 0075 0100 0000 0008  6o..0.0075 0100 0000 0008 0010 0c18  ...@.u........
0x0050:  002a 1343 6c69 656e 7420 6b65 7920 6578  .*.Client.key.ex
0x0060:  6368 616e 6765 3213 436c 6965 6e74 206b  change2.Client.k
0x0070:  6f6e 3a00 206f 0040 0075 0100 0000 0008  6o..0.0075 0100 0000 0008 0010 0c18  ...@.u........

Client certificate verify message

09:48:33.483746  08:00:27:4d:bf:88 (oui Unknown) > 08:00:27:48:d1:bf (oui Unknown), ethertype 802.1Q (0x8100), length 415:
Client and server change cipher spec messages

09:48:33.484267 08:00:27:4d:bf:88 (oui Unknown) > 08:00:27:48:d1:bf (oui Unknown), ethertype 802.1Q (0x8100), length 155:
0x0000: 3200 0000 0000 0001 0100 0002 0001 0000 2............
0x0010: 0000 0040 0081 0100 0000 0008 0010 0c18 ........@........
0x0020: 002a 1943 6c69 656e 7432 0635 7269 6679 3219 436c  ".Client.certificate.verify2.Client.certificat
0x0030: 6965 6e74 2063 6861 6e67 6520 7370 6563 3219 5365  ".identification.cipher.spec2.Server.certifi
0x0040: 7276 6577 7e66 696e 676d 6572 2063 6861 6e67 6520  ".server.change.
0x0050: 5000 9201 020a 009a 0100 a201 00aa 0100 P............
0x0060: 0070: b201 00ba 0100 c201 00ca 0100 d201 00da ..............
0x0080: 0100 e001 004b 852d ae .....K.-.
09:48:33.486089 08:00:27:48:d1:bf (oui Unknown) > 08:00:27:4d:bf:88 (oui Unknown), ethertype 802.1Q (0x8100), length 155:
0x0000: 3200 0000 0000 0001 0100 0003 0001 0000 2............
0x0010: 0000 0040 007d 0000 0000 0008 0010 0c18 ........@}........
0x0020: 002a 1953 6572 7e66 696e 676d 6572 2063 6861 6e67 6520  ".Server.certificat.verify2.Server.certificat.
0x0030: 5000 b201 00ba 0100 c201 00ca 0100 d201 00da ..............
0x0040: 0100 e001 004b 852d ae .....K.-.
Encrypted client and server finished messages

09:48:33.48267 08:00:27:4d:bf:88 (oui Unknown) > 08:00:27:48:d1:bf (oui Unknown), ethertype 802.1Q (0x8100), length 155:
0x0000: 3200 0000 0000 0000 0100 0002 0001 0000 2……………. 
0x0010: 0000 0040 007d 0000 0000 0008 0010 0c18 ...@.}……………. 
0x0020: 002a 1943 6c69 656e 7420 6368 616e 6765 .*.Client.change 
0x0030: 2063 6970 6865 7220 7370 6563 3219 436c .cipher.spec2.Cl 
0x0040: 6965 6e74 2063 6861 6e67 6520 6369 7068 ient.change.ciph 
0x0050: 6572 2073 7065 6338 0042 0332 0108 4800 er.spec8.B.2..H. 
0x0060: 5000 9201 020a 009a 0100 a201 00aa 0100 P……………. 
0x0070: b201 00ba 0100 c201 00ca 0100 d201 00da ………………. 
0x0080: 0100 e001 00be dce0 1b ……….. 

09:48:33.486089 08:00:27:48:d1:bf (oui Unknown) > 08:00:27:4d:bf:88 (oui Unknown), ethertype 802.1Q (0x8100), length 155:
0x0000: 3200 0000 0000 0000 0100 0003 0001 0000 2……………. 
0x0010: 0000 0040 007d 0000 0000 0008 0010 0c18 ...@.}……………. 
0x0020: 002a 1953 6572 7652 7665 7220 6368 616e 6765 .*.Server.change 
0x0030: 2063 6970 6865 7220 7370 6563 3219 5365 .cipher.spec2.Se 
0x0040: 7276 6572 2063 6861 6e67 6520 6369 7068 rver.change.ciph 
0x0050: 6572 2073 7065 6338 0042 0332 0108 4800 er.spec8.B.2..H. 
0x0060: 5000 9201 020a 009a 0100 a201 00aa 0100 P……………. 
0x0070: b201 00ba 0100 c201 00ca 0100 d201 00da ………………. 
0x0080: 0100 e001 00be dce0 1b ………..
IPCP test1.IRATI log

9094(1464768076)#librina.syscalls (DBG): Invoking SYS_writeManagementSDU (330)
9094(1464768076)#ipcp[2].rib-daemon (DBG): Sent CDAP message of size 188 through port-id 2:
Opcode: 0_M_CONNECT
Abstract syntax: 115
Authentication policy: Policy name: PSOC_authentication-tlshandshake
Supported versions: 1;

Source AP name: test1.IRATI
Source AP instance: 1
Source AE name: Management
Destination AP name: test2.IRATI
Destination AP instance: 1
Destination AE name: Management
Invoke id: 1
Version: 1

9094(1464768076)#cdap (DBG): Waiting timeout 10000 to receive a connection response
9094(1464768076)#rib (DBG): Starting add object operation over RIB(0x20c7e30), of object(0x20cc3f0) with fqn: '/rmt/n1flows/port_id=2' (parent '/rmt/n1flows')
9094(1464768076)#rib (DBG): Add object operation over RIB(0x20c7e30), of object(0x20cc3f0) with fqn: '/rmt/n1flows/port_id=2', succeeded. Instance id: '33'
9094(1464768076)#ipcp[2].routing-ps-link-state (DBG): flow allocation waiting for enrollment
9094(1464768076)#ipcp[2].rib-daemon (DBG): Received CDAP message from N-1 port 2
Opcode: 12_M_WRITE
Object class: Server Hello
Object name: Server Hello
Scope: 0
9094(1464768076)#librina.syscalls (DBG): Invoking SYS_readManagementSDU (329)
9094(1464768076)#ipcp[2].rib-daemon (DBG): Received CDAP message from N-1 port 2
Opcode: 12_M_WRITE
Object class: Server Certificate
Object name: Server Certificate
Scope: 0

9094(1464768076)#librina.syscalls (DBG): Invoking SYS_writeManagementSDU (330)
9094(1464768076)#ipcp[2].rib-daemon (DBG): Sent CDAP message of size 1072 through port-id 2:
Opcode: 12_M_WRITE
Object class: Client Certificate
Object name: Client Certificate
Scope: 0

9094(1464768076)#librina.syscalls (DBG): Invoking SYS_writeManagementSDU (330)
9094(1464768076)#ipcp[2].rib-daemon (DBG): Sent CDAP message of size 354 through port-id 2:
Opcode: 12_M_WRITE
Object class: Client key exchange
Object name: Client key exchange
Scope: 0

9094(1464768076)#librina.tls-handshake (DBG): Generated encryption key of length 16 bytes: 1fa1e09e799f5951fc368e61a11330e6
9094(1464768076)#librina.syscalls (DBG): Invoking SYS_writeManagementSDU (330)
9094(1464768076)#ipcp[2].rib-daemon (DBG): Sent CDAP message of size 366 through port-id 2:
  Opcode: 12_M_WRITE
  Object class: Client certificate verify
  Object name: Client certificate verify
  Scope: 0

9094(1464768076)#librina.syscalls (DBG): Invoking SYS_writeManagementSDU (330)

9094(1464768076)#ipcp[2].rib-daemon (DBG): Sent CDAP message of size 106 through port-id 2:
  Opcode: 12_M_WRITE
  Object class: Client change cipher spec
  Object name: Client change cipher spec
  Scope: 0

9094(1464768076)#librina.syscalls (DBG): Invoking SYS_readManagementSDU (329)

9094(1464768076)#ipcp[2].rib-daemon (DBG): Received CDAP message from N-1 port 2
  Opcode: 12_M_WRITE
  Object class: Server change cipher spec
  Object name: Server change cipher spec
  Scope: 0

9094(1464768076)#ipcp (DBG): Requesting the kernel to update crypto state on port-id: 2

9094(1464768076)#librina.nl-manager (DBG): NL msg RX. Fam: 25; Opcode: 42_UPDATE_CRYPTO_STATE_RESP; Sport: 0; Dport: 9094; Seqnum: 1464768070; Response; SIPCP: 2; DIPCP: 0

9094(1464768076)#librina.nl-manager (DBG): NL msg TX. Fam: 25; Opcode: 41_UPDATE_CRYPTO_STATE_REQ; Sport: 9094; Dport: 0; Seqnum: 1464768070; Request; SIPCP: 2; DIPCP: 2
added event of type 41_UPDATE_CRYPTO_STATE_RESPONSE and sequence number 1464768070 to events queue

Got event of type 41_UPDATE_CRYPTO_STATE_RESPONSE and sequence number 1464768070

Encryption and decryption enabled for port-id: 2

Invoking SYS_writeManagementSDU (330)

Sent CDAP message of size 95 through port-id 2:
Opcode: 12_M_WRITE
Object class: Client finish
Object name: Client finish
Scope: 0

Invoking SYS_readManagementSDU (329)

Received CDAP message from N-1 port 2
Opcode: 12_M_WRITE
Object class: Server finish
Object name: Server finish
Scope: 0

Authentication was successful, waiting for M_CONNECT_R

Invoking SYS_readManagementSDU (329)

Connection response received
Received CDAP message from N-1 port 2
Opcode: 1_M_CONNECT_R
Abstract syntax: 115
Authentication policy: Policy name:
Supported versions:

Source AP name: test2.IRATI
Source AP instance: 1
Source AE name: Management
Destination AP name: test1.IRATI
Destination AP instance: 1
Destination AE name: Management
Invoke id: 1
Result: 0
Version: 1

9094(1464768076)#ipcp[2].enrollment-task (DBG): M_CONNECT_R cdapMessage from portId 2
9094(1464768076)#librina.ipc-api (DBG): IPCManager.getRegisteredApplications called
9094(1464768076)#librina.syscalls (DBG): Invoking SYS_writeManagementSDU (330)