Deliverable-4.1

Draft Conceptual and High-level Engineering Design of Innovative Security and Reliability Enablers

Deliverable Editor: Hamid Asgari,TRT

Publication date:	30-September-2014
Deliverable Nature:	Report
Dissemination level (Confidentiality):	PU (Public)
Project acronym:	PRISTINE
Project full title:	Programmability In RINA for European supremacy of virTualised NEtworks
Website:	www.ict-pristine.eu
Keywords:	DIF, management, system, RIB, elements, common, security, reliability
Synopsis:	D4.1 describes the techniques developed within WP4 focused on authentication, access control, data protection, autonomous security coordination, resiliency and high availability to enable networks that are more secure and reliable than those we have today.

The research leading to these results has received funding from the European Community's Seventh Framework Programme for research, technological development and demonstration under Grant Agreement No. 619305.

Disclaimer

Executive Summary

This document, D4.1, is the result of WP4 activities and is a key deliverable of the project. A high-level security functional architecture is defined that identifies the key functional blocks designed to offer secure data delivery across RINA infrastructure. The functional decomposition of the architecture identifies and details the various internal RINA functions embedded in Inter Process Communication (IPC) processes, and Distributed Application Facilities (DAFs)/ Distributed IPC Facilities (DIFs) that support the network in secure delivery of data in each of the management, control and data planes.

This deliverables explains the following principal functions for achieving security in RINA-based networks:

- Authentication and Authorisation models, which define the principal actors in facilitating secure communication and content delivery within and across multiple domains. These models are mainly based on current practices that are adapted to the RINA concept. An emphasis is also put on how we can achieve Multi-Level Security in RINA
- The Key Management function that includes the generation, exchange, storage, use, and replacement of keys for different functions including authentication, authorisation and user data protection.
- The secure channel is studied to investigate how it can be established, used and managed within RINA to protect data from eavesdropping and tampering. Primarily, the aim is to protect the messages exchanged when an IPC is in the process of joining a DIF and to allow keys to be negotiated per connection. SDU protection, which uses cryptographic mechanisms to achieve integrity and confidentiality, is detailed.
- The identification of threats to the RINA infrastructure and the functions required to combat the threats and vulnerabilities is carried out. There are several types of attacks on network communications: eavesdropping, disrupting or blocking communication, injecting fabricated packets, modifying the storage, tables or packets. Here, we perform a security risk assessment to identify runtime threats to a RINA network and define measures to mitigate them includes monitoring, analysis, and execution of the strategies which should be put in place.
- A primary objective for RINA is to maintain the network resiliency in the case of failures and attacks, ensuring high-availability of the network for providing

the assumed services. In this deliverable methods for improving resiliency are explained, specifically how to deal with IPC and link failures and exploitation of vulnerabilities.

Finally, the work plan is defined to further design, develop and realise these functions for WP6 activities.

Table of Contents

Acronyms 7
1. Introduction 13
1.1. The Functional Picture of the Security Solution 15
1.2. Security Policy Management 18
1.3. Network-Wide Resiliency and Availability 19
2. Authentication of RINA Processes 21
2.1. General considerations about authentication mechanisms 21
2.2. Authentication procedures in RINA networks 22
2.2.1. Application process authentication at the IPC service API (authentication between layers, vertical) 22
2.2.2. Mutual authentication of APs (or IPC Processes) within a DAF/DIF (authentication within a layer, horizontal) 24
2.3. Authentication Policies under study 28
2.3.1. The *AuthNNone* Authentication Mechanism 30
2.3.2. The *AuthNPassword* Authentication Mechanism 31
2.3.3. The *AuthNSessionKey* Authentication Mechanism 32
2.3.4. The *AuthNAsymmetricKey* Authentication Mechanism 33
2.3.5. The *AuthNCertificate* Authentication Mechanism 34
2.3.6. The *AuthNToken* Authentication Mechanism 35
2.4. Summary 35
3. Access Management 36
3.1. Authorisation and Access Control 36
3.1.1. Access Control Mechanisms 36
3.1.2. Use of Capability Based Access Control in RINA 38
3.1.3. Summary and Open Issues 40
3.2. Multi-Level Security (MLS) 40
3.2.1. Overview of MLS 41
3.2.2. Three Facets of an MLS framework 42
3.2.3. MLS architectures 45
3.2.4. Achieving MLS in RINA 51
3.2.5. Summary and Open Issues 60
4. Secure Channel and SDU Protection 62
4.1. Secure Channel 62
4.1.1. Aim of the Secure Channel 62
4.1.2. Use of Secure Channel 62
4.1.3. Transport Layer Security 64

4.1.4. Secure Channel Protocol in RINA 67
4.1.5. Summary and Conclusions 71
4.2. SDU Protection 72
4.2.1. Protection Mechanisms 73
4.2.2. Operation 74
4.3. The Selected Mechanisms for Design and Implementation 75
4.3.1. Objectives of Simulation 75
4.3.2. Methodology 76
4.4. Summary 76
5. Key Management Function 77
5.1. Key Management Architectures 77
5.2. Conclusion 80
6. Threat Identification, Monitoring and Countermeasures 81
6.1. Risk Assessment Methodology 81
6.2. Context and Scope 81
6.3. Asset Identification 83
6.4. Threat Scenarios 85
6.5. Security Risk Assessment 89
6.6. Security Controls 93
6.7. Monitoring and Counter Measures 97
6.8. Summary 98
7. Resiliency 99
7.1. State of the Art and Relevance to RINA 99
7.1.1. Failure detection in packet switched networks 99
7.1.2. Recovery in packet switched networks 101
7.2. Policies for Failure Detection 107
7.2.1. Flow Liveness Detection 109
7.2.2. Flow Loopback Request Policy 111
7.3. Policies for Resilient Routing 114
7.3.1. Definition of Terms 115
7.3.2. Narrative description of the Loop Free Alternates policy 116
8. Summary and Conclusions 122
9. References 124

Acronyms

ABAC
Attribute Based Access Control

AC
Access Control

AC-IB
Access Control Information Base

ACL
Access Control List

ACM
Access Control Manager

AE
Application Entity

AEAD
Authenticated Encryption with Additional Data

AP
Application Process

BFD
Bidirectional Forwarding Detection

BPC
Boundary Protection Component

CA
Certificate Authority

CACEP
Common Application Connection Establishment Protocol

CBAC
Capability Based Access Control

CCM
Continuity Check Message

CCP
Continuity Check Protocol

CDAP
Common Distributed Application Protocol

CER-id
Connection Endpoint Id

CLI
Command Line Interface

COTS
Commercial Off The Shelf

DA
Distributed Application

DAF
Distributed Application Facility

DIF
Distributed IPC Facility

DTCP
Data Transfer Control Protocol

DTLS
Datagram Transport Layer Security

DTP
Data Transfer Protocol

EFCP
Error Flow Control Protocol

EFCPI
Error Flow Control Protocol Instance

ESP
Encapsulating Security Payload

FA
Flow Allocator

FAI
Flow Allocator Instance

FLD
Flow Liveness Detection

FLR
Flow Loopback Request

FMGR
Flow Manager

FMGRI
Flow Manager Identifier

FMON
Flow Monitor

FSDB
Flow State Database

FSM
Finite State Machine

FSO
Flow State Object

FT
Forwarding Table

GPB
Google Protocol Buffers

ICMP
Internet Control Message Protocol

IPC
Inter Process Communication

IPCP
Inter Process Communication Process

IRM
IPC Resource Manager

IS-IS
Intermediate System to Intermediate System

IV
Initialisation Vector

KM
Key Manager

LB
Loopback

LFA
Loop Free Alternate

LSA
Link State Advertisement

LSR
Label Switched Router

LT
Link Trace

MA
Management Agent

MAC
Message Authentication Code

MILS
Multiple Independent Levels of Security

MLS
Multi Level Security

MP
Merge Point

MPTCP
Multipath TCP

MSL
Multiple Single Levels

NHOP
Next-hop

NNHOP
Next-Next-hop

OAM
Operations, Administration, and Maintenance

OS
Operating System

OSI
Open Systems Interconnection

OSPF
Open Shortest Path First

PCI
Protocol-Control-Information

PDU
Protocol Data Unit

PFF
PDU Forwarding Function

PFT
Protocol Data Unit Forwarding Table

PFTG
PDU Forwarding Table Generator

PKI
Public Key Infrastructure

PLR
Point of Local Repair

R
Restricted

RA
Resource Allocator

RBAC
Role Based Access Control

RIB
Resource Information Base

RINA
Recursive InterNetwork Architecture

RINASim
RINA Simulator

RMT
Relaying and Multiplexing Task

RSVP-TE
ReSerVation Protocol with Traffic Engineering extensions

RT
Routing Table

RTT
Round Trip Time

S
Secret

SDU
Service Data Unit

SRP
Secure Remote Password

TCP
Transmission Control Protocol

TLS
Transport Later Security

TTP
Trusted Third Party

U
Unclassified

UDP
User Datagram Protocol

VMM
Virtual Machine Monitor

1. Introduction

Recursive InterNetwork Architecture (RINA) is a clean-slate network architecture built on the premise that networking is Inter Process Communication (IPC). A RINA network consists of a Distributed IPC Facility (DIF) as a layer that repeats as many times as is necessary to effectively cover the range required for the operation of the network. A DIF is a distributed application that performs a coordinated set of policy-managed mechanisms to provide IPC services. Every DIF implements the same functions and uses the same protocols, but is configured with different policies to fulfil the particular requirements of the layer.

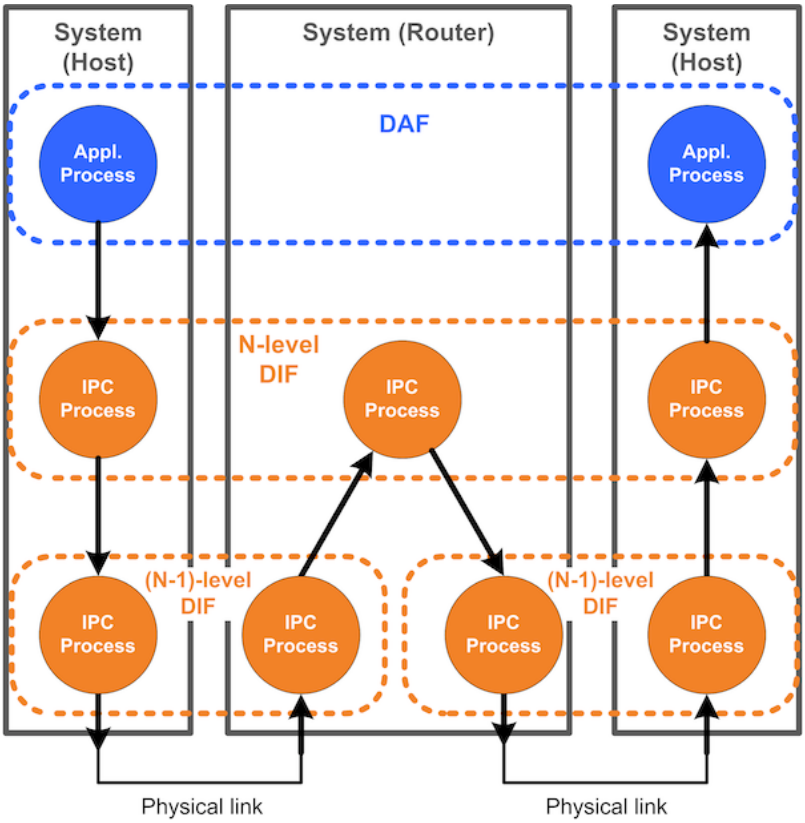

Figure 1. RINA network

Figure 1, "RINA network" shows a RINA network. Application Processes in a Distributed Application Facility (DAF) communicate via an underlying DIF. They may also themselves be IPC processes for a higher level DIF. There is no direct IPC between DIFs of the same level, e.g. between two (N-1)-level DIFs in Figure 1, "RINA network"; they must relay to the DIF above. A DIF enforces strict layer boundaries so that what happens internally to the DIF is not visible externally. This means that all entities external to the DIF cannot address its members. It also means that a DIF cannot instruct how its underlying DIF to forward its data, e.g. whether the data is sent via another underlying DIF or a physical wire.

Every DIF uses the same two protocols: a data transfer and control protocol called Error and Flow Control Protocol (EFCP) and an object-oriented application protocol called Common Distributed Application Protocol (CDAP). EFCP is made up of two protocols: Data Transfer Protocol (DTP), which relays the payload from the source to the destination; and Data Transfer Control Protocol (DTCP), which performs the error and flow control functions, sending feedback from destination to source. CDAP is used within the DIF for layer management functions, e.g. IPC enrolment, routing, security management and access control. It enables six operations to be performed on objects (e.g. in the RIB): create/delete, read/write, and start/stop.

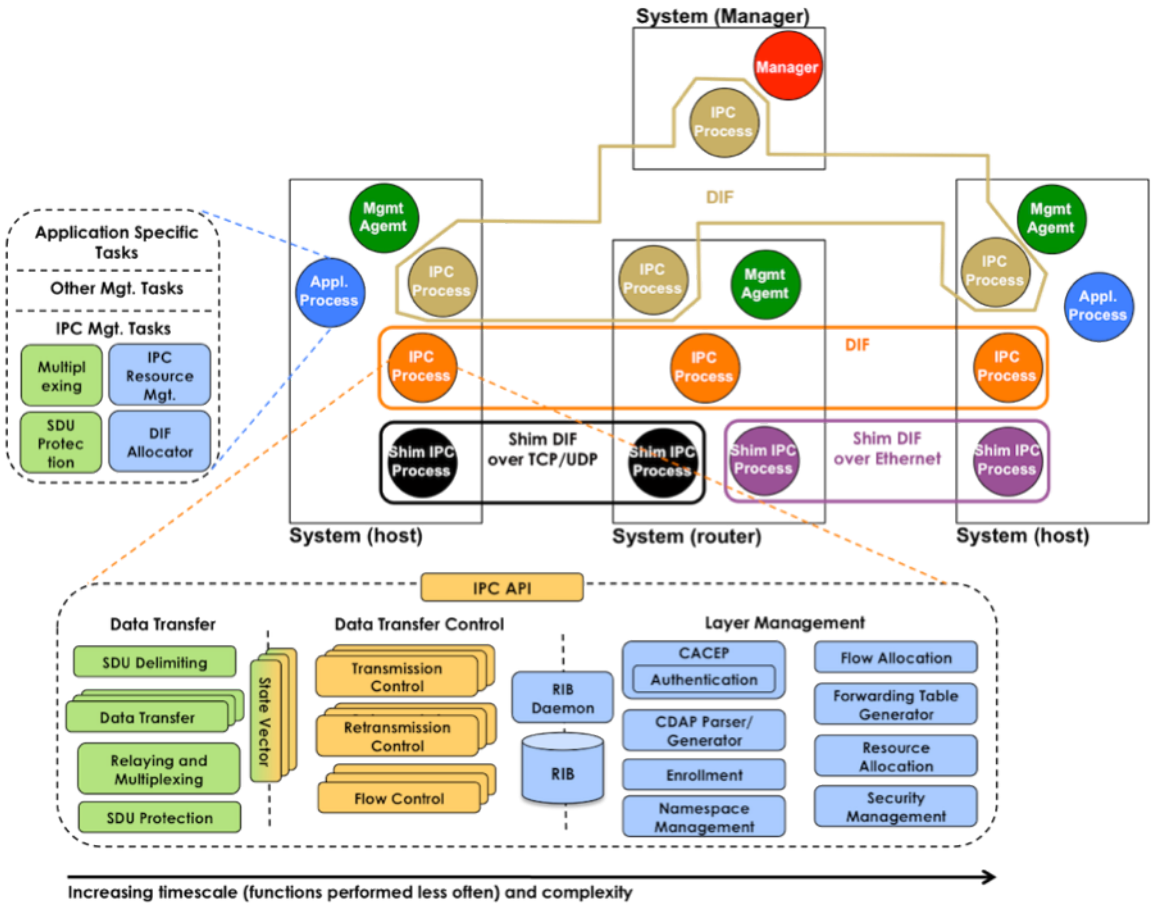

Figure 2. RINA Architecture

Figure 2, "RINA Architecture" shows the internal parts of an IPC process and its associated protocols. A DIF is a collection of IPC processes that execute three sets of mechanisms: data transfer, data control and IPC management. Data transfer is performed using EFCP's DTP. When transferring data, SDU (Service Data Unit) Delimiting performs fragmentation or concatenation on the data, depending on the flow configuration . SDU Protection applies encryption for confidentiality or integrity protection, according to the configured policies, This is to protect the SDU in transit passing through the N-1 DIFs. It may also perform compression, error correction or

apply time to live mechanisms. Data transfer control is achieved using DTCP, which may perform flow and retransmission control. IPC management process includes support for an IPC joining a DIF (enrolment), resource allocation, flow allocation and routing.

IPC processes communicate and share state information using CDAP to update the relevant objects stored in the Resource Information Base (RIB). The RIB is the logical representation of all information known by the IPC. Each member of a DIF maintains a RIB. It includes policies, forwarding table data and addresses, as well as security data such as authentication credentials and key material. Access to the RIB is controlled by the RIB Daemon, which acts as a broker. The RIB Daemon manages the information stored in the RIB and its veracity, updating and making states available to IPC and application processes.

1.1. The Functional Picture of the Security Solution

Within 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 security or administrative domain manages aspects that apply to multiple DIFs or the interactions between DIFs. The high-level security architecture is shown in Figure 3, “High-level Security Architecture”. 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 anomalies, intrusions and failures and is responsible for implementing mitigation strategies and countermeasures. Although security policies are stored in the RIB of the IPC, they are managed at the level of the security or administrative domain. 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, although some key management functions may sit in the DIF. The domain also manages network flow protection as well as network-wide resiliency and redundancy functions.

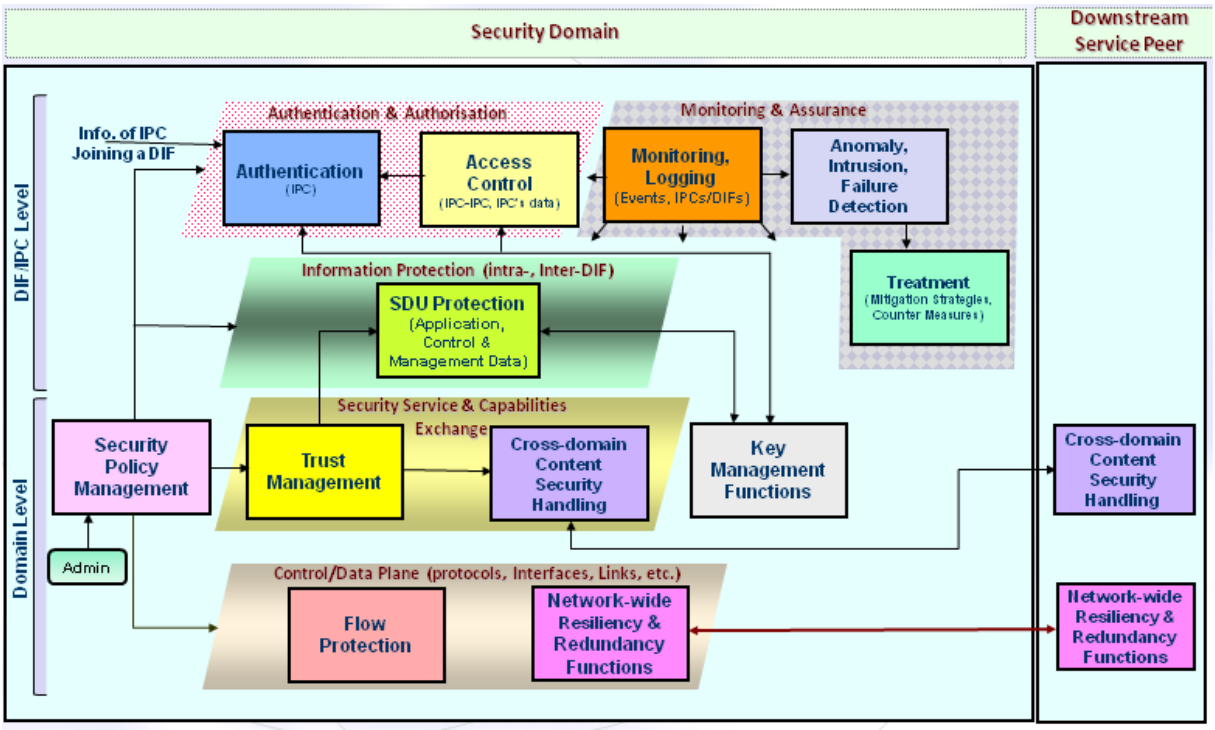

Figure 3. High-level Security Architecture

Figure 4, "Placement of Main Security Functions in RINA" depicts the placement of authentication, access control and data protection functions in RINA. This deliverable is organised to firstly look at the security aspects that are shown in Figure 3, "High-level Security Architecture" including Authentication, managing Access Control, Secure Channel and SDU Protection, Key Management functions, identification of security threats, monitoring and specifying the counter measures for combating the threats. The second part of this deliverable looks at network resiliency and availability.

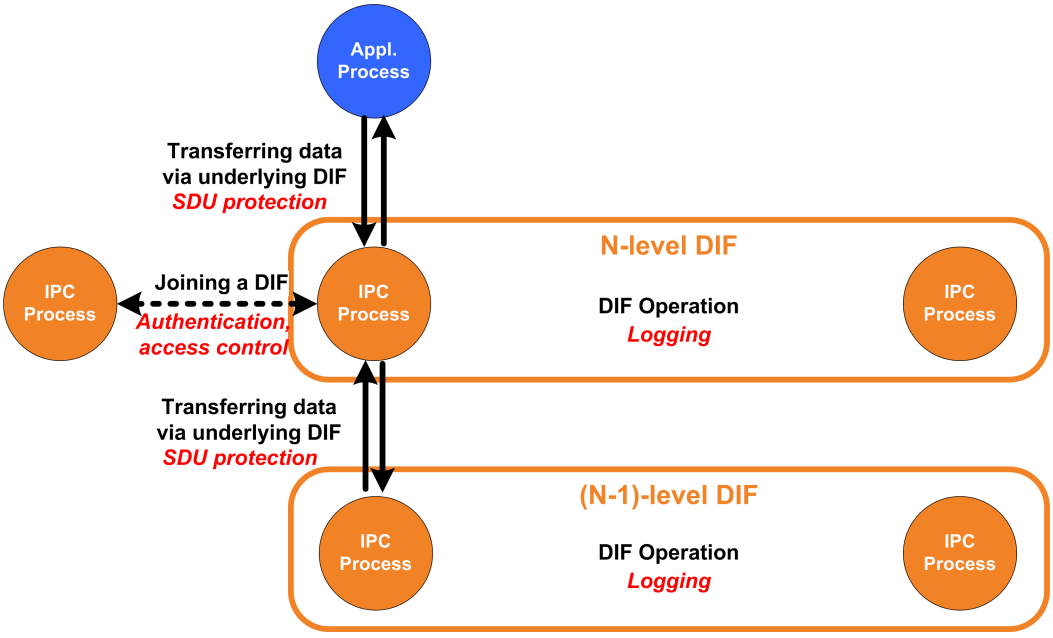

Figure 4. Placement of Main Security Functions in RINA

Once the flow through the underlying DIF has been created, the new IPC member must create an application connection with the DIF member using CACEP. The new IPC member and the DIF member may authenticate each other according to the DIF's policy. 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 in the DIF, 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. These remote operations (communicated via CDAP messages targeting one or more objects of the IPC Process RIB) may be invoked by other IPC Processes, in the case of layer management related operations, or by the Network Management System. Fine grained 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).

In cryptography, a **secure channel** is a means of transferring data that is resistant to overhearing and tampering. When joining a DIF, an IPC authenticates sending its credentials to the DIF. A secure channel is needed to prevent the credentials from being eavesdropped or modified. Once an IPC is enrolled in a DIF, a secure channel to other members of the DIF can be achieved using SDU protection. The secure channel is principally usable for protecting RINA management traffic (e.g. CDAP), but is also capable of being offered as a service to applications.

The **secure management of keys** used for authentication, access control and cryptographic purposes is one of the most critical elements when integrating cryptographic functions into a system. The purpose of key management functions is to provide secure procedures for handling cryptographic keying material to be used in symmetric or asymmetric cryptographic mechanisms. The main functions of a key management system include entity registration, key generation, and retrieval, verification, key distribution and revocation.

Identifying the abnormal events and information to be monitored is a key part of detecting potential attacks from misbehaving DIF members. We will categorise the different types of attacks that a DIF can suffer, their symptoms and the counter-measures that can be taken to respond to them.

Logging and monitoring processes involve collecting raw data, deriving statistics for presentation to the other functional entities, and measuring performance to ensure it is within contracted levels. The main goals are 1) to perform information

security monitoring 2) to evaluate and analyse the situation, and detection of security incidents and compromised entities, in order to inform other entities who can then initiate corrective actions, 3) to gather information in order to verify whether security guarantees are in fact being met.

1.2. Security Policy Management

RINA separates mechanism and policy; in RINA, a mechanism is the invariant part of a component, while a policy is any aspect that can be changed. This means that while each DIF implements the same mechanisms, they can be configured differently using policies. This allows different DIFs to have their own authentication, access control and SDU protection policies. Generally, policies are defined as a predefined set of rules and guidelines reflecting the intention of e.g., adequately protecting valuable data. A policy may be a configuration parameter, e.g. how often to update a key, or it may be a particular implementation, e.g. password authentication or certificate. Policies are codified into logics that are given by the Distributed Management System (DMS) to IPC processes and are stored in the RIB. The policies are used by Policy Consumers, which are associated with specific functionality in IPC or other processes within DAF/DIF. Policies that are stored in the RIB are accessed and updated via the RIB Daemon using CDAP.

There exist many functionally different Policy Consumers associated with the functional blocks shown in Figure 1, “RINA network”. IPC processes communicate with the DMS through the Management Agent (MA) in order to get configuration/information about relevant policies or policy-triggering event (e.g., detection of a policy violation). Configurations are generated to implement the security policy, which are passed to the relevant IPC processes and then to their Policy Consumers. The Policy Consumers receive these pieces of information, interpret them to real element-specific commands or parameterised functions, and then deploy the configurations to the enforcement point of the relevant Policy Consumer.

The relevant policies in the context of security management in RINA will cover the following across the DAF/DIF:

- Authentication and access control policies
- Content protection policies
- Policies related to filtering and information flows between different levels to enable information sharing without compromising information security.
- Domain-level security handling policies such as integrity and domain-specific policies.

1.3. Network-Wide Resiliency and Availability

The network's primary objective is the transmission of data as a service to applications. From this point of view, its security concerns are first and foremost related to maintaining the availability of this service. As such, the priority security requirements will be resiliency and high availability.

There are two approaches to achieving resiliency: reactively restoring the network when an error occurs or proactively protecting the network through redundancy. Methods for improving resiliency usually include the following actions: failure detection, failure localisation and recovery. Failure detection follows the most basic premise; we need to become aware of a failure before we can take any action to mitigate its impact on the overall system's performance. After detecting a failure, we can take actions to restore the performance of the system to an acceptable level. This may require an additional action: failure localisation, as some recovery mechanisms need to know the location of the failure before they can efficiently perform recovery. Finally, to return the system to its original state a fourth action is needed: fault repair.

Within a RINA network, the responsibility for detecting, 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. There are two types of failures in a RINA network: link failures and IPC process failures. Link failures are failures that occur between two IPC processes in a DIF. These failures will always correspond to a failure in the underlying DIF. 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 lowest-ranked higher level DIF with sufficient capabilities.

In addition to improving resiliency, ways to maintain the availability of the service include preventing unauthorised modification of management data, and the exploitation of vulnerabilities in RINA protocols, APIs and their implementations. We consider these in turn below.

To prevent unauthorised modification of management data, integrity of management communications and access control to the RIB are a priority. Therefore the secure channel identified earlier would be of significant use.

In terms of exploitation of vulnerabilities in RINA protocols, the situation is more complicated. Part of the solution lies in correct design of the RINA protocols and APIs,

which, although important, is outside of the scope of PRISTINE. Part of the solution will also lie in the correct design of DIF management applications and their protocols, such as secure routing protocols for example. With the exception of some aspects of routing, these are too wide a scope for us to cover in PRISTINE.

Some of the potential vulnerabilities in RINA protocols can be alleviated by using the secure channel identified earlier to guarantee integrity of data, and authentication of the endpoints. It could also provide confidentiality, which although not a primary requirement, could be useful to hide details of the workings of the DIFs that may be useful for attackers. Note that such a protocol does not protect IPCs from less trusted or potentially compromised IPCs, and hence vulnerabilities of the RINA protocols may need to be considered from that perspective.

2. Authentication of RINA Processes

This section is intended to define the essential procedure and methods to be applied for the identification and authentication of RINA IPC Processes (IPCPs), and especially during the application connection establishment or CACEP. In general, this procedure and methods are applicable to any pair of Application Processes (APs) within a DAF, but this section will focus on the particular case of IPCPs within a DIF.

2.1. General considerations about authentication mechanisms

Authentication can be defined as "the act of confirming the truth of an attribute of a single piece of data (datum) or entity". The concept identification, in contrast, refers to "the act of stating or otherwise indicating a claim purportedly attesting to a person or thing's identity", where the authentication is the process of actually confirming that identity.

In the field of computer science though, one usually refers to authentication in general for both identification and authentication itself. The fundamental requirement here is to unequivocally identify the other peer AP using an authentication mechanism, to further perform proper encryption of the messages exchanged by the APs, and also to properly apply authorisation mechanisms to the communication.

To perform authentication of entity B by an entity A, there is the need to have some *a priori* knowledge of B by A. There are fundamentally two well-known high-level strategies to perform an authentication:

a) The simplest approach is using symmetric cryptographic techniques. An entity A has a *shared secret* (e.g. a key phrase) with B. The shared secret between A and B must be exchanged **securely** beforehand e.g. physically, electronically..., otherwise the authentication process could be faked (faking the identity of B) or intercepted (e.g. man-in-the-middle). In this scenario, A requests B to perform an operation over a piece of data or *challenge*, chosen by A at its discretion, and the shared secret. A verifies that the result is the one expected, and on success, and given the right premises exposed before, A has authenticated B. It is important to note that both, the length of the shared secret and the challenge, should be sufficiently large to ensure that the authentication procedure is **computationally secure** [SP800-56B] [1] .

[1] Said of a cipher that cannot be broken with the current computer technology within a period short enough to be practicable.

b) A more advanced approach is the use of asymmetric cryptographic techniques like a pair of public/private keys. The key concept is that entity B has a private key, only known by itself, whereas the public key is publicly available. The keys are complementary, meaning a piece of data encrypted with the private key can only be decrypted by using the public key and vice-versa. The authentication procedure, therefore, works in a similar way as the symmetric case: the entity A requests B to perform a certain operation with a piece of data or *challenge*, and entity B should use its private key to perform it and encrypt the result. Entity A, given the fact that it knows the public key of B, can decrypt the message to certify that B is the right peer. The premises for this method to be reliable are fundamentally the same; the challenge and the key(public/private) shall be large enough to be computationally secure, and the distribution of the public key must be done in a way that cannot be mangled or replaced by a third entity.

In both approaches there is a strong requirement of correctly distributing the shared secrets or the public keys. This problem will be further discussed in the Key Management section. Nevertheless, it is worth to mention now that in most asymmetric cryptographic systems, such as RSA, more advanced key management techniques such as Public Key Infrastructure (PKI) are used.

2.2. Authentication procedures in RINA networks

There are two main locations where authentication/access control can be performed in the RINA architecture, explained in the following subsections.

2.2.1. Application process authentication at the IPC service API (authentication between layers, vertical)

The IPC service API is the API seen by an Application Process using a DIF, including IPCPs (IPCPs are just Application Processes that perform IPC). The scope of this API is local to the processing system, where the DAF in layer N requests a service (flow allocation) to a DIF in layer N-1. As part of the information passed during a flow allocation request, the AP has to provide its credentials (if the DIF that will be servicing the application request doesn't perform any authentication procedure, this credentials can be null). The local IPC Resource Manager (or origin IRM) can optionally - based on a configuration or policy - apply certain authorization checks (such as Access Control Lists or ACLs) to the requests from a DAF in layer N to a DIF in N-1; based among others on the DAF name or the application process name. These ACLs are still local to the processing system. The whole procedure is illustrated in the left side of Figure 5, "Authentication at the IPC API level" below.

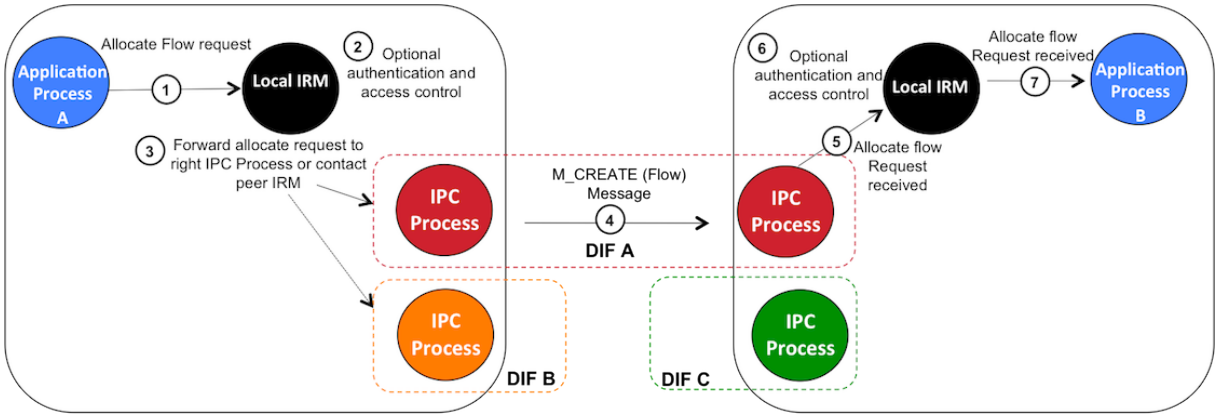

Figure 5. Authentication at the IPC API level

In order to prevent spoofing by other APs within the processing system - which could lead to resource abuse by unauthorized APs/IPCPs - the IPC service API should authenticate the APs. The authentication, from the perspective of the AP, can be done implicitly - meaning that the IRM can securely authenticate the AP using OS process data that the process itself cannot forge or mangle - or explicitly - by passing the authentication information as a parameter to the DIF service API. The authentication/ authorization decision may be performed by the IRM or by the IPC Process of the DIF chosen to provide the IPC service.

Once the allocate request has arrived at the local IPC Process and optional authentication/authorization procedures have been successfully carried out, the IPC Process will start the flow allocation procedure. As part of this procedure the IPC Process at the source system will send a CDAP M_CREATE message to a peer IPC Process, which will be forwarded (based on directory lookup rules) until it reaches the destination IPC Process. This message contains all the data necessary to negotiate the flow creation with the destination IPC Process, including the credentials of the application that requested the flow. Once the message reaches the destination IPC Process, it can perform an authorization/access control decision to decide if the source application has enough privileges to request a flow allocation to the destination application. The destination IPC Process can also delegate this check to the local IRM, who will end up receiving the allocate request. Finally, if all authentication/ access control procedures complete successfully, the allocate request is delivered to the application via the IPC API. The procedure is illustrated at the right side of Figure 5, "Authentication at the IPC API level" above.

In order to perform a secure authentication and avoid probe attacks, the destination IRM should properly authenticate the requester AP during the flow allocation request. The parameters of this authentication shall be provided as optional ACL authentication parameters by APsource during the flow allocation to the DIF API. The DIF(s) N-1, and

more specifically the IPCP(s), will use this data during the lookup of an IPCPdest that has the APdest in layer N.

Due to the nature of the initial flow allocation request -single direction-, and the will to be able to hide the presence of a certain AP in the DAF/DIF, only asymmetric cryptographically authentication techniques can be used, otherwise the authentication procedure may require a series of message exchanges, which would implicitly imply that that the APdest exists, which would defeat the initial purpose of the ACLs.

Due to the reduced interest on this mechanism for the PRISTINE use-cases, the PRISTINE project will concentrate its efforts on other authentication scopes.

2.2.2. Mutual authentication of APs (or IPC Processes) within a DAF/DIF (authentication within a layer, horizontal)

The authentication between IPCPs within a DIF is a policy, and more specifically, a policy of the CACEP module executed at application connection setup time. For IPCPs or any other AP from a DAF using CDAP in general, the CACEP authentication should be the fundamental authentication point, and no further mechanisms for authentication between IPC Processes should be envisioned elsewhere. Figure 6, “Authentication between APs when establishing an application connection” below illustrates the procedure of application connection establishment between two arbitrary application processes:

- First one application process sends an M_CONNECT message to the other one, providing its naming information and indicating the desired abstract and concrete syntaxes of the CDAP protocol, as well as the version of the RIB to be used in the dialogue. As part of this information, the AP may also propose one or more authentication options.
- The other application process receives the M_CONNECT message, and decides if it has to authenticate the other IPC Process (this is a policy of the DAF). If so, a number of messages may be exchanged between both processes in order to carry out the authentication procedure.
- If authentication is successful and the application process accepts the application connection in the terms proposed by the requestor, the AP responds with a positive M_CONNECT_R (in the case that authentication is no successful or the connection is rejected, the AP may or may not reply with a negative M_CONNECT_R message).

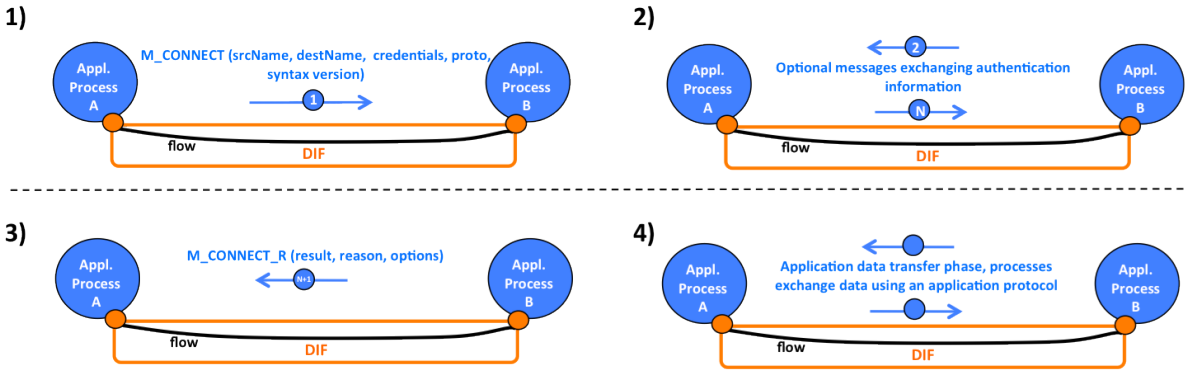

Figure 6. Authentication between APs when establishing an application connection

Establishment of application connections between APs is typically performed just before trying to enrol to join a DAF. However, there are other situations in which application connections can be established. If we focus on DIFs - PRISTINE's interest - application connections between IPC Processes may be established not only before enrolment but also after, for instance:

- By request of an existing member IPC Process that informs another IPC Process that there is a third IPC Process with which it shares an N-1 DIF, therefore they have the potential of becoming nearest neighbours by establishing an application connection.
- An IPCP may only accept flow allocation requests of the IPCPs in the DIF he has an application connection with (and therefore that have been authenticated by him).

Authentication policies within a DIF may vary depending on the type and state of the IPC Processes establishing the application connection: for example IPCPs executing in end-user devices vs. IPCPs in ISP routers, IPCPs that are already members of the DIF vs. IPCPs that still have to join, etc. These policies may vary depending on the DIF conditions; for example if a DIF detects that one or more of its IPCPs have been compromised, authentication policies may transition to become more restrictive.

PRISTINE will initially focus on a single authentication mechanism per DIF use-case, as being the most relevant and common authentication schema. With the constraints previously mentioned, one can identify three main groups of policies (configurations) for the authentication within the application connection establishment procedure that are of interest within PRISTINE:

a) No authentication. If there exists already a trust relation between peers of the DIF (for example because all IPCPs reside in systems controlled by a single owner in trusted locations), there is no need to use authentication procedures. Examples can be: small scope shim-DIFs, lower DIFs in general.

b) Authentication with nearest neighbours during enrolment. IPCPs enrolling to the DIF authenticate themselves with the nearest neighbours only. Once the authentication to its nearest neighbour(s) has been successful, the IPCP can successfully enrol. It is important to mention that in this

c) All-with-all authentication; the IPCP joining the DIF authenticates against each and every current IPCP that is a member of the DIF. This configuration is very expensive (O(N)), so it may be only suitable for small scope DIFs, where the number of members is limited, and where there is no trust in the authentication performed by any other IPCP but itself. This configuration, though possible, appears to be of limited or no use.

Both authentication methods, b) and c), require revocation methods when members need to be excluded from the DIF, due to administrative decisions or because they have been compromised, in order to prevent the offending IPCP to be able to re-join the DIF via other IPCPs.

Proposal of authentication policy between two peers within a DIF

For the authentication of IPCPs during enrolment we consider two entities: the Initiating Process (IP-IPCP), the one willing to initiate the communication, and the Destination Process (DP-IPCP), the one receiving the initial request from the IP-IPCP. The DP-IPCP is a process that is already a member of the DIF. The IP-IPCP is a process not participating in the DIF yet, or already participating but that doesn't have an application connection to the DP-IPCP; e.g. due to network partitions.

- The IP-IPCP sends a M_Connect Request to the DP-IPCP. Along with other necessary information it shall include the identity and authentication parameters:
  - The IP-IPCP application name
  - The DP-IPCP application name
  - The authentication mechanism requested from the DP-IPCP
  - The authentication mechanism(s) offered by the IP-IPCP
  - The initial authentication data, if needed by the requested and supported mechanism(s)
- According to these identity and authentication parameters and its policy, the DP-IPCP may directly accept or reject the M_Connect Request by means of an M_Connect Response, or initiate a CDAP dialog to exchange further authentication data.

- If the DP-IPCP rejects the connection due to an authentication issue it shall include the cause (authentication failure, unsupported DP-IPCP authentication requested, or unsupported IP-IPCP authentication provided) and a list of the authentication mechanisms acceptable for the IP-IPCP and available from the DP-IPCP.
- If the DP-IPCP accepts the connection using an M_Connect Response, it shall include the following identity and authentication paramaters:
  - The DP-IPCP application name
  - The IP-IPCP application name
  - The authentication mechanism accepted for the IP-IPCP
  - The authentication mechanism provided by the DP-IPCP
  - The initial authentication data, if needed by the accepted and provided mechanism(s)
  - Optionally, additional data derived from the authentication, and suitable for SDU protection (like a session key) and/or further authentication procedures (such as the same session key or an authorization token)
- If the DP-IPCP has to initiate a CDAP authentication data exchange, it shall include in its first message the following identity and authentication parameters:
  - The DP-IPCP application name
  - The IP-IPCP application name
  - The authentication mechanism accepted for the IP-IPCP
  - The authentication mechanism provided by the DP-IPCP
  - The initial authentication data, if needed by the accepted and provided mechanism(s)
- In the case of a positive M_Connect Response or a request for further authentication data, the IP-IPCP shall analyze the response from the DP-IPCP, consider the connection established, proceed to the exchange of additional authentication data, or release the connection.
- If additional authentication data need to be exchanged, IP-IPCP and DP-IPCP shall do so using appropriate CDAP messages.
- When the DP-IPCP considers the authentication process finished, it shall send to the IP-IPCP a M_Connect Response, accepting or rejecting the connection.
  - If the connection is rejected, the authentication failure shall be identified as the cause.

    - If the connection is accepted, the message may contain additional data derived from the authentication, and suitable for SDU protection (like a session key) and/or further authentication procedures (such as the same session key or an authorization token)

- At any point of the exchange, the IP-IPCP can decide to release the connection due to an authentication failure. It shall include this cause in the M_Release Request sent

At the end of a successful authentication data exchange, both processes will have adequately identified each other according their applicable policies, and optionally exchanged additional data required for SDU protection and/or to simplify further authentications within the same DIF.

In the case of an unsuccessful authentication data exchange, the IP-IPCP may decide to use or request alternate authentication mechanisms, or retry after a configured interval. A DP-IPCP may decide not to accept further attempts from the same IP-IPCP during a given interval or after a number of retries. This behaviour with respect to retries in case of authentication failures is a matter for policy specification.

2.3. Authentication Policies under study

Six different authentication policies are proposed here, ranging from the simplest possible (no authentication, trusting the application names exchanged in the connect messages) to one requiring the support of Public Key Infrastructure (PKI) mechanisms.

- None: No authentication beyond the application names in the connect messages is required and/or provided (*AuthNNone*)
- Symmetric: The authenticity is proved by the fact of being able to encrypt a random challenge sequence. The APs need to have as a previous knowledge a shared secret that in this case is a cryptographic key. The AP that wants to be authenticated will receive a challenge and will use the key to encrypt the challenge and send it back. This key can have the form of:
    - A shared secret associated with the application name, similar to the username/password pairs (*AuthNPassword*)
    - A DIF session key previously exchanged (*AuthNSessionKey*)
- Asymmetric: These methods use public-key cryptography, where a key is composed of a public key and a private (secret) key. Any message encrypted with the public key can only be decrypted with its corresponding private key, and vice versa. The

authenticity is proved then by being able to *sign* messages, which means encrypting messages with a private key. In these cases there is no secret that needs to be shared in advance, but it is still needed to trust the source of the public key to avoid man-in-the-middle attacks. There are different ways to apply these methods:

- By means of a secret/public key pair, associated to the application, where public keys have previously been directly distributed (*AuthNAsymmetricKey*)
- By means of a secret/public key pair, associated to the application, where public keys have previously been distributed by a PKI infrastructure (*AuthNCertificate*)
- By means of an authentication token, that is signed by a trusted third party (TTP), and received in a previous exchange (*AuthNToken*)

Note that the last two methods (*AuthNCertificate* & *AuthNToken*) avoid the direct exchange of the public keys by trusting another common entity and thus they are recommended when many AP are involved, for scalability reasons.

Another important characteristic of the authentication methods is the length of the validity period:

- Temporal (*AuthNToken* & *AuthNSessionKey*): Based on previous authentication procedures within the same DIF, and suitable to be completed in a single M_Connect Request/Reply round, without additional CACE messages. To be applicable, temporal mechanisms require the fulfillment of a previous non-temporal one that provides the necessary data. This fulfillment can be done online (in a direct interaction with another process in a DIF) or offline (typically by configuration).
- Non-temporal (*AuthNPassword*, *AuthNAsymmetricKey* & *AuthNCertificate*): They require the exchange of challenge/response messages during the authentication phase. Note that the name *non-temporal*, does not imply that the authentication is valid forever. Actually non-temporal authentication methods can be cancelled by revocation or reconfiguration.

The proposed methods are only illustrations of the different possibilities that have been considered to perform authentication. Among them, the ones being chosen for further specification and implementation will have to follow the matching standard defined in ISO/IEC 9798 [2] [3] [4] [5] [6] [7].

[2] ISO/IEC 9798-1:1997 Information technology — Security techniques — Entity authentication — Part 1: General

[3] ISO/IEC 9798-2:1999 Information technology — Security techniques — Entity authentication — Part 2: Mechanisms using symmetric encipherment algorithms

The following diagrams show in detail the messages needed by each of the methods presented before, as well as the previous information needed and the kind of calculations involved. Note that the green squares show the process of key negotiation, which takes place after the process of authentication is finalized. This process is only shown in the cases of *AuthNAsymmetricKey* and *AuthNCertificate* as an idea of how could they be implemented, given the relation to the TLS protocol.

2.3.1. The *AuthNNone* Authentication Mechanism

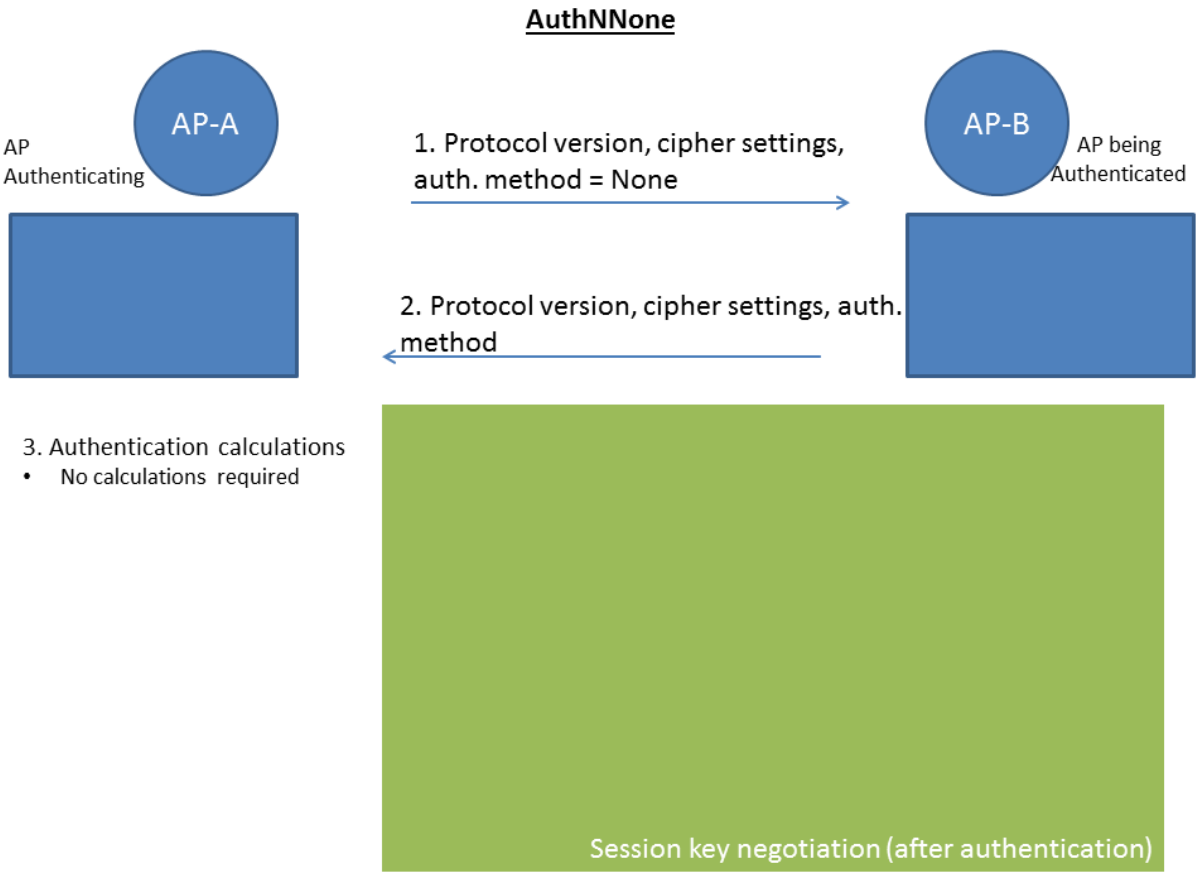

Figure 7. Detailed messages for the *AuthNNone* mechanism

[4] ISO/IEC 9798-3:1998 Information technology — Security techniques — Entity authentication — Part 3: Mechanisms using digital signature techniques

[5] ISO/IEC 9798-4:1999 Information technology — Security techniques — Entity authentication — Part 4: Mechanisms using a cryptographic check function

[6] ISO/IEC 9798-5:2004 Information technology — Security techniques — Entity authentication — Part 5: Mechanisms using zero-knowledge techniques

[7] ISO/IEC 9798-6:2005 Information technology — Security techniques — Entity authentication — Part 6: Mechanisms using manual data transfer

2.3.2. The *AuthNPassword* Authentication Mechanism

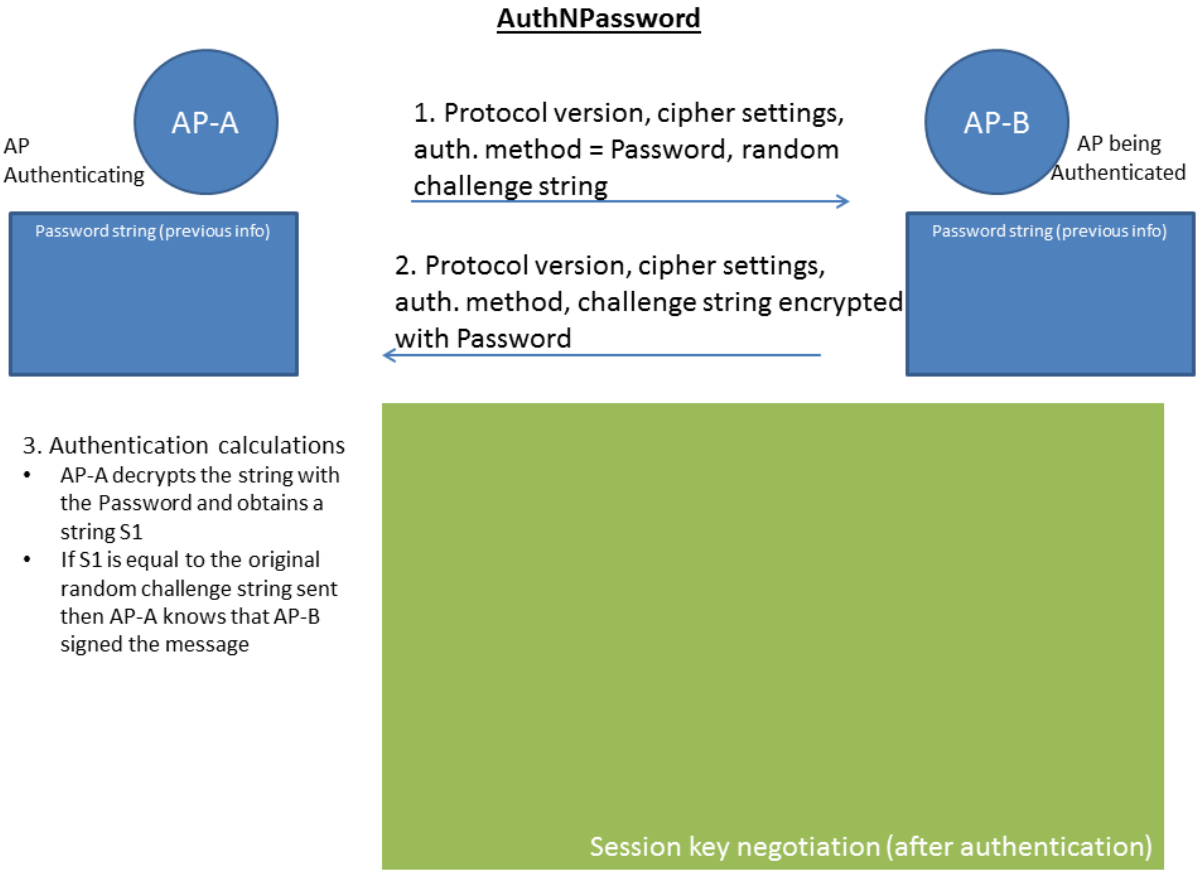

Figure 8. Detailed messages for the *AuthNPassword* mechanism

2.3.3. The *AuthNSessionKey* Authentication Mechanism

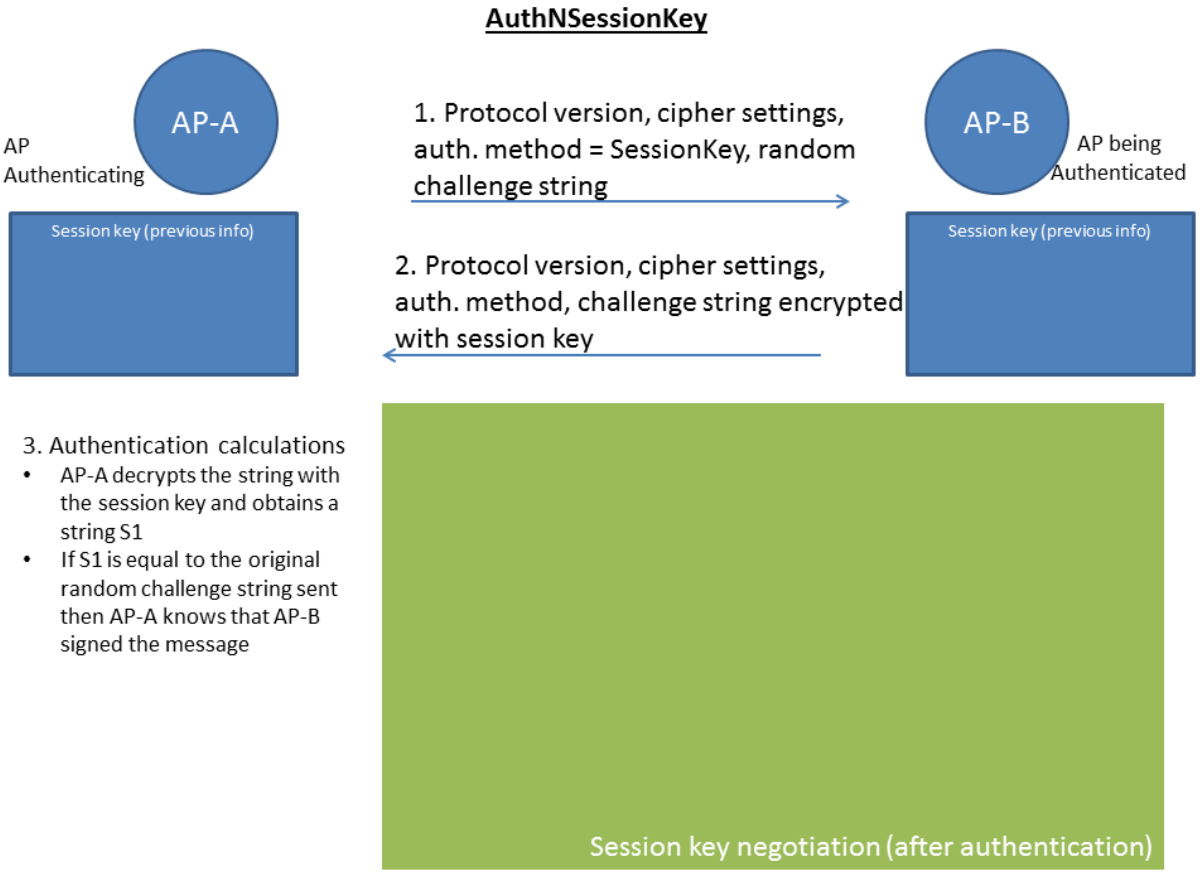

Figure 9. Detailed messages for the *AuthNSessionKey* mechanism

2.3.4. The *AuthNAsymmetricKey* Authentication Mechanism

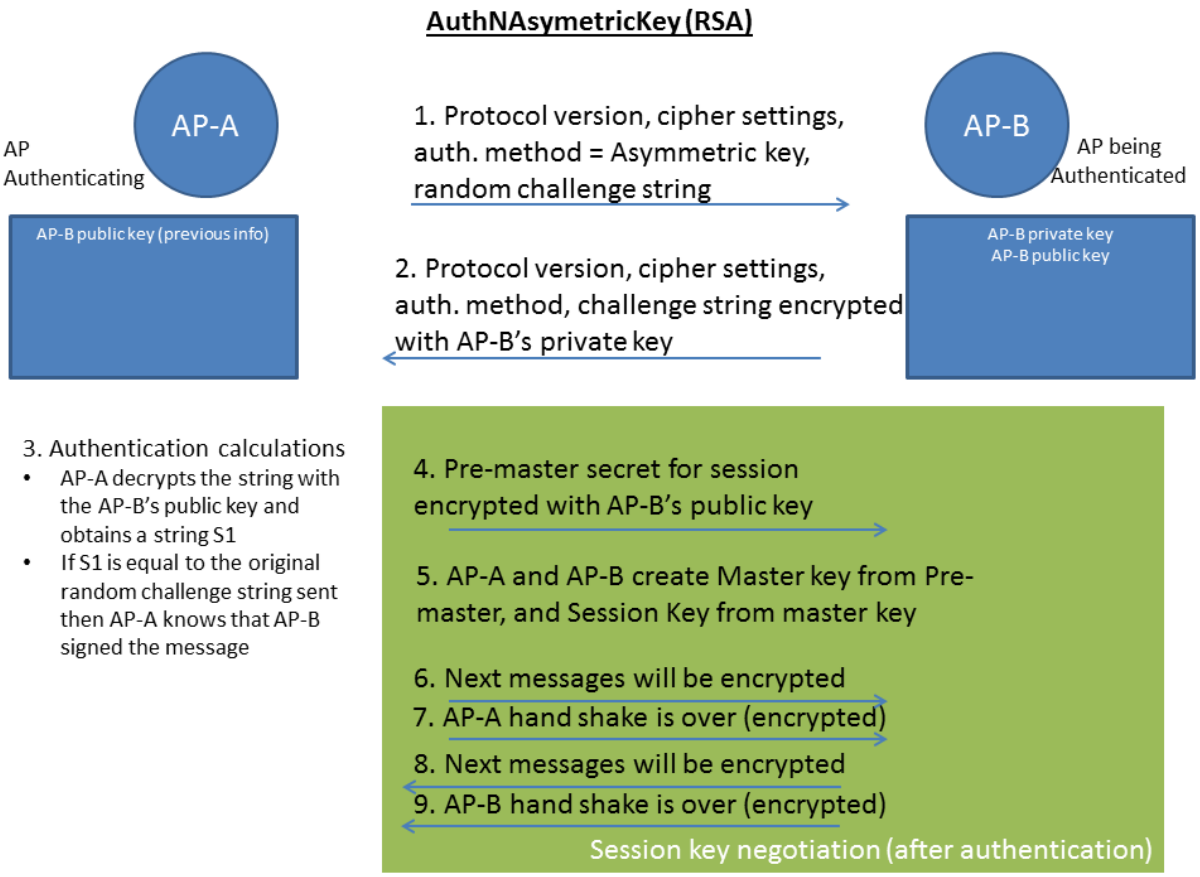

Figure 10. Detailed messages for the *AuthNAsymmetricKey* mechanism

2.3.5. The *AuthNCertificate* Authentication Mechanism

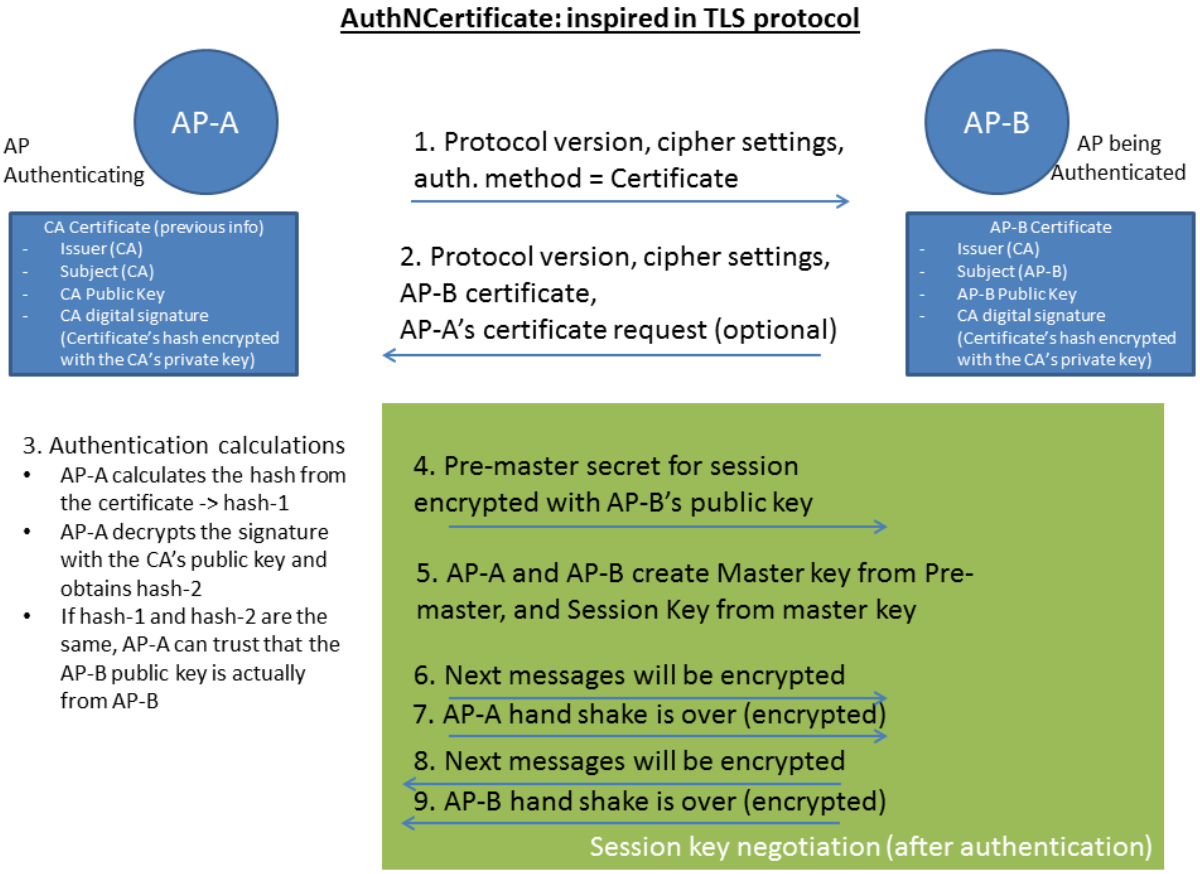

Figure 11. Detailed messages for the *AuthNCertificate* mechanism

2.3.6. The *AuthNToken* Authentication Mechanism

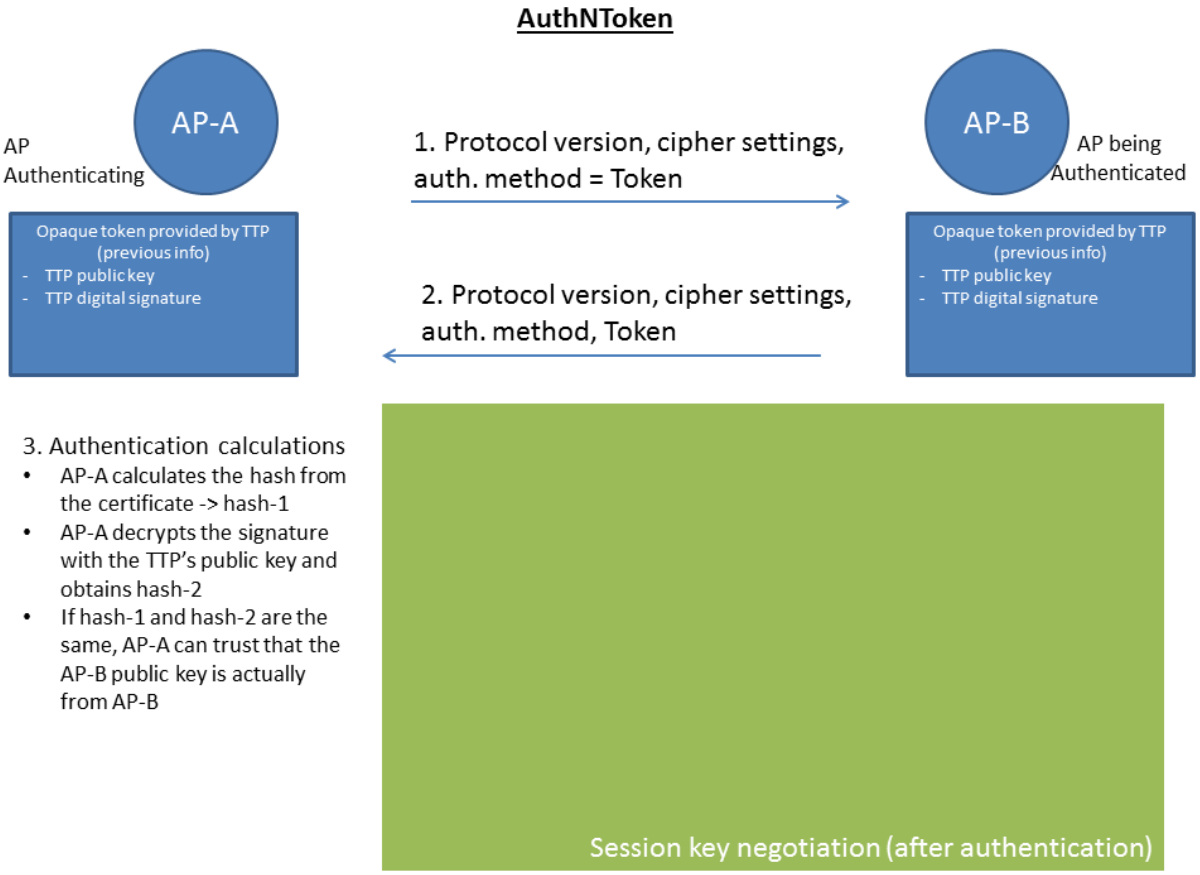

Figure 12. Detailed messages for the *AuthNToken* mechanism

2.4. Summary

The following table summarises the six different authentication methods described above. A decision will be made as to which authentication method will be implemented during the course of the project.

Authentication Method	Identity check	Direct / Indirect	Symmetric	Temporal
AuthNNone	No	-	-	-
AuthNToken	Yes	Indirect	No	Yes
AuthNSessionKey	Yes	Direct	Yes	Yes
AuthNPassword	Yes	Direct	Yes	No
AuthNAsymmetricKey	Yes	Direct	No	No
AuthNCertificate	Yes	Indirect	No	No

3. Access Management

3.1. Authorisation and Access Control

Authorisation and Access Control (AC) refer to the concepts of allowing/constraining a set of subjects (e.g. users, processes, roles, entities) accessing objects (e.g. data, files, contents). Authorisation defines whether the subject should be allowed or not to access the objects. Access control on the other hand manages this access at a very granular level.

The main objective is to prevent unauthorised information disclosure (confidentiality) and improper malicious modifications (integrity), while ensuring access for authorised entities (availability). An access control policy defines the rules that specify the conditions under which authorised subjects to have access to objects.

Authorisation and Access Control are performed following the authentication process, i.e., the process of confirming the identity of "subjects" in a system, as described in Section 2 of this document. A subject is assigned an "authorisation profile" in which the authorised actions on an object are specified in order to take an access controlled action.

3.1.1. Access Control Mechanisms

There exist a number of access control models including Discretionary, Mandatory, Non-Discretionary, Access Control List (ACL), Role-based, Attribute-based, Capability-based, Administrative, Distributed, Governance-based, Risk Adaptive, and so on. For a description of these AC types please see [Benantar2006]. In this chapter, we focus on two Access Control models that are relevant to our interests in realising them in RINA: Capability Based Access Control (CBAC) and Multi Layer Security.

ACL model uses a table where the rows represent subjects, columns represent objects, and the table elements represent a set of rights associated with the subject/object combination. In ACL-based methods a trusted entity must authenticate the identity of the subject and check the claimed rights against the ACL of the object.

However, when a centralised entity is used to hold large amount of entries, ACL may not be scalable. Moreover, it may suffer from security threats [Close2009].

Capability-based approach manages the access control through capabilities.A capability consists of a token that designates an object and grants the subject (i.e. the

holder of the token) authority to perform actions on that object. It is defined as: the name for identifying the object, and the set of access rights for that object. The capability could be seen as a ticket, if a subject does possess this ticket it has the proof of the holder's rights to access the object.

CBAC is defined to simplify administration of permissions for large numbers of users. It could be implemented either in the classical Role Based Access Control (RBAC) depicted in Figure 13, "Role based Access Control [ABAC]"or in the advanced Attribute Based Access Control (ABAC) illustrated in Figure 14, "Attribute Based Access Control [ABAC]". The capability is computed based on the role in case of RBAC and based on the attribute in case of ABAC.

RBAC models categorise users based on similar needs and group them into roles. Permissions are assigned to roles rather than to individual users. Its objective is to reduce the number of assignments. The more users and permissions one single role captures, the greater the administrative efficiency gains.

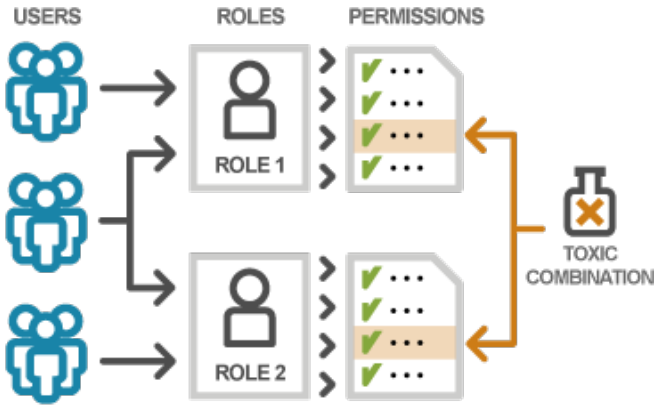

Figure 13. Role based Access Control [ABAC]

Ideally, users should be assigned permissions that at any point in time represent a true reflection of current business rules, risk mitigating precautions and context-related security measures.

ABAC approach defines the capability or the authorisation token as one of the attributes of the entity that requires access to a certain resource in the system. Where RBAC provides coarse-grained, predefined and static access control configurations, ABAC offers fine-grained rules which are evaluated dynamically in real-time.

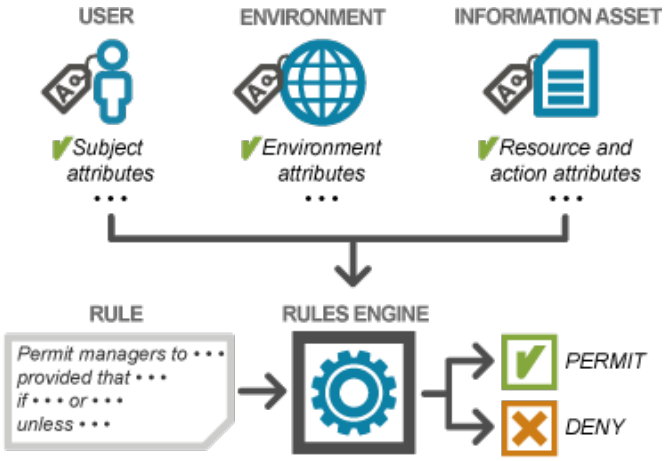

Figure 14. Attribute Based Access Control [ABAC]

3.1.2. Use of Capability Based Access Control in RINA

In the context of RINA framework, we consider the Capability Based Access Control as one of the possible approaches to offer access control to IPCs requesting to perform different actions on objects within a DAF/DIF. For CBAC, capability computation need to be performed either using roles or attributes. As a requirement to AC, it should be noted that IPC authentication to join a DIF and secure channel set-up for the exchange of initial credentials have been described in the different chapters of this deliverable.

Description of CBAC

In RINA APs/IPC processes are considered as subjects that are required to be authorised to proceed with some actions on the objects. Objects are the data and contents within the DIF/DAF that the AP/IPC processes are members. Basically, the CBAC will provide the corresponding capabilities to allow APs/IPCs get access to the required resources in the DAF/DIF.

Here we introduce specific IPCs called IPC-ACs for managing access control (AC manager) operating at the DIF level. The IPC-AC stores the profiles of IPC processes and their resources in an Access Control Information Base (AC-IB). The capability computation by “Authorisation Token Computation - ATC” as a function of IPC-AC will be based on a set of rules/policies that considers the attributes of a new IPC that is joined a DIF. This new IPC has been authenticated and by using the attributes of the targeted object, the ATC generates the permission to access the resource and provides the token to the newly joined IPC. Obviously, the IPC-AC must be created in advance prior to any request initiation. Without this IPC-AC no access is granted since there will be no token issued. It may be created by DMS.

Note that we consider that the new IPC profile is added/updated in the AC-IB just after the authentication. We need to define how the new IPC profile is generated when the new IPC has joined the DIF. For the time being, we consider a successful authentication allow the new IPC to be categorised in a group/role that already has a set of authorised actions. Thus, the new IPC will get those rights.

One approach that is being investigated to compute the token is the third party capability based approach inspired from Kerberos [Neuman2005], where an IPC will obtain tokens from a third party with specific properties on the access control actions related to a given object. This third party entity can be the Key distribution manager present at the DIF level (see Figure 15, "Access Control Architecture", IPC-Key DistribManager).

The AC Manager is associated with a Key Management (KM) function. For more details, please see the Key Management section of this deliverable. Two distinct KM options have been considered there: distributed and centralised architectures. In our architecture we consider the distributed approach where each DIF has its own Key distribution Manager.

RINA CBAC Architecture

Figure 15, "Access Control Architecture" shows an architecture employing a distributed approach where an Access Control Manager IPC (IPC-AC) with its associated storage information base is used per DAF/DIF. This IPC-AC is entirely responsible for the access control tasks within its DAF/DIF.

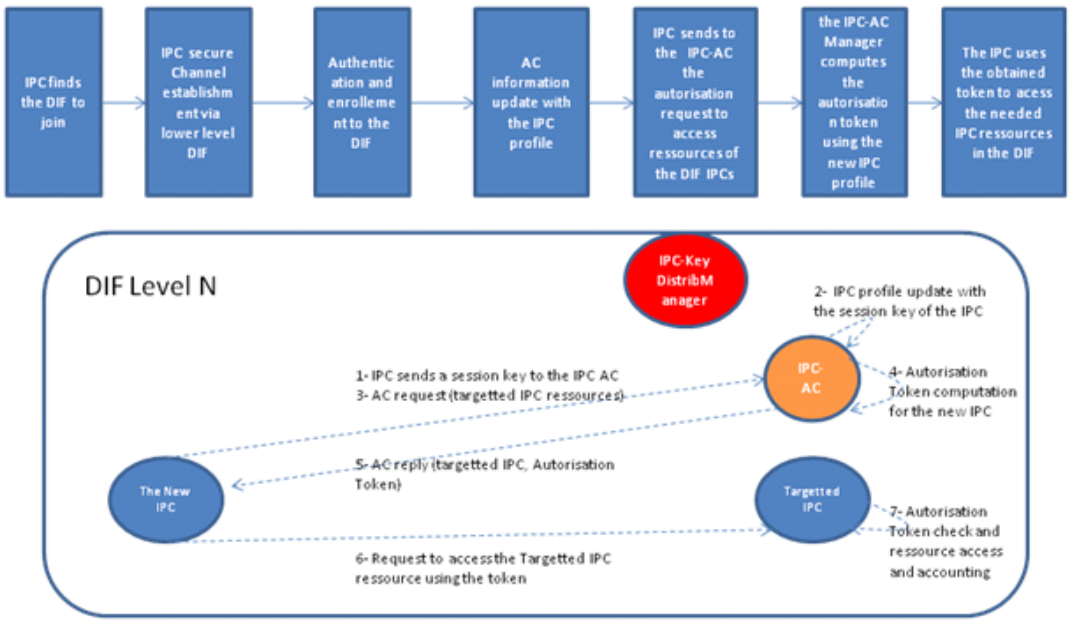

Figure 15. Access Control Architecture

A newly joined IPC that has already been authenticated to join the DIF asks the IPC-AC to obtain authorisation tokens to access the objects. The obtained token becomes one of the attribute of the new IPC profile. The key distribution entity is in charge of exchanging the key with the new IPC that is member of DIF after its authentication.

A token is generated upon request by IPC-AC based on the information of the profile of the user and the authentication information. The profile describes a set of objects and rights that the user can use. The profile of each user is defined and stored in the IPC-AC.

After the enrolment phase, and the authentication step, the IPC will be able to extract its profile from the corresponding IPC-AC and can ask for a token to use the requested printing service.

Based on the authorisation profile of the given AP/IPC, the IPC-AC generates a token to benefit from the required service. The access control policies are the rules applied by the APs/IPCs to access the objects. For instance, it could include the access type (read and/or write) and the time frame during which the AP/IPC is allowed to access a given data within the DAF/DIF but also other attributes.

3.1.3. Summary and Open Issues

Here we briefly explain authorisation and access control, focussing on the Capability Based Access Control model. There are open issues that need further investigation, including the following:

- IPC-AC deployment needs further elaboration to whether be part of DIF/DAF management entity, part of key management entity or completely separate entity.
- The way the authorisation profile is assigned to the IPC processes needs more investigation. It must be determined either it is assigned from the beginning of the IPC enrolment or negotiated during the session.

3.2. Multi-Level Security (MLS)

Multi Level Security is an important application area for content based security, and one of relevance in many use-case scenarios. MLS refers to protecting data or “objects” that can be classified at various sensitivity levels, from processes or “subjects” who may be cleared at various trusted levels. Such a model is appropriate in many high assurance applications, and is often mandated in organisational, governmental and national and multi-national contexts by policy.

The needs for MLS have emerged as organisations have had to deal with securing and protecting separate networking environments having different security classifications and possibly managed by different administrations. This separate networking model no longer supports the needs for real-time communication, situational awareness and rapid response to crisis in the modern communications era. With MLS, it is possible to provide document sharing within domains and across multi-domain interconnected infrastructures where each domain may be managed by a separate administration authority.

Here, given the existing MLS architectures, we investigate how secure data sharing can be achieved on the common RINA infrastructure.

3.2.1. Overview of MLS

Here, we provide a high level overview of Multi-Level Security and ways in which it could be applied within RINA. A strict definition of MLS includes a formal model of classification levels for data and clearance levels for user/objects, together with rules to prevent inappropriate access by users/objects to data/subjects at a higher classification level than their clearance. The levels refer to data that in some sense is more "valuable" or “sensitive” the higher the level it is in, and users who are in some sense more "trusted" to access the data the higher the level they are in.

The classic example is in many government and emergency services applications, where formal confidentiality classification labels are defined for data, as are formal clearances for users. An example hierarchy of classification labels could range from Unclassified (U), Protect, Restricted (R), Confidential, Secret (S) up to Top Secret. Users may be cleared to an equivalent set of levels, allowing them to read data up to and including that level but not above. This example will be reused throughout this paper for sake of illustration.

Such a model is appropriate in many high assurance applications, and is often mandated in organisational, governmental and national and multi-national contexts by policy. Common Criteria specifies the standard that defines assurance requirements that a system must meet, organised into seven levels from 1 (the lowest) to 7 (the highest) [CC].

Many security policy models have been proposed #[Anderson01], of which the most commonly used in an MLS system is the Bell-La Padula model [Gollmann05]#, commonly referred to as the "no read up, no write down" model (see Figure 16, “Bell LaPadula Multi-Level Security (MLS) Model.”). The "no read up" property is fairly self-explanatory, as it prevents users from reading data at a higher classification level than

their clearance. The "no write down" property means that users cannot write data at a lower classification level than their clearance, which prevents users accidentally or deliberately labelling data at a lower classification level than its true classification level. To make an MLS system practical, however, it is generally necessary to allow for at least some "write down" capability, as otherwise higher cleared users cannot create and share data that is readable by lower cleared users. For the same reason, this model would also lead to all data "flowing up" to the highest classification level over time as it is combined with other data to create new data. To this end, such systems usually include a "Trusted Downgrade" capability to allow this to happen.

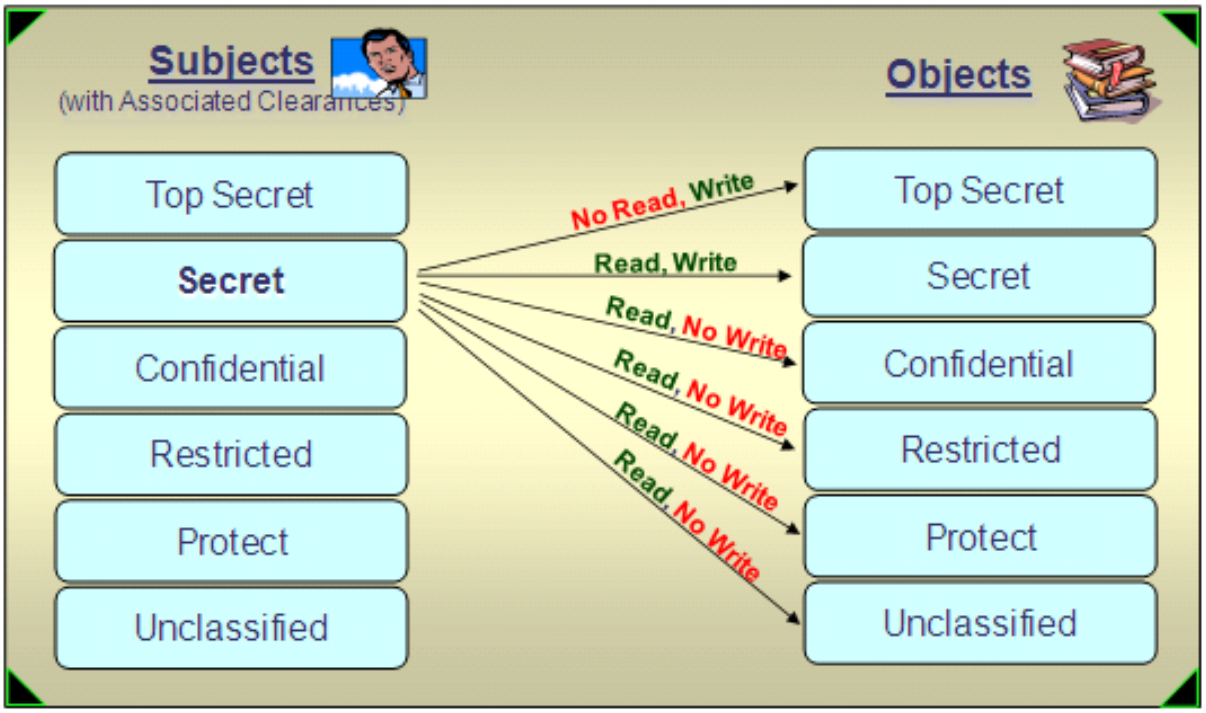

Figure 16. Bell LaPadula Multi-Level Security (MLS) Model.

In this MLS section, we explain MLS, MLS architectures, and how we can achieve MLS for content security given the RINA networking model. We investigate how secure data sharing without the loss of control over access to the data can be achieved on the common RINA infrastructure through configuration of RINA internals.

3.2.2. Three Facets of an MLS framework

Communications security

Generally speaking, MLS solutions for communications security are the most developed and tractable part of the MLS problem, which covers the end-to-end transfer of data between users or processes. From an MLS perspective, the requirement is to ensure that data cannot be inappropriately read from the communication channel (through

eavesdropping or accidental leakage into other communication channels or systems), and that data at different classification levels is not inappropriately mixed (i.e. the integrity of any classification labels is maintained during communication). This also includes authentication of the end points to ensure that they are suitable for accepting the data being communicated, based on its classification level.

Considering the Open Systems Interconnection (OSI) stack, the MLS communication solutions are characterised by the layer at which they operate:

- **Physical layer** – Communications can be separated by using different cables for different classification levels of data. However, this can be expensive and requires significant duplication of services and resources.
- **Link layer** – Link layer MLS solutions exist, and are particularly common in radio, although data is not protected at relay points (which are generally required for end-to-end communications).
- **Network layer** – Many high assurance network layer solutions exist, typically based on IPsec protocols #[IPsec]. The advantage of these over link layer solutions is that communications are protected while being routed over multiple links. However, data is still not protected at any intermediate storage points where e.g., IPsec tunnel is terminated.
- **Transport Layer** – The most common transport layer security solution is SSL/TLS. The advantage of such a solution is that it provides application-to-application security, and therefore allows a finer granularity of protection as opposed to the device-to-device security provided by network and link layer solutions. However, data is still not protected at any intermediate storage points where the transport layer session is terminated.
- **Application layer** – These solutions apply protection directly to individual items of data, and therefore offer the finest granularity of protection possible. Such technologies also protect data in any intermediate storage points.

MLS on a Single Physical Component

From an MLS point of view, the device and all software that runs on it (including OSs and applications) must as a whole prevent inappropriate access to or leakage of data that is processed or stored on it. They must also keep track of and maintain the integrity of classification labels associated with data and clearance levels associate with any subjects the device handles, such as processes, I/O devices, users, applications, etc. While RINA concerns inter process communication across physical components, rather

than protecting objects on a single physical device, we should be aware of the various MLS approaches in this regard. Depending on which approach is implemented, it may be possible, or even necessary, to have IPC Processes handling differing classification levels residing on a single physical component.

Single level devices can be constrained (through physical and procedural controls) to process data at one classification level only, and to permit users to have access only at that level. The main advantage of this approach is that Commercial off the Shelf (COTS) devices and their associated OSs and applications can be used with consequent cost savings. At the other extreme, a bespoke device can be built from scratch for a specific application, and designed to handle data and users at multiple levels (i.e. to be an MLS system) in the way that the application requires. This is for obvious reasons a very expensive approach and only used in special cases (e.g. for hardware crypto devices). Alternatively, an MLS OS may be used to formally label and maintains MLS separation of all subjects and objects, including end users and applications at multiple clearance levels, enabling standard applications to be used, thereby limiting the cost impact of high assurance development to the OS (and perhaps the device) [SELinux]. The use of a virtualisation layer, called a Virtual Machine Monitor (VMM) or Hypervisor, which sits between “guest” OSs and the resources on a device (I/O, memory, CPU etc.) allows multiple guest OSs and their associated applications to reside on the same device and share its resources, but to be isolated so that they cannot interfere with each other and treated as a single classification level device #[Shamon]. A similar effect can be implemented using thin clients, through which users access data and applications that have limited or no data storage or processing capability themselves but are connected to central servers that host the users’ data and applications, separated by classification level.

Trusted downgrade and boundary protection

To make an MLS system practical it is generally necessary to allow for at least some "write down" capability: for example, to allow higher cleared users to create and share data that is readable by lower cleared users. Clearly, this “write down” facility needs to be carefully controlled to prevent accidental or deliberate release of sensitive information by users or malicious code, and this is where “Trusted Downgrade” and “Boundary Protection Component” (BPC or “Guard”) products are used.

Trusted Downgrade is typically a facility provided within MLS OSs that allows highly trusted users, and perhaps applications, to modify the labels on data in special cases. This facility would typically be protected to high assurance levels so that the risk of malicious code exploiting it is very low. There primary methods used are:

- **Manual transfer** – This approach requires a person to check the true classification level of the data to be transferred, and to re-enter the data (perhaps suitably sanitised) into the low classification system manually. Clearly, this is a costly and inefficient solution. It is also subject to human error, depending on how complex the data is.
- **Label checking** – Where such labels exist, a BPC can simply search for them and ensure that release rules are adhered to. This can be effective against accidental release of sensitive data, but as the labels are not trustworthy, users or malicious code could deliberately mislabel data to bypass the protection.
- **Deep content inspection** – Another approach is to inspect all of the data to determine, through some knowledge of the data semantics, what its classification level is and/or that it does not contain hidden data. Examples include keyword searching of text in e-mails or documents, or the analysis of images to detect hidden data.
- **Content modification** – This approach aims to modify content to remove potential ways in which sensitive data can be leaked within it. Generally, these techniques concentrate on the protocols used to transport the data, rather than the data itself, and are aimed at limiting or eliminating the possibility of covert channels. A “protocol break” BPC is a common approach, where the BPC acts as a proxy for the protocol. It will terminate the protocol and re-encode protocol messages according to its own rules and interpretation of the original message. Note that although confidentiality is the focus here, protocol break BPCs can be very effective in protecting the integrity of the high system from messages sent from the low system. In particular, malformed protocol messages and buffer overflows can be effectively stripped out by this approach.

Note that a special form of boundary protection can be provided by one-way diodes [LINK]#, which allow data to flow from a low to a high system and prevents any possible covert channels in the opposite direction. Of course, this does not allow "write down", but can be useful in some cases to allow a more automated flow of information into a high system. Such diodes can be produced to very high levels of assurance, but in practice can only be used to mirror data from low to high systems rather than allowing application to application transfer.

3.2.3. MLS architectures

In this section, we describe existing MLS architectures, before considering their provision in RINA in subsequent sections.

Multiple Single Levels (MSL)

Historically, the most common way to deal with the MLS issue was, in fact, to sidestep it by implementing multiple parallel systems where each individual system handles a single data classification level. Such an approach is often referred to as "Multiple Single Levels (MSL)" and is illustrated in Figure 17, “An example MSL Architecture”.

Figure 17, “An example MSL Architecture” shows U, R and S LANs. Each terminal, server or storage device is connected to one, and only one, LAN, and the device and all data on it is treated as if its classification level is that of the LAN. Remote LANs at the same level can be joined through a common core network, with cryptographic devices used to protect and separate data on it. A user who needs access to U, R and S data and systems has to have access to three separate terminals.

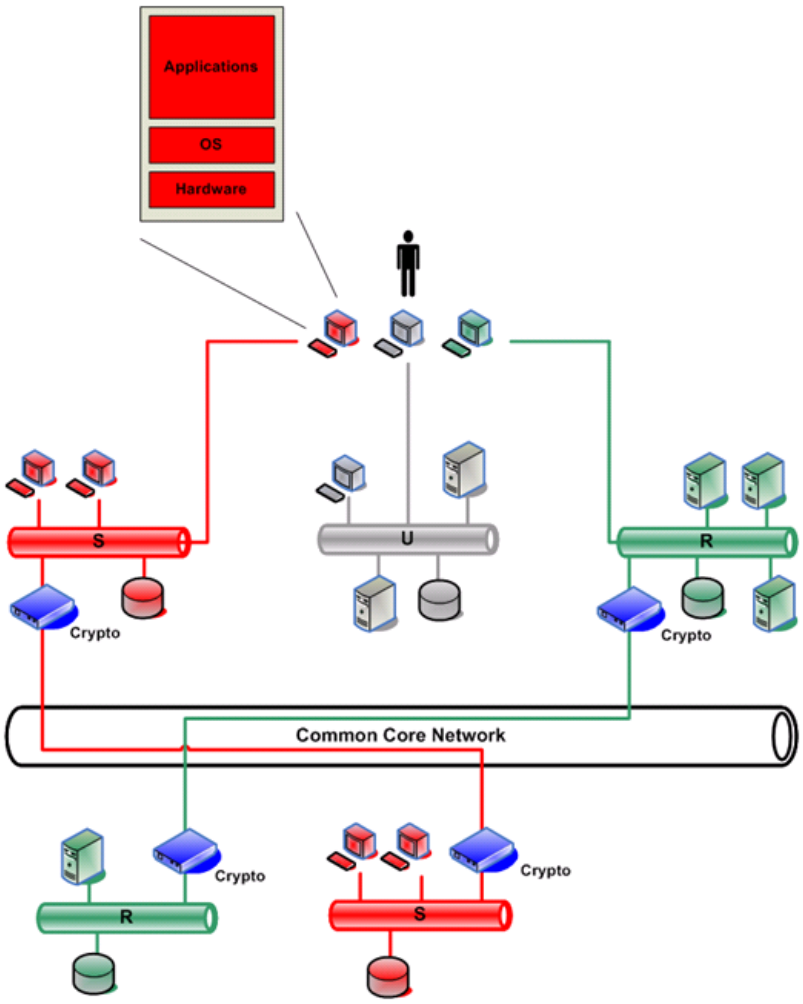

Figure 17. An example MSL Architecture

For each single level system, only users who are cleared to that level (or higher) are allowed access, and all data within the system is treated as if classified at that level. The only way to import or export data from the system is by manual transfer. In so-

called "System High" mode, each system can actually contain data classified up to and including the system's classification level, which gives more flexibility to the individual systems. However, any data exported from a system will always be treated as if it is at the (high) classification level of the system regardless of its true classification level.

The main advantage of this approach is that data leakage can be prevented effectively by using strong physical and procedural controls, as physical access is required to a system in order to import or export data. However, this approach has two significant disadvantages. Firstly, practicality in terms of cost, space and flexibility can be severely affected as the number of levels that need to be supported increases. Essentially, every time a new level needs to be introduced a new set of computing systems and networks have to be purchased, deployed and supported. By using strong cryptographic separation, a shared core network can be used to communicate between physically separate parts of each single level system, which can reduce the impact of adding levels from the network point of view. However, additional end devices are still required, and can, for example, lead to the situation of users having large numbers of terminals on their desktop. Secondly, this approach makes it very hard to share data between systems. It also only supports a data "push" model, as a user only cleared to R for example has no way of finding out if any (R or lower) information on an S level system exists that might be of use to them.

Domain Based Security

To get around some of the data sharing restrictions of an MSL approach, limited interconnection between systems at different levels can be allowed if protected by BPCs. This is the so-called "Domain Based Security" approach, which is currently the most commonly implemented approach to MLS. The name comes from a particular modelling technique where business needs are used to group users, applications and data into "domains" within which relatively free transfer of data is required. These are then mapped onto the underlying (MSL) infrastructure, and the need for communication between different levels, and hence BPCs, is highlighted.

This approach is illustrated in Figure 18, "An example domain based security architecture", which is the same overall system as in Figure 17, "An example MSL Architecture" but with the addition of a communication channel between the R and U systems and an associated BPC (shown as a firewall in this Figure 18, "An example domain based security architecture"). The main advantage of this approach is, of course, that it allows (in theory at least) more automated sharing of data between systems at different levels. Another advantage is that it can significantly reduce the overall cost of systems, as less duplication is required. For example, if an R system can

now send e-mails to a U system, a user on the R system may no longer need both an R and a U terminal.

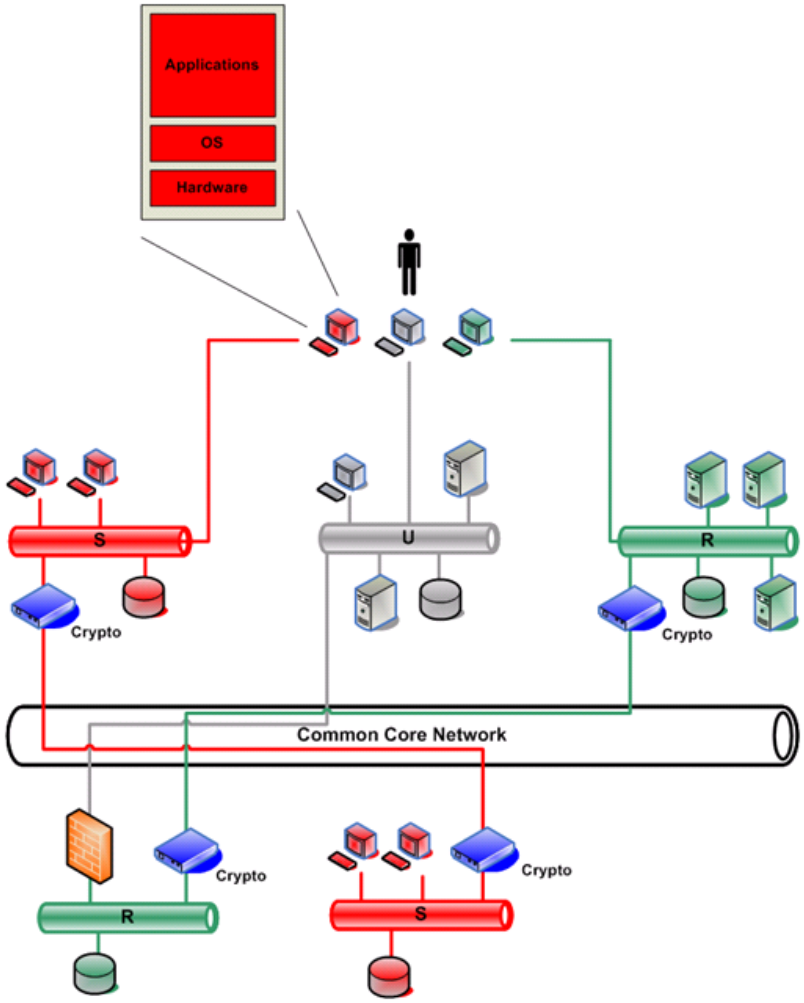

Figure 18. An example domain based security architecture

However, in practice this approach has been found to have significant disadvantages. The main issue is the difficulty of producing BPCs. The level of assurance in BPCs is relatively low, and therefore their use is often restricted to systems that have a relatively low classification level, and between systems whose classification levels are close together (e.g. an acceptable BPC between R and U is often possible, but highly unlikely between S and U). Their use is also often restricted only to certain types of applications, such as e-mail, due to the difficulty in producing BPCs for some kinds of data flows (e.g. video streaming). In addition, as the number of required application data flows increases, the number of BPCs that need to be developed, purchased, deployed and maintained also increases (as BPCs tend to have to be highly application-specific). This has cost implications, and also leads to inflexibility as it is difficult and time-consuming to add new application data flows.

MILS

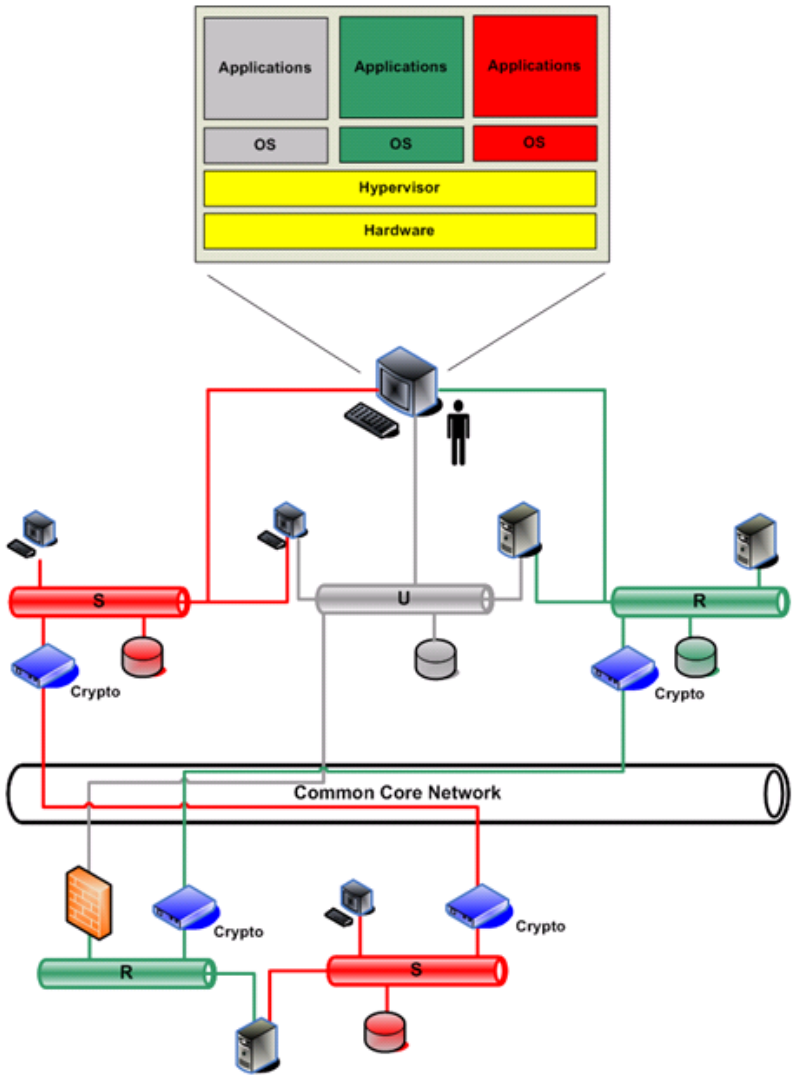

Figure 19. Example of an MILS based architecture

The Multiple Independent Levels of Security (MILS) approach can be thought of as a virtualised implementation of the Domain Based Security approach, as fundamentally it consists of MSLs interconnected by BPCs. The difference lies in the fact that multiple levels can coexist on the same physical devices, with separation enforced between them by software on the device. The OS virtualisation approaches, described previously, are often used to implement MILS, and this is illustrated in Figure 19, “Example of an MILS based architecture”. The same LANs, BPCs and cryptos that are used in Figure 18, “An example domain based security architecture” are also used in Figure 19, “Example of an MILS based architecture”. However, each terminal, server or storage device can now be connected to multiple LANs. A user who needs access to U, R and S data and systems need only use one terminal. Some single level devices may still be used, for example storage devices as illustrated in Figure 19, “Example of an MILS based architecture”.

MLS OS

The other main approach to MLS is based on MLS OSs. In fact, this approach supports many different types of device, some of which may be single level, low assurance devices, but some can be high assurance MLS OS devices. The key point is that those applications and users that require access to multiple levels simultaneously can be supported on the same hardware. This is illustrated in Figure 20, "An example of MLS OS Architecture".

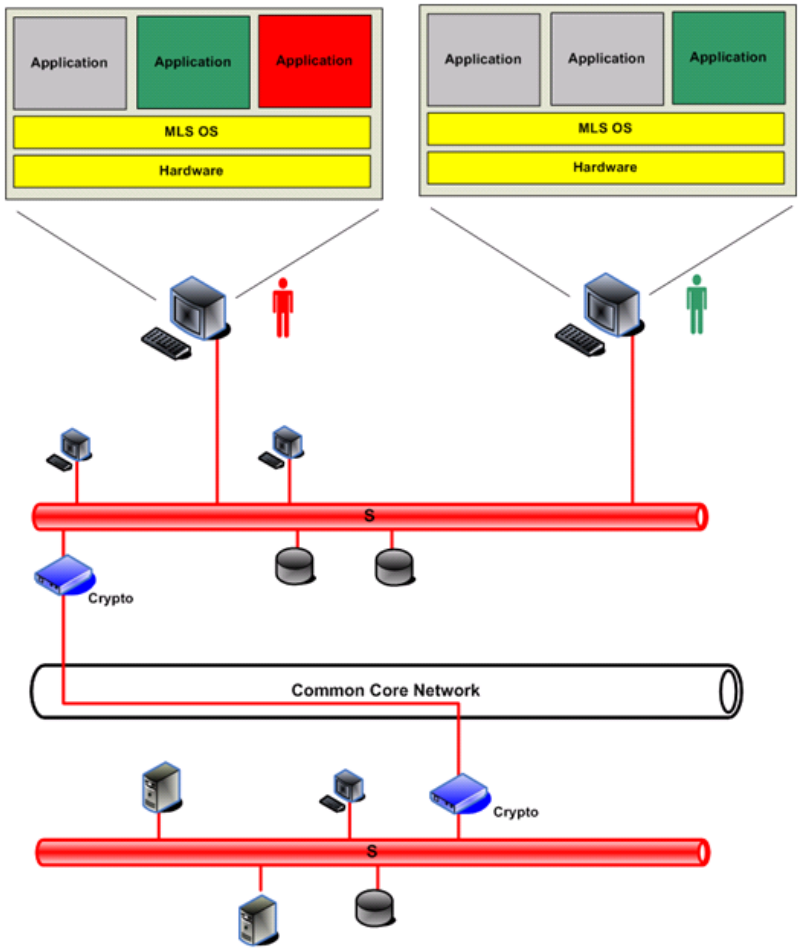

Figure 20. An example of MLS OS Architecture

In Figure 20, "An example of MLS OS Architecture", there is only one level of LAN and one level of terminal/server/storage device, both are able to carry (labelled) data at U, R and S. Users can only get access to data through terminals, and the MLS OS restricts which applications and data can be accessed depending on the clearance of the user. An S (red) and R (green) user are illustrated. Note that a user can, for example, run applications at lower levels, but cannot run applications at higher levels. For example, the R (green) user in Figure 20, "An example of MLS OS Architecture" can run U (grey) applications but not S (red) ones.

An advantage of this approach is that duplication of equipment at multiple security levels is no longer required. A device with an MLS OS can be accessed by any user or application at any clearance level, and that user or application will be able to search for and read all information in the system that they are cleared to see. This approach therefore offers a “pull” model for access to data, and hence the most flexible data sharing as well as the least use of equipment. Currently, the main disadvantages of this approach are practicality in terms of lack of application support, and the costs associated with the need to modify applications and with the MLS OS itself.

In terms of “write down” capability, users and applications are generally not allowed to do this, and hence data that they create will be labelled at their clearance level, even if the data is actually of a lower classification. Having said that, facilities for trusted downgrade are provided so that trusted users and applications are able to “write down” if required. In addition, the classification labels on data are preserved, so that, for example, an S user who reads some R data is free to further distribute it to R users. In the other “System-High” approaches, an S user who has imported R data would then not be able to further distribute it to R users (the data is reclassified as S on import as the System-High system is not trusted to maintain its true label).

This preserving of labels reduces the problem of all data “flowing up” to the highest classification level over time. It does not completely remove it, however, as fusion of data still creates the problem of how to decide the classification level of the new, fused, data. The secure way would be to take the maximum classification of all data used as an input to the fusion process; however, this can again lead to unnecessary flowing up of data to the highest classification level over time.

3.2.4. Achieving MLS in RINA

We consider in this section how RINA may help facilitate MLS across distributed networks. Fundamentally, RINA concerns inter-process communication, and thus the communication security facet of MLS is most relevant. Where MLS is applied at the various layers of the OSI Model and TCP/IP layers, it is pertinent to consider how MLS applies within each DIF in a RINA architecture, which enables a consistent set of protocols across all DIF layers.

For example, IPsec is extended to support data labelling to facilitate MLS, identifying the sensitivity of the data as it traverses through the network. A security label may be applied to data explicitly, by transmitting the label in the packet carrying the data, or implicitly, by deducing the classification of the data from characteristics of the channel over which it is communicated (e.g., port number, source/destination, key used for

encryption). Access control policies utilise the labels to prevent unprivileged users and systems accessing or transmitting sensitive data. The Authentication Header protocol, of the IPsec protocol suite, ensures the integrity of the data origin of an IP packet. Where such a header includes the security classification of the data contained within the packet, the label may be protected against unauthorised removal or modification of the security label associated with the data being transmitted. The Encapsulating Security Payload (ESP) protocol, of the IPsec protocol suite, can be used to ensure the confidentiality of the data being transmitted by encryption. Tailoring key management and access control policies to utilise labels enables MLS solutions to be implemented.

Thus labelling data with its security classification, protecting the label against malicious modification/removal and restricting communication based upon the classification of data and clearance level of users, application processes, IPC Processes, DAFs and DIFs is of primary interest in this section.

MSL

Application Layer – Inter-DIF SDU Protection

The Multiple Single Levels (MSL) approach, of implementing multiple parallel systems where each individual system handles a single data classification level, is certainly achievable in RINA. All applications on the same DAF could be required to handle a single data classification, as illustrated in Figure 21, “MSL at DAF in RINA”.

In each of the example MLS architectures in Section #4, collections of processes of a certain classification communicate over a common core network shared amongst classification levels. In Figure 21, “MSL at DAF in RINA”, the level 1 DIF connects all hosts via a single router acting as the untrusted common core network in this case. Underlying DIF1 are four level 0 DIFs, each one connecting a single host to the router (omitted from the figure for clarity).

As in Figure 17, “An example MSL Architecture”, networks of shared resources of the same classification level are connected via the common core network, but communication between systems of differing classifications is denied. In Figure 21, “MSL at DAF in RINA”, AP3 and AP4 that reside within DAF2 share the same classification, handling Unclassified data only. Similarly, AP1 and AP2 share a common classification in DAF1. No DAF exists to support communication between APs of differing security clearances; this may be enforced by policy dictating that only APs of the same clearance can join the DAF. This can be applied by DAF through its enrolment protocol. When this is to be applied to DIFs (i.e. a DIF has a specific classification

level), the IPC Processes must enforce that new members to have the same classification level, and authentication and access control policies must take this into account. Thus a DAF/DIF may be said to adopt the clearance that is common to its members and data labelling may be implied from the DAF across which the data is communicated. Acting as a secure container, each application process is assured that all other processes in the DAF are authorised to handle data of the same classification; it must be confirmed that the joining IPC Process is cleared to access such data by the IPC Process to which it joins the DAF. *How to establish, authenticate and attest to the security clearance of an AP/IPC Process remains an open question.*

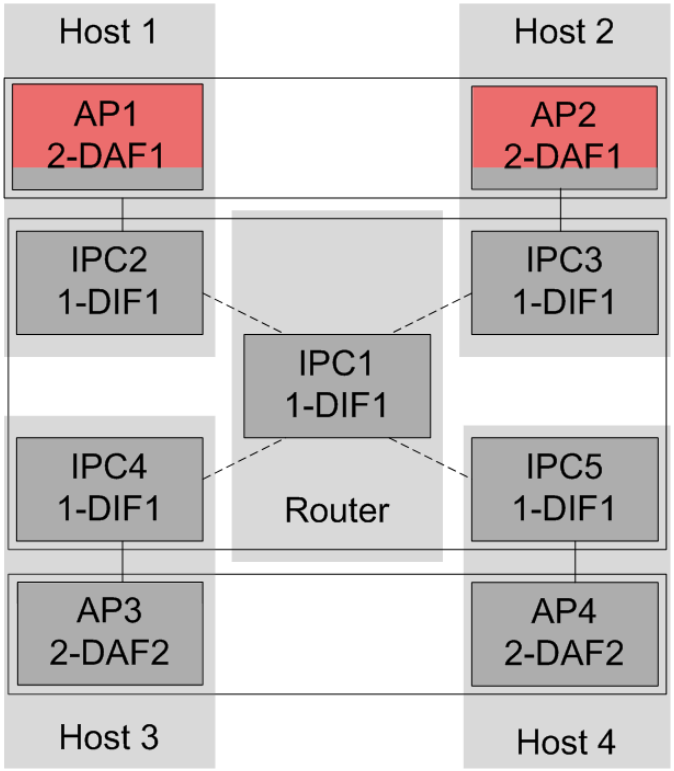

Figure 21. MSL at DAF in RINA

While all applications within DAF1 are able to handle Secret data, they may be required to protect data labelled Secret, when communicating across the DAF over the underlying DIF1 that acts as the untrusted common core network. Applications Processes (or IPC Processes) are ultimately responsible for ensuring the confidentiality and integrity of the SDUs they pass to the lower DIF. Therefore, if they use a DIF that is not trusted, proper SDU protection mechanisms have to be put in place. For example, an encryption policy that encrypts the application SDUs of AP1 and AP2 before passing them to DIF1 (and decrypts them at the other side, right after reading the SDUs from the DIF).

Alternatively, encrypting SDUs may be superfluous should the processes, cleared to access data up to security classification Secret, wish to communicate Unclassified data

(e.g., AP1 and AP2 in Figure 21, “MSL at DAF in RINA”). In this case the application may act as its own boundary protection component, initiating flows over lower classified DIFs without enciphering the SDU only when communicating Unclassified data over the flow.

Cryptos – Delegating SDU Protection to Trusted Middleboxes

Figure 22, “MSL at DIF in RINA” extends the scenario illustrated in Figure 21, “MSL at DAF in RINA”. Host 1, Crypto 1, Crypto, and Host 2 are trusted entities with processes classified at R level. Here, IPC13 and IPC14 at 2-DIF5 perform the encryption task previously done by AP1 and AP2 in the level 2 DAF, which has now become a level 2 DIF including two additional level 2 IPC Processes residing on physically separate hosts. All IPC Processes within the level 2 DIF5 are classified to the same level. The advantage is that existing applications can be used without any modification. However, IPC13 and IPC14 provide protection of SDUs passed between them on flows each instantiate on DIF1 immediately below them by enciphering data sent over a DIF with a lower security clearance, over which only Unclassified data is communicated. IPC13 and IPC14 are responsible for encrypting data within Crypto 1 and Crypto 2, respectively, reflecting the two cryptos protecting transmission between the Secret LANs illustrated in Figure 17, “An example MSL Architecture”. Alternatively Unclassified data passed between the applications may be communicated in the clear, or Unclassified data may be extracted from secret data and communicated in the clear, if this suffices for the particular application.

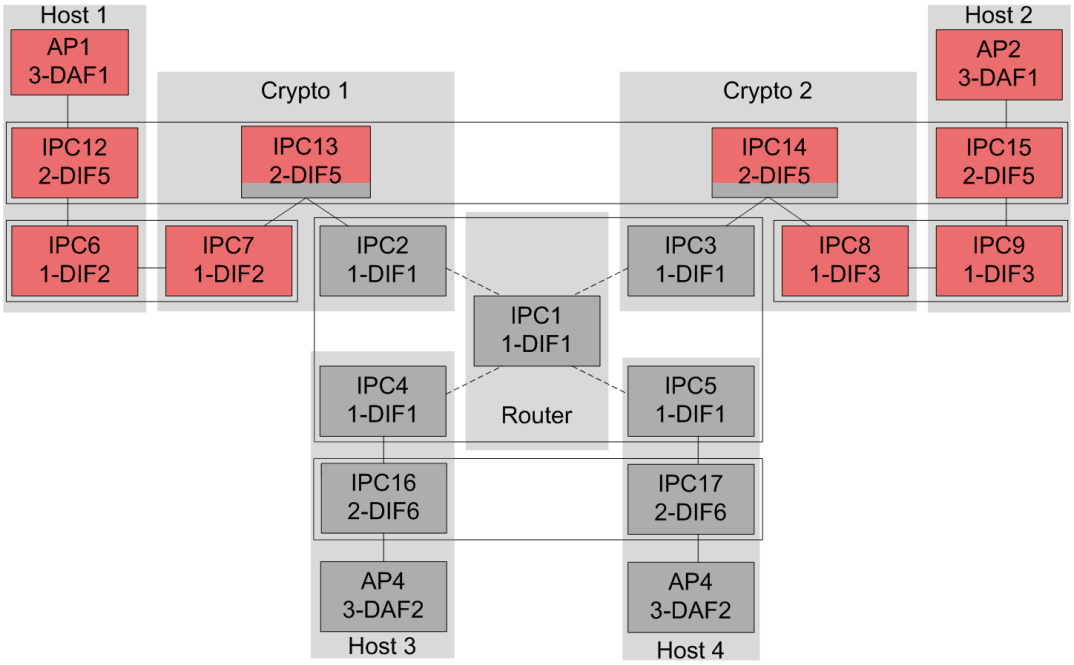

Figure 22. MSL at DIF in RINA

The advantage of this approach over the previous application layer scenario is the separation of SDU protection and access control responsibilities from the higher level application function. Multiple higher level DIFs, DAFs, applications and IPC Processes need not be concerned with protecting data communicated over the lower classified components residing in the layers below. This avoids duplication of functionality and affords the potential to produce dedicated physical components to perform the necessary protection processes at the boundary between classification levels. Further consideration of the mechanisms required of each IPC Process and DIF to support policies based on security classification of data is required.

MSL solutions to MLS, in which data cannot pass between classification levels, mirrors the view of DIFs in RINA being securable containers, providing the necessary separation of data and responsibilities. The view of DIFs as securable containers is not so befitting of the Domain Based model, which allows policy-controlled communication between processes of differing classification levels.

Domain Based

In this section, we consider the case in which limited communication between processes of differing clearance levels may be implemented. In this regard, we must consider the requirements imposed on the functionality of AP/IPC Processes within a DAF/DIF in terms of both SDU protection and access control.

Application Layer – Intra-DIF Access Control

Consider a case in which an application privy to data up to Restricted resides in a DAF with an application only authorised to access Unclassified data, i.e., AP3 and AP4 in Figure 23, “An example MLS architecture in RINA over a Common Core Network”. AP3 may be required to ensure that no data leaks between classification levels. The Restricted data on which AP3 may perform some processing, possibly in combination with data communicated up a classification level from AP4, must not be communicated in the reverse direction.

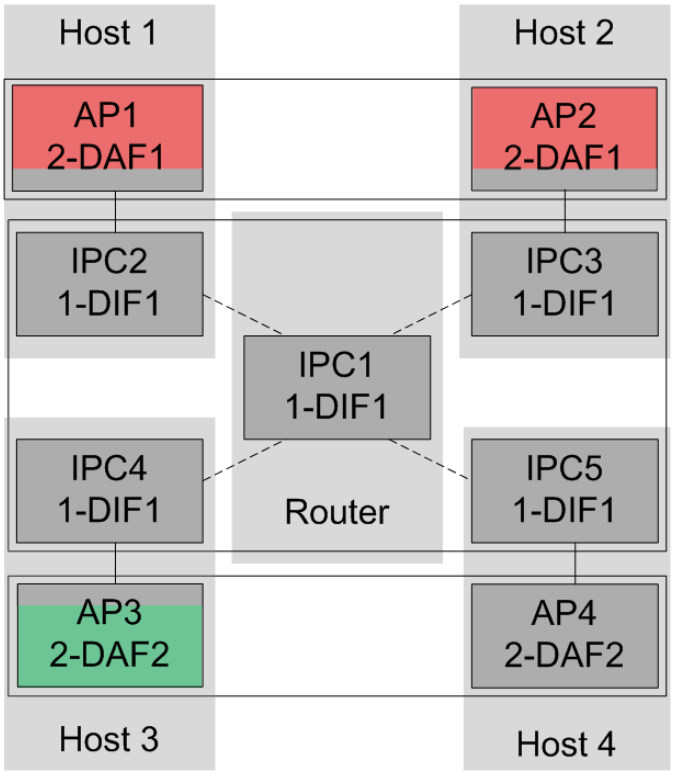

Figure 23. An example MLS architecture in RINA over a Common Core Network

Unlike in the MSL architecture of RINA, here DAF/DIFs no longer handle data of a single classification. The security label indicating the classification level of the data is no longer implied by the DAF/DIF over which the data is being communicated. The label may instead be inferred from its source/destination. It should be noted that here, the clearance level of each AP in the DAF should be recorded by and used to determine whether data is of an appropriate classification to be communicated to the destination. *It remains an open question how to establish, protect and maintain such clearance levels within an AP/IPC Process and across the DAF/DIF.*

It could be said, that the applications here are enforcing their own boundary protection. However, it is intended that common communication protocols be utilised 'recursively' at any layer of the architecture. Should lower layer DIFs, or at least IPC Processes residing therein, be able to enforce access control and SDU protection then there is the potential to avoid duplication of such mechanisms at the application layer. This also affords the possibility of delegating such responsibilities to separate components designated specifically for this purpose of boundary protection, as discussed below.

Boundary Protection Components - Delegating Down Access Control

It is advantageous in avoiding duplication of functionality and leveraging dedicated components not just for encryption/decryption, but also access control and trusted downgrade required in an MLS environment. IPC17, illustrated in Figure 24, "An MSL

Architecture in RINA with Cryptos", is responsible for ensuring that data is released (or operations are performed) between classification levels appropriate to the security policies adopted in the architecture.

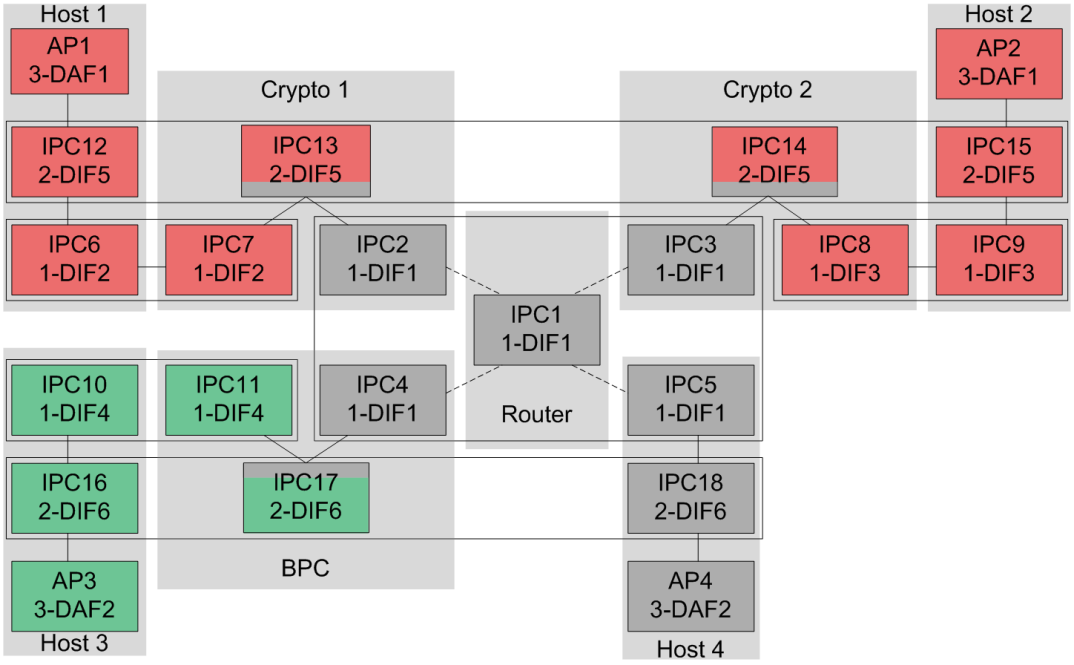

Figure 24. An MSL Architecture in RINA with Cryptos

Not only must security clearance of the AP/IPC Processes across a DAF/DIF layer be recorded, as in the previous example, security classifications of data must also be communicated down the DIF layers in this scenario if Host 3 communicates with Host 4 sending grey data. Applications Processes (or IPC Processes) are ultimately responsible for ensuring that data is not leaked to a lower classification level, so must be responsible for recording the classification of the data it stores and produces. When data is communicated via a lower level DIF and it has been decided to delegate boundary protection based upon security labelling to a separate physical component, SDUs must be labelled with the classification of the data they carry such that the IPC Process at the boundary (e.g., IPC17) can accurately control the flow of information between classification levels as dictated by policy. *How and where to include classification levels/security labels of data in an SDU, and the mechanisms required to protect them, remains an open question.*

Security Clearance of Joining IPC Processes

When DIFs include only IPC Processes of a single classification it must be ensured that only IPC Processes of the same classification can be successful in the joining process. However, Figure 24, "An MSL Architecture in RINA with Cryptos" illustrated a scenario in which IPC Processes within a DIF were authorised to access data of different

classification levels. As yet, it has not been considered what admission to the DIF should mean for the security classification of the joining IPC Process this case.

When a new IPC Process wants to join a DIF, it first needs to establish a flow to one of the DIF members. The new member and the DIF member need to share an N-1 DIF in common, through which the flow will be established. Subsequently an application connection is established requiring the joining IPC Process to authenticate itself. If the application connection is successfully established, the existing member will decide whether the new member is admitted to the DIF or not (access control).

In previous sections we have considered the types of protection mechanisms that may be required of IPC Processes when residing in a DIF of IPC Processes of differing security clearances. Should an IPC Process already existing in the DIF be cleared to handle data up to Restricted, then it may be entitled to attest to the joining IPC Process having a classification up to Restricted but probably not above. Potentially, it may be required that IPC Processes be joined to a DIF only by IPC Processes at the same classification level, or maybe a procedure for IPC Processes to escalate the security classification of a lower classified IPC Process within the DIF should be specified.

Protecting RIB Data

Here security “labels” can be assigned to the different RIB objects. Remote operations on the IPC Process RIBs are another aspect where the access control function is pertinent. In an MLS setting it may be required that the six CDAP operations (create, delete, read, write, start, stop) be restricted when invoked by another IPC Process, perhaps adopting the "no read up, no write down" model.

A key attribute that we wish to maintain in the RIB is the clearance levels of the AP/IPC Processes within the DAF/DIF. This is particularly sensitive information. It must be ensured that no lower level IPC Process can maliciously escalate their security clearance within the DIF. Mechanisms by which such threats are mitigated against require further consideration. It should also be considered precisely what data stored within the DIF requires protection from IPC Processes of differing security clearances.

Alternative routes for APs to Communicate

Another pertinent situation to consider is one in which there are multiple underlying DIFs and thus multiple communication routes exist between applications in the DAF. Figure 25, “MSL and Routing in RINA”, illustrates one such scenario in which AP 1 and AP 2 on a level 1 DAF, that may communicate via IPC1 by initiating a flow in either DIF 1 or via IPC6 by initiating a flow in DIF 3. It should be noted that the diagram

in Figure 25, “MSL and Routing in RINA” is simplified for readability reason. Such a network has the advantage of resilience and load balancing, if DIF1 goes drops out or is subject to high amounts of traffic, the alternative route can be used instead. However, the choice of flow is of consequence to whether AP1 must encipher the SDU or not. As both IPC Processes in DIF1 are privy to data classified up to Secret, the secret data to be passed from AP1 to AP2 via this route need never be encrypted at any level. However, AP1 is required to encrypt SDUs communicated to AP2 when transmitting sensitive data via flow through DIF3 and DIF2.

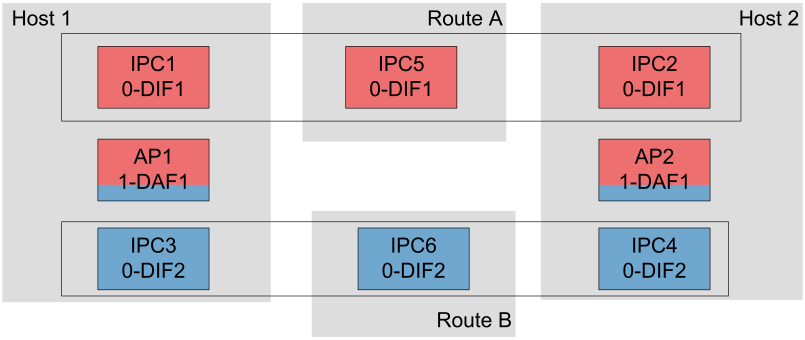

Figure 25. MSL and Routing in RINA

MILS and MLS OS

Having already considered how the MSL and Domain based MLS architectures could be achieved in RINA, little else is required of RINA in terms of connecting AP/IPC Processes protected by MLS solutions that support the use of data at different security classifications on a single device by virtualisation (or using thin clients), i.e., the MILS architecture illustrated in Figure 19, “Example of an MILS based architecture”. Each VM would handle data at a single classification level and contributes IPC Processes to DIFs in which either (i) all IPC Processes also act only on data at the same classification level, i.e., RINA MSL, or (ii) IPC Processes of differing security clearances reside on the same DIF, i.e, RINA Domain Based MSL.

There are two views one could take regarding how RINA relates to the MLS OS example illustrated in Figure 20, “An example of MLS OS Architecture”. Firstly, we could consider only how to achieve the communication functionality of LAN and WAN networks. In the example illustrated, each LAN is able to communicate even the most classified data, due to the protection afforded by the OS. Encryption is still employed for communications across the WAN. We have already seen how this can be achieved in RINA, communication between Host 1 and Host 2 in Figure 22, “MSL at DIF in RINA” provides a sufficient illustration.

RINA is a general approach to inter-process communication within and across networked devices. However, it could be considered how to use RINA concepts to implement MLS in a single network device.

3.2.5. Summary and Open Issues

Here we described different architectures for implementing MLS in a general network and mapped these architectures to RINA.

MLS perhaps muddies the glass of the perspective that DIFs in the RINA architecture are securable containers, since we may wish to ensure that data is appropriately protected even within the secure container should IPC Processes of differing security clearances reside within the same DIF. This domain based MLS architecture that is most pertinent for RINA raises many questions, as listed below. Many questions arise even when applying RINA to MSL, which is more befitting of the view of DIFs being securable containers but is not true MLS due to the inability to communicate between classification levels (i.e., the information flow for data sharing). We paid most consideration to these two MLS architectures in relation to RINA, as MILS and MLS OS were considered special cases of the former.

- MSL - All AP/IPC Processes within a DAF/DIF are cleared to handle data of the same security classification. The security classification of the data communicated within the DAF/DIF is implied by the common security clearance of the AP/IPC Processes that reside within it. SDU protection is required when communicating between the AP/IPC Processes within a DAF/DIF via an underlying DIF of a lower security clearance. The following open questions remain unanswered:
  - How to establish the clearance levels of AP/IPC Processes joining a DAF/DIF
  - Policies by which an IPC Process is allowed/denied joining a DIF based upon the clearance levels of the DIF and the joining IPC Process
  - How to establish the clearance levels of underlying DIFs
  - The process by which an AP/IPC Process decides to perform SDU protection based upon the security classification of the data being sent and the security clearance of the underling DIFs used to transmit the data
  - How to take into account possibly differing security clearances of DIFs on alternative paths/routing through the underlying network of DIFs
- Domain Based MLS – Data is communicated between AP/IPC Processes of potentially differing security clearances. The AP/IPC Process is ultimately responsible for the confidentiality of the data it sends, it is responsible for not

leaking data between classification levels. The data communicated within the DAF/DIF is explicitly labelled with the accurate security label by the sender of the data, such that the IPC Processes of the DIF, and all lower level DIFs, can make decisions based upon the security labels inline with security policy regarding access control, SDU protection and routing to ensure that no unauthorised leaking of data between classification levels is possible. Beyond those open questions remaining from MSL, the following questions arise:

- The wider implications of allowing a DAF/DIF to include AP/IPC Processes of differing security clearances
- How to establish, record, protect and maintain data pertaining to the security clearance levels of AP/IPC Processes within a DAF/DIF (possibly by adopting a "no read down, no write up" security model for maintenance of RIB resources)
- Mechanisms for deciding whether to release data (i.e., send an SDU) based upon the classification of the data and the security clearance of the proposed recipient IPC Process (probably acting as a requestor) and where to locate such mechanisms in the RINA protocols
- How to include security labels indicating the classification level of data within an SDU
- Mechanisms for protecting security labels from unauthorised modification/removal as the data/SDUs traverse the network and where to locate such mechanisms in the RINA protocols
- How to tailor SDU protection mechanisms based upon the security classification of the data being protected

4. Secure Channel and SDU Protection

4.1. Secure Channel

4.1.1. Aim of the Secure Channel

The RINA architecture includes a number of components that perform security mechanisms, e.g. the Authentication Module and SDU Protection Module. However, to to establish, use and maintain a secure channel [SecureChannel] in RINA, these components must be coordinated and work together. It is not clear in the current RINA specifications how this can be achieved, particularly as several of the required components are policy. The aim of this section is to investigate how a secure channel can be established, used and managed within RINA to protect data from eavesdropping and tampering. Primarily, using a secure channel protocol in RINA would:

- Protect the messages exchanged when an IPC Process is in the process of joining a DIF.
- Allow keys to be negotiated per session
- Enable application data to be protected prior to joining a DIF
- Provide mechanisms to manage keys, e.g. change session keys
- Enable the destination IPC Process to be authenticated, preventing man in the middle attacks.

In the following sections, we look at the Transport Later Security (TLS) protocol [RFC5246] as an example of an existing secure channel protocol to extract all of the functionality that is required. We map this functionality to the RINA specifications to identify how RINA components could be used and what additional functionality is needed. We then propose how RINA components can be used to establish, use and maintain a secure channel in RINA.

4.1.2. Use of Secure Channel

A secure channel is a means of transferring data that is resistant to overhearing and tampering [SecureChannel]. It protects the confidentiality and integrity of data sent between two application entities (AEs). In order to establish a secure channel, both AEs must agree on policies for protecting data, such as which encryption and integrity protection algorithms to use. The destination AE should be authenticated to prevent a man-in-the-middle attack. The source AE may also be authenticated depending on the agreed policies. The AEs need to agree on the cryptographic keys to use for the session.

The keys may be pre-configured or negotiated as part of the secure channel set-up. Once the secure channel has been established, encryption and/or integrity protection can be applied to data according to the agreed policies. The channel also needs to be maintained, e.g. rekeying, to change the session key (see Continuing Management of Secure Channel).

Within a RINA network all DIFs implement the same protocols (DTP, DTCP and CDAP) and have the same mechanisms (e.g. enrolment). This means that a single secure channel protocol can be repeated at all DIF layers for protecting e.g., security credentials and data messages. It can also alleviate some of the potential vulnerabilities, such as those described in Section 6. The SDU Protection mechanisms used by the secure channel are the last processing step applied to a packet before it is written to a flow through the underlying DIF; therefore it can be used by the IPC Processes to guarantee the integrity of all the data exchanged via the N-1 flow and the authentication of the AEs. It could also provide confidentiality, which although not a primary requirement could be used to hide details of the workings of the DIFs that may be useful for attackers.

A secure channel can be used by any DAF or DIF to protect data from being accessed or modified by an untrusted lower level DIF. In particular, a secure channel usable by management applications would be of significant use to protect management data, e.g., exchanged between DIF manager and Management Agents, the unauthorised modification of which could compromise the availability of the network. Key management is another function that may require a secure channel. Most cryptographic schemes become trivially breakable if the key is exposed, so a secure channel between the key manager and IPC process could be used to transfer a key. A secure channel could also be of use when an IPC process joins a DIF to protect the exchange of credentials and to secure objects exchanged during enrolment.

Transport Layer Security (TLS) provides a fairly complete example of setting up, using and managing a secure channel. It includes authentication of one or both AEs (with the use of digital certificates as a common option), generation of secret keys and negotiation of policies for protecting application data (including compression, encryption, etc). TLS is a protocol strictly layered in TCP/IP and can make assumptions about TCP that are not necessarily correct in RINA. We therefore want to establish what functionality we can take from TLS and use in RINA. In this section, we analyse TLS to extract the main features that are required to achieve a secure channel. We then discuss how a secure channel could be set up, used and managed in RINA, particularly when an IPC Process joins a DIF, and how it is linked to RINA components, e.g. authentication and SDU protection.

4.1.3. Transport Layer Security

TLS is designed to provide communications security over the Internet. The protocol creates a secure channel between client/server applications to allow them to communicate in a way that prevents eavesdropping, tampering, or message forgery. TLS was originally designed to sit on top of a reliable transport protocol, e.g. TCP. However, there is a variant of TLS, DTLS (Datagram Transport Layer Security) [RFC6347], which can be used with UDP-based protocols. TLS is application independent; it provides security to any two communicating applications that transmit data over a network via an application protocol.

TLS is composed of two layers. The record protocol is the lower layer and is used to encapsulate and protect higher-level protocols. It uses symmetric cryptography to provide confidentiality of and a keyed message authentication code (MAC) to provide integrity protection of the data. TLS has three sub-protocols that are layered on top of the record protocol: the handshake, change cipher spec and alert protocols. The handshake protocol is used to negotiate a session, e.g. an encryption algorithm and cryptographic keys. The change cipher spec protocol is used to signal transitions in the cryptographic parameters of a session. The alert protocol is used to notify the other party of an error condition.

Setting up a Channel using TLS

The handshake protocol is used to set up the secure channel. It consists of a sequence of messages sent between the client and server. It is used to negotiate the set of algorithms for authentication, confidentiality and integrity, generate shared secrets and negotiate other parameters for the session. It also allows peers to optionally authenticate each other.

To initiate the TLS Handshake Protocol, the client and server exchange hello messages to select the algorithms, exchange random values, and check for a resumed session. They then exchange the necessary cryptographic parameters to establish a premaster secret key. The client and server may authenticate themselves by exchanging certificates and cryptographic information. The premaster secret key and the exchanged random values are then used to generate a master secret key. These security parameters are provided to the record layer.

Once the security parameters are in place, the ChangeCipherSpec message is used to notify the other party to start using the newly negotiated security parameters. All subsequent messages are protected using the negotiated confidentiality and integrity algorithms and the derived secret keys.

A Finished message is sent by both parties immediately after the ChangeCipherSpec message. The message includes an integrity check of all the handshake messages and is protected using the new keys and algorithms. This allows both parties to verify that their peer has calculated the same security parameters and that the handshake occurred without tampering by an attacker.

Authentication Mechanisms Supported by TLS

TLS supports a number of authentication mechanisms. The mechanism used for a session is negotiated as part of the TLS handshake. The most commonly used authentication mechanism is public key certificates. The server sends its certificate to the client, which validates the certificate and the trust chain. The client uses the public key contained in the certificate to encrypt the premaster secret. The server then proves possession of the public key in the certificate by successfully decrypting the premaster secret. If client authentication is required, the client sends its public key certificate to the server. The client proves possession of the key by digitally signing a message using the corresponding private key.

TLS can be configured to support authentication using a pre-shared key [RFC4279], see also Section 5. This allows symmetric key encryption to be used, avoiding public key operations. It is suited to closed environment where the connections are mostly configured manually in advance, symmetric key more suited to constrained environments. TLS can also be configured to support Kerberos [RFC2712] or Secure Remote Password (SRP) [RFC5054] authentication.

Protecting Data using TLS

The TLS record protocol protects data according to the cipher suites negotiated during the handshake. When sending data the record protocol first fragments the data into manageable blocks. The fragmentation does not preserve client message boundaries; multiple messages of the same type may be concatenated into a single record, or a single message may be split across several records. The record protocol may compress the data, depending on the algorithms agreed in the handshake. A Message Authentication Code (MAC) is then applied and the data is encrypted before being transmitted. On receipt the data is first decrypted and the MAC is verified. The data is then decompressed, if compression was used, reassembled and delivered to the higher-level application.

TLS provides confidentiality using encryption. The encryption keys are generated from the master secret and the random numbers exchanged during the handshake. Two

encryption keys are derived: one to encrypt and decrypt the client's messages and one to encrypt and decrypt the server's messages. The encryption algorithm is negotiated during the handshake. TLS supports a range of ciphers; it can be used with block ciphers, stream ciphers or authenticated encryption with additional data (AEAD). A block cipher encrypts every block of plaintext to a block of ciphertext. It requires an Initialisation Vector (IV), which is chosen at random and must be unpredictable. How this IV is communicated is not defined in the TLS specification. One option is to send it in the same message as the premaster key. A stream cipher exclusive-ORs the plaintext with an identical amount of output generated from a cryptographically secure keyed pseudorandom number generator. For decryption to be successful, the sender and receiver must be synchronised. How this is done is not defined it the TLS specification. AEAD encryption encrypts and applies integrity protection to the plaintext simultaneously.

Integrity protection is provided using a keyed MAC algorithm, which is negotiated during the handshake. Two MAC keys are used: one for messages sent by the client and a second for messages sent by the server. The default is to calculate the MAC first and then encrypt both the message and the MAC. However, there is an extension to the protocol to enable the MAC to be calculated after the message has been encrypted [RFC7366]. The MAC of the message also includes a sequence number so that missing, extra, or repeated messages can be detected. Sequence numbers are 64 bits long, so there should be no need to wrap. If a sequence number would need to wrap, then the session should be renegotiated.

Continuing Management of Secure Channel

In addition to specifying how to create a TLS session using the handshake, the specification defines protocols to enable the continuing management of the secure channel. TLS session resumption allows multiple connections to be instantiated using the same session or enables the existing session to be duplicated with same security parameters. It uses an abbreviated TLS handshake, as the previously negotiated encryption and compression algorithms and master secret are used. The client and server exchange new random values, which are used with the previous the session's master secret to generate the session key.

TLS renegotiation allows the server and client to renegotiate security parameters in existing connection. For example, it can be used to change the master secret or to authenticate the client. The renegotiation can be initiated by either the client or the server. It requires a full TLS handshake and takes place over the existing TLS connection. There is an attack against the renegotiation protocol, so an extension

has been proposed [RFC5746] that requires the client and server to include and verify information about previous handshakes in any renegotiation handshakes. An extension to TLS, the Heartbeat extension [RFC6520], allows a peer to test and keep the connection alive without having to renegotiate the connection.

Although TLS allows peers to securely negotiate a secret key over an unsecured channel, it does not specify how to manage the additional keys used for authentication. Additional key management infrastructure may therefore be needed depending on the mechanism used. Authentication using X509 certificates requires a public key infrastructure, while using pre-shared keys may avoid the need for public keys, depending on the ciphersuite used.

4.1.4. Secure Channel Protocol in RINA

Here, we map the TLS functionality identified above to the RINA specifications and identify how existing RINA components can be used plus any additional functionality required in developing a secure channel protocol that is suitable for RINA networking environment.

Setting up a Channel in RINA

Setting up a secure channel requires the two AEs to negotiate the set of algorithms for authentication, confidentiality and integrity, generate shared secrets and negotiate other parameters for the session. It enables the AEs to authenticate each other. The AEs should also verify that they have both calculated the same security parameters and ensure that the set up of the secure channel occurred without tampering.

In RINA, the algorithms for authentication and protection of data, e.g. encryption and integrity protection, are determined by the DIF's policies. These policies and associated credentials and cryptographic keys can be exchanged at enrolment, when the joining IPC is initialised with the state it needs to participate in the DIF, and stored in the RIB. Once an IPC process has successfully joined the DIF, it therefore has the necessary polices and security parameters to establish secure communication with other IPC processes in the DIF.

The current RINA specifications do not define how a secure channel can be established before the IPC is fully a member of the DIF, i.e. when the IPC is in the process of joining the DIF. Therefore, it is not yet defined how to protect the messages exchanged between the IPC joining the DIF and the existing DIF member, which may include sensitive data such as authentication credentials and key material. In addition, prior to joining the DIF, the IPC process has no knowledge of the DIF's security policies, e.g. which

authentication mechanism and encryption algorithms the DIF supports. We therefore consider how the features of the TLS handshake protocol could be used or adapted to RINA to establish a secure channel when an IPC is in the process of joining a DIF.

In the RINA specifications, the first step of joining a DIF is for the joining IPC to request a flow to be allocated in the underlying DIF. Once a flow has been established, the application connection can be created using CACEP. The joining IPC process sends a connect request to e.g., an existing DIF member, and authenticates using the mechanism configured in the DIF policy. Once the IPC process has successfully authenticated, it is initialised with the data it needs to participate in the DIF, e.g. an address and policies, which is sent using CDAP. This initialisation data exchanged at enrolment could include sensitive management data, e.g. cryptographic keys, authentication credentials, which needs to be protected. Therefore the secure channel should be established, with the necessary SDU protection policies, cryptographic keys and security parameters in place, before this enrolment data is sent.

A communication service needs to be in place before a secure channel can be set up. Therefore, the minimum requirement is for a flow to be established to the destination IPC through the underlying DIF. Initially, the IPC process joining the DIF will not have been initialised with the necessary cryptographic keys to protect data within the DIF or with the SDU protection and delimiting policies of the DIF. This means that DTP will need to operate in the clear, i.e. with the IPC processes' SDU protection policies set to null, before the secure channel has been set up. Once the secure channel has been established, the SDU protection policies can be updated to use the negotiated algorithms and security parameters, so that subsequent communication is protected by the SDU Protection Module.

Once the DTP flow has been established, the secure channel can be set up. Current RINA specifications define an Authentication Module as part of CACEP. Establishing the secure channel requires both AEs to be authenticated. This means that the secure channel should be set up as part of CACEP. After sending the CACEP Connect request, the IPC process joining the DIF and the existing DIF member negotiate the authentication, SDU protection and SDU delimiting policies to be used for the connection. Both IPC processes are authenticated according to the agreed authentication policy. Note that the policy may only require one of the IPC processes to authenticate. They then agree session keys and other security parameters for the connection.

Once all of the algorithms and security parameters have been agreed, both IPC processes need to verify that they have both calculated the same security parameters,

e.g. encryption keys. They also need to check that the set up of the secure channel occurred without tampering by an attacker. In the TLS protocol, this is achieved through the Finished message, which includes an integrity check of all the messages sent while setting up the channel and is protected using the new keys and algorithms. A similar message would need to be sent when establishing a secure channel in RINA. This requires both IPCs to store all the messages sent during the set up, calculate a cryptographic hash of the messages and protect the message using the negotiated algorithms and keys. This functionality could be part of the Security Management component of the IPC process, or it could be part of the Authentication policy.

Once each IPC process has established that there was no tampering, it needs to send a notification to the other IPC process that the set up was successful and that subsequent messages will be protected using the negotiated policies and security parameters. This could be achieved using the Security Management component, which can send the notification and then then update the IPC process's SDU protection and SDU delimiting policies and the security parameters. It would also need to instruct the SDU protection module to apply protection to SDUs sent over the DTP flow using the agreed compression algorithms, cipher suites and the cryptographic keys for the session. Note that as the secure channel is negotiated per application connection, an IPC process may need to maintain separate policies, session keys and security parameters for each connection.

The secure channel is considered to be established once the SDU protection and the SDU delimiting modules begin enforcing the negotiated policies with the agreed security parameters. Therefore, by setting up the secure channel using the Authentication Module as part of CACEP, the data exchanged over CDAP at enrolment, e.g. addresses, configuration data, keys, is protected by the secure channel. However, any data sent prior to enrolment, e.g. the authentication credentials, is sent in the clear. This means that any authentication mechanisms that include plaintext credentials being transmitted, .e.g. password authentication, should not be used.

Authentication of IPC processes in RINA

In RINA an IPC process is authenticated when it joins a DIF. The authentication mechanism used is determined by the DIF's authentication policy. The current RINA specifications do not provide any authentication policies; they only provide a template for specifying a policy. Therefore there are a number of open issues. For example, it is not yet clear how the credentials for authentication will be obtained. The current RINA specifications do not provide a means of authenticating the destination IPC Process, i.e. the existing DIF member. It is also not yet clear how this authentication is persisted in

future communications, i.e. how the destination IPC Process can verify that the sender is still the same.

If public key certificates are to be used for authentication, they will be issued and managed by the security domain's key management function. The certificate and associated private key will be stored in the IPC Process' RIB. The authentication would be performed by the IPC Process's authentication module. See Section 2 for more detailed information.

Protecting Data in RINA

To send data, e.g. CDAP messages, in RINA, the IPC Process in the N-level DIF writes units of data called Service Data Units (SDUs) to the N-1 flow through the underlying DIF. The first processing step by the N-level IPC Process is to apply SDU delimiting, which may fragment or concatenate one or more SDUs according to the configured SDU delimiting policy. The resulting "user data fields" are then input to EFCP, the data transfer protocol (DTP), which generates a DTP packet, called a Protocol Data Unit (PDU), per user data field. DTP packets are composed of Protocol-Control-Information (PCI) and the user data field, The PCI includes the source and destination addresses, source and destination connection endpoint identifiers, a quality of service identifier, the length of the PDU and a sequence number. The DTP PDU is handled to the Relaying and Multiplexing Task (RMT), which determines the output N-1 port for the DTP PDU. Once the N-1 port for the PDU is known, and before writing the PDU to the N-1 flow, SDU protection can be applied to the PDU.

SDU protection can apply compression, encryption and integrity protection, as well as error detection and limiting the lifetime of a PDU. The algorithms and how protection is applied are defined in the SDU protection policy. For example, depending on the policy, protection may be applied to just the PCI, just to the user-data field, the PCI and user-data field separately (i.e. different mechanisms applied to the PCI and the user-data field) or the entire PDU.

There are many overlaps between the TLS record protocol and the IPC process's Data Transfer functions; both fragment packets and include the ability to compress, encrypt and apply integrity protection according to configured policies. Therefore it should be possible to achieve the functionality of the TLS record protocol with the existing functionality of the IPC Process. Policies for SDU Delimiting could be specified to perform the fragmentation of SDUs in blocks of the adequate size. Policies for SDU protection could also be specified to allow the IPC Process to provide the functionality of the TLS record protocol, i.e. to compress, apply a MAC and encrypt the user data field of the PDUs using the algorithms negotiated during the set up of the secure channel.

However, the SDU delimiting and SDU protection policies must be coordinated by the Security Management component, as they are strongly linked. For example, encrypting using a block cipher may require data blocks of a specified size, or the ciphertext expansion of a cryptographic algorithm (i.e. the increase in length of a message when it is encrypted) may result in blocks that exceed the maximum packet size specified in the DIF's policies.

There are, however, some open issues. As mentioned previously, TLS requires a reliable transport protocol that synchronises and reorders packets - mainly due to the use of chaining block ciphers. Therefore "TLS-style" policies applied to data over an N-1 flow assume that the N-1 flow provides reliable and in-order delivery of data. DTLS is an extension to TLS that includes additional mechanisms, e.g. sequence numbers, to enable it to cope with lost, duplicated, reordered, or even modified packets. Further work is required to investigate whether additional mechanisms are needed in RINA, e.g. those in DTLS, to allow the secure channel to operate over DTP. In addition, the current RINA specifications only provide a template for defining SDU delimiting and SDU protection policies; they do not yet define any policies. This means that further work is required to establish how the required functionality can specified in the policies and how these can be coordinated by the Security Management component.

Continuous security management in RINA

The current RINA specifications define a Security Management component in the IPC Process that is responsible for implementing a consistent security profile for the IPC Process, coordinating all the security-related functions and may also executing some of them. This component is therefore a natural fit for performing the continuous management functions needed to manage a secure channel. However, this component is only briefly mentioned in the RINA reference model briefly and to date there is no specification available. Further work is therefore needed to define the functionality the Security Management component needs to perform to manage a secure channel, e.g. to allow the session keys or cryptographic algorithms to be changed. Although an IPC Process can leave and rejoin a DIF, it is not yet clear whether the keys from the previous session would be reused. A DIF can be configured to support a set cryptographic algorithms concurrently, with the IPC Processes determining which to use for a particular application connection.

4.1.5. Summary and Conclusions

We investigated how to establish a secure channel in RINA. The advantages and limitations of mapping the secure channel concept to RINA are as follows:

- Advantages:
  - Allow session keys to be negotiated per N-1 flow
  - Enable application data to be protected prior to joining a DIF
  - Provide mechanisms to manage keys, e.g. change session keys
  - Enable the destination IPC to be authenticated, preventing man in the middle attacks
- Limitations:
  - May require PKI if authentication is based on X509 certificates
  - Does not provide anonymity

We looked at the TLS protocol as an example of an existing secure channel protocol. We extracted the functionality of TLS and mapped it to policies for the RINA specifications to determine whether a secure channel could be established, used and maintained in RINA. We found that the existing components of an IPC Process provide the necessary functions to enable a secure channel to be established and used. However, there are a number of open issues that require further investigation. As the secure channel involves multiple RINA components, further investigation is needed to establish how these can be coordinated by the Security Management component. During the set up phase, the Security Management component needs a means of ensuring that the set up has not been tampered with and of verifying that they are both using the same algorithms and keys. It also require a means of notifying the other IPC Process that subsequent communication will be protected using the new policies. Mechanisms are also required to manage the secure channel, allowing the cryptographic keys to be changed and policies to be renegotiated. Furthermore, policies for authentication, SDU delimiting and SDU protection have yet to be defined. This means that, although on paper is seems plausible that the TLS functionality can be implemented in the policies, it remains to be seen whether this is the case in practise.

4.2. SDU Protection

Current RINA specifications mandate that the SDU protection module provides as a minimum integrity functions, whether cryptographic or not, lifetime limiting and confidentiality. From a security perspective encryption is a fundamental mechanism that SDU protection can use to achieve integrity and privacy of the communication. In the following section SDU protection mechanisms are discussed and potential issues that need to be further investigated are highlighted.

4.2.1. Protection Mechanisms

The protective mechanisms associated with SDU protection consist of error detection, cryptographic identity, lifetime limiting, confidentiality provision and compression.

Error detection

A mechanism of error detection is provided by means of using error detection or error correction codes. It is a matter of policy and mainly DIF parameters what particular error detection scheme is used. Usually, network and telecommunication systems employ the following error detection methods: checksum, cyclic redundancy check or error detection codes.

Lifetime limiting

This mechanism should serve to limit the lifespan of a PDU in a network. This mechanism is substantial for the proper functioning of DTCP because it assumes that a maximum packet lifetime (MPL) is enforced.

Cryptographic integrity

Cryptographic integrity serves as a detection of tampering. It is based on adding a message authentication code (MAC) to the message. In theory, any MAC with suitable properties can be used. However, for practical reasons the cipher suite of an IPC Process would provide a limited set of MACs in its profile and negotiate the use of the particular algorithm with communicating party during authentication phase. The security policy may contain a method for selecting the most suitable MAC algorithm among the ones available on both AEs. To improve robustness of integrity protection, Keyed-Hashing for Message Authentication (HMAC) can be used instead of MAC algorithms [RFC2104]. The SDU protection uses the cryptographic integrity algorithm negotiated during the authentication phase, which is external to this module. The only condition is that both sides must agree on the HMAC algorithm and parameters used.

Confidentiality

Protecting the content of communication against eavesdropping is performed by applying an encryption algorithm. The selection of the particular encryption algorithm used is a matter of policy. This policy may take into consideration, for instance, QoS requirements or the capabilities of the underlying DIF (advertised via its QoS cubes). The selection of a crypto algorithm may also cause additional processing of the SDU. For example, block ciphers require that the data blocks to be encrypted have a size equal

to the length of the key used; thus if the SDU size does not match it needs to be delimited - therefore proper coordination with SDU delimiting policies must be ensured. If the underlying DIF is able to provide reliable data transfer then stream ciphers can be used.

Compression

Compression is a mechanism that helps to reduce the amount of data or remove blocks of invariant data before they are further manipulated by security mechanisms. For instance, TLS can support using a lossless data compression method that can be negotiated during the TLS Handshake and applied to the payload of a TLS Record before encryption is performed. Hollenbeck in [RFC3749] proposed to add the DEFLATE compression method to TLS and elaborates on the required properties of compression algorithms for using with TLS. He pointed out that it may be problematic to use some of the most efficient compression methods because they require state information to be maintained. Certain applications may enjoy advantages of data compression, for instance, when data format is as verbose as XML, while for other applications this may represent only little or no benefit. It may be interesting to consider lossy compression algorithms for flows that represent, e.g., multimedia data. Selection of a specific compression algorithm should be governed by the compression policy.

4.2.2. Operation

SDU Protection depends on a policy that is specific for each (N-1)-flow. SDU Protection creates a secure channel between two IPCPs, though it is not excluded that SDU Protection may apply the same policy to all (N-1) flows thus creating shared security for whole N-DIF.

To apply encryption, communicating parties need to know at least a single shared key. From the perspective of SDU protection, it is a question of how to manage a security policy that involves the use of cryptographic keys associated with a (N-1)-flow. If cryptographic keys are not distributed by some other method, the SDU encryption uses keys that come either from the application authentication phase or IPC Process enrolment phase. But before these parameters are known to involved IPCPs the communication channel with either no encryption or using some predefined key is used. When the authentication or the enrolment phase is finished the crypto keys are known and are used for protecting subsequent SDUs. The synchronisation of security parameters itself is external to SDU protection, but SDU Protection needs to know which crypto key to use when decrypting a received SDU. Note that DTLS solves this problem by using generation and sequence numbers [RFC6347].

4.3. The Selected Mechanisms for Design and Implementation

This section presents a plan, methodology and approach for simulating and testing the secure communication protocol using a simulation environment.

4.3.1. Objectives of Simulation

The identified security mechanism will be integrated into simulation models and will be simulated in isolation in the first round and then in complex security scenarios to understand how the secure channel protocol behaves within the RINA architecture. Also a preliminary performance evaluation, especially in conjunction with QoS requirements, will be analysed using the simulator.

The simulation model can either be used to solve specific problems or as a training tool. Both uses are supposed here. The simulation model will be used to answer questions raised during the design of security mechanisms that are within the scope of simulation approach. The simulation model will also be used for explaining concepts of the security mechanisms proposed for RINA.

The objectives of the simulation models are to:

- Define a set of architectural principles and concepts to capture structure of the Secure Channel and describe the principles underlying the operation of the Secure Channel.
- Get a clear picture of the behaviour of the Secure Channel concept applied to RINA. Firstly, it provides a means of verifying the feasibility of the proposed approach. Secondly, through simulation it should be possible to understand the differences in flow processing in the RINA environment both with and without the Secure Channel enabled.
- Evaluate the performance of the Secure Channel for different applied mechanisms and policies. Experiments will be provided for different classes of application communication to provide information on possible combinations of QoS and Security parameters.
- Adjust security mechanism parameters with respect to RINA architecture features and limitations. Simulation may reveal inefficiencies in, e.g. number of messages, timing, etc.

The output from the simulation experiments will provide a source of information for detailed specification of the Secure Channel Protocol and guidelines for

implementation. Also, simulation models can be a suitable foundation for security analysis of the proposed mechanisms using specialised tools, e.g. using AVISPA tool.

4.3.2. Methodology

Our approach will follow the methodology as specified by Ulgen et al.[Ulgen1994], which includes the phases:

- Define the problem
- Design the study
- Design the conceptual model
- Formulate inputs, assumptions, and process definition
- Build, verify, and validate the simulation model
- Experiment with the model

The current state of the RINA simulator provides models that implement AE to AE communication. The missing parts of the models that would be required for integrating security are mainly the DTCP model and enrolment mechanism. Also, QoS cube selection has limited support and this would require development before we can experiment with security and performance parameters. Thus the current effort is to add critical components to enable adding security mechanisms into the RINA simulation model. It is expected that this will be finished at M10. Based on the outcomes of this document, the goals of the simulation study will be specified in detail.

4.4. Summary

Here we investigated how to achieve a secure channel in RINA, particularly when an IPC process joins a DIF. We have also looked at SDU protection in the RINA specifications. We then specified a methodology plan for implementing and testing the set up, use and management of a secure channel in RINA.

5. Key Management Function

A trusted entity is needed to generate, maintain and distribute keys to relevant processes. Those keys are used to encrypt/decrypt data and authenticate requests. This entity is usually called the key server and is of course a security sensitive item. Therefore there is some interdependency between key management and security mechanisms (e.g. encryption). A key agent is normally embedded in applications and checks the credentials of incoming requests by validating those keys. Since in RINA DIFs are independent from each other, so is the key agent. A part of its task is to reject keys that are syntactically valid but have been flagged to be revoked. To do so a black-list of keys need to be consulted. This black-list may be maintained centrally in the key server and pushed out to the key agents to update their information or remain in the key server which gets consulted for verification purposes.

A DIF is a homogeneous DAF in which all its members (IPCPs) perform the same task (providing and managing IPC services). Within a DIF, Key management can be one of the functions that the IPC Processes perform to manage IPC's operation. A procedure whereby untrusted keys are excluded, e.g. by a black list (or optionally by a white list) shall also be present. This information is available to the key server and its clients to protect themselves from forged requests.

5.1. Key Management Architectures

Broadly, there are two options for the Key Server. The first option, shown in Figure 26, "Distributed Key Management Architecture", is to have logically one server per DIF, generating, maintaining and distributing keys to IPC Processes within the DIF for their operations.

Please note the following regarding Figure 26, "Distributed Key Management Architecture":

- The Domain-Level Key Manager (KM) must be a trusted entity identified during network initialisation.
- A management entity or the first IPCP which forms the DIF is responsible for talking to the Domain-Level KM, which should have a well-known address to be already given to the management entity/the IPCP.
- Domain-Level KM is responsible for creating/destroying DIF-Level KMs. It creates an IPCP as DIF-Level KM to be part of the DIF.

- The management entity/first IPCP may have an association with the Domain-Level KM in destroying it whenever the DIF is no longer needed.
- The address of the DIF-Level KM may be given to any new IPCP joining the DIF at the end of enrolment process after the secure channel has been established.

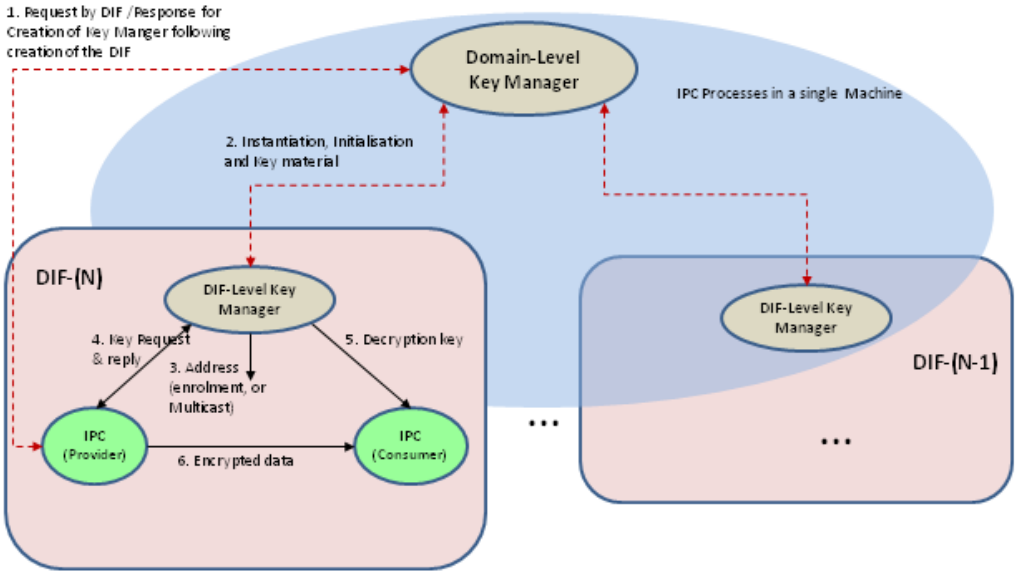

Figure 26. Distributed Key Management Architecture

An alternative key management architecture is shown in Figure 27, "Centralised Key Management Architecture". Here all IPC Process instances of a DIF share the same list within their key server instance, so that agents and servers are merged.

By enrolling to a DIF/DAF, application credentials shall be available a priori to the application instance to which the Application Entity belongs. The credentials should have been provided via a trusted path originating from a shared root of trust. In the Internet, the credentials are created by one of a handful of Certification Authorities and their delegates, stored and protected by the OS and browser, and the trusted path is via the browser and OS vendors. RINA can choose to use that model by bundling RINA Certificate Authority (CA) certs with the RINA software distribution being installed in a network. Once acquired, it is an OS's responsibility to ensure that those credentials are only available to binaries or process-IDs that are permitted to use them. Since the initial key needs to be embedded into the IPC Process before joining a DIF, the key needs to be provided by other means such as a Command Line Interface (CLI), CD, USB, SIM-card or by a trusted connection to the CA via a computer network such as the Internet. The initial implementation will utilize CLI to embed the initial key but is open for alternative ways of key dissemination.

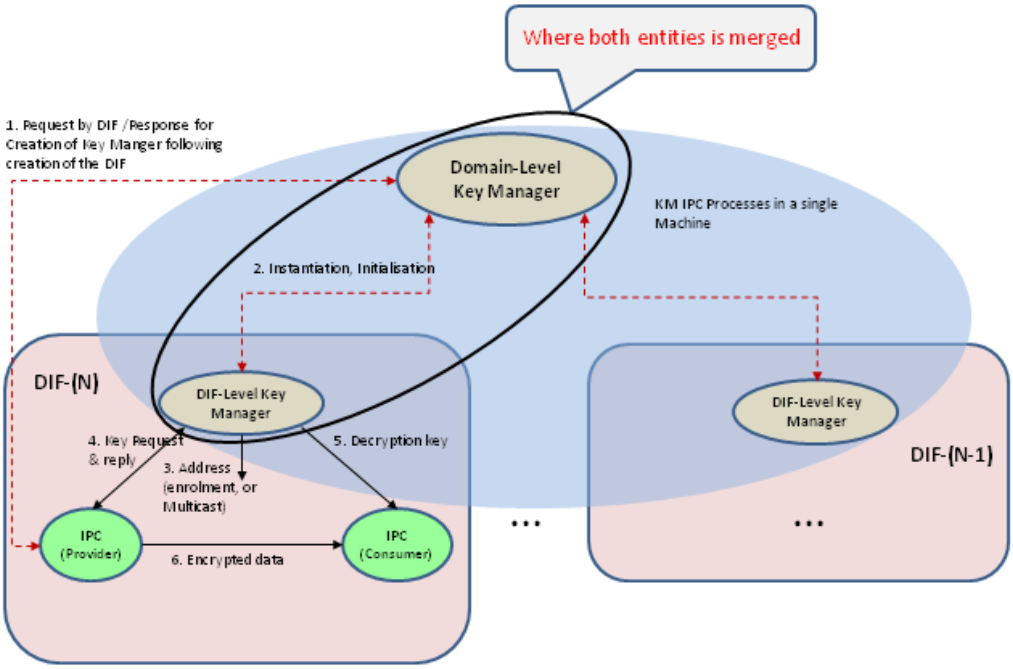

Figure 27. Centralised Key Management Architecture

During enrollment, the joining IPC Process holds an embedded key that would allow it to identify itself to the DIF. In case of a distributed key server, IPC processes are talking to a single IPC Process instance of the joining DIF during the enrollment procedure, which can immediately validate a key with its local key-server instance. It does not need to relay the key validation to a central entity inside the DIF. Consequently in terms of performance, the distributed key server is more effective.

On the other hand a centralized key server is easier to administer, but requires a means to announce the address of the Key Server whithin the DIF such that IPC Processes can reference the key server when required for enrolling a new IPC Process. A centralized key server also needs to manage redundancy to avoid a single point of failure within a DIF. Performance wise, a potentially longer distance communication between the joining IPC Process and the need for an existing IPC Process to relay key information to the key server during enrolment would slow the procedure down. However this might still be acceptable in initial configurations since enrolment is a non-frequent process.

Within the IPC Process, a key needs to be associated to a property of the IPC Process itself such as name, address or process-id. The name of the IPC Process remains unchanged throughout the IPC Process lifetime, whereas addresses and IDs are temporary synonyms assigned to IPC Processes for use within a DIF. An IPC Process can therefore have multiple addresses or process-IDs during its lifetime as a DIF member. For this reason keys could be associated with the IPC Process name rather

than its address to enable a static key binding to the IPC Process. This method also requires process names to be unique within a DIF and potentially duplicated names to be identified before accepting a new IPC Process enrolling in a DIF. Another viable way is to associate key and IPC Process dynamically every time the binding changes e.g. the process ID after a re-start. The preferred choice for PRISTINE is to use the application Name as an association as it contains the information needed to uniquely identify the instance: process name, process instance, entity name, entity instance

5.2. Conclusion

We specified two key management architectures in this chapter. It needs to be studied whether the key server is part of the DIF Management System or distributed within the IPC processes sharing key information. Both methods are possible. While a key server associated with the DIF-NMS can be more efficient in a single management domain, it could become a bottleneck when several domains need to collaborate and Key servers need to align their information. For such cases a distributed server architecture has advantages since the means to communicate between servers is embedded in the code.

6. Threat Identification, Monitoring and Countermeasures

The main aim of a RINA network is to provide an inter-process communication service to applications in a more efficient way. To this end the main attacks to the network will focus on disrupting this communication service. There are several types of attacks on network communications: eavesdropping, disrupting or blocking communication, injecting fabricated packets, modifying the storage, tables or packets. In this section, we perform an initial security risk assessment to identify runtime threats to a RINA network and define measures to mitigate them.

6.1. Risk Assessment Methodology

This evaluation of the threats to a RINA network will loosely follow the SecRAM methodology, which is the ISO/IEC 27005 (Information security risk management)-based Risk Assessment methodology developed by the SESAR programme [SecRAM]. SecRAM was intended for another context and as such it is too heavy weight to apply it in full to RINA. We therefore tailor the SecRAM methodology to apply it to RINA.

The steps of the security risk assessment are to:

- Establish an accurate scope: describe the system that is the target of the study and the dependencies on other systems and infrastructure.
- Identify the asset impacts: identify the assets and evaluate the harm resulting from each asset being targeted by an attack.
- Identify the threats or threat scenarios: identify possible (or credible) threat sources and the related threat scenarios to highlight all routes through the system that an attacker may use to access an asset.
- Evaluate the likelihood of each threat scenario, i.e. the chances that the threat occurs and that the threat scenario sequence is completed successfully;
- Assess the security risk: evaluate the risk level associated to each combination of threat and threat scenario based on their likelihood and asset impact;
- Define a set of security controls and requirements to reduce the risk to an acceptable level (i.e. within the risk appetite).

6.2. Context and Scope

Design time identification of vulnerabilities in the RINA protocols and APIs and mitigation of these are out of the scope of PRISTINE. We are focussing on intentional

attacks on the network and its assets, rather than faults or accidental damage, e.g. unintentional misconfiguration of policies. Therefore we only consider run-time attacks and countermeasures here.

We consider a simple RINA network with a DAF, an N-level DIF and an (N-1)-level DIF and the attacks originating from each of the three layers (see Figure 28, "Example RINA network"). An AP in the DAF relies on the N-level DIF to transport data. The AP cannot see the internals of the underlying N-level DIF and has no visibility of the N-1-level DIF; a malicious AP can only attack the N-level DIF through the IPC API. However, it is able to establish a CDAP connection to another AP in the DAF using CACEP and then send application data using CDAP. The N-level DIF can only see the IPC API of the N-1 DIF, so this is the only means of attacking its underlying DIF. However, the IPC Process can access other IPC Processes in the DIF. The N-1 DIF transports SDUs between IPC Processes in the N-level DIF, but cannot access the internals of the N-level DIF.

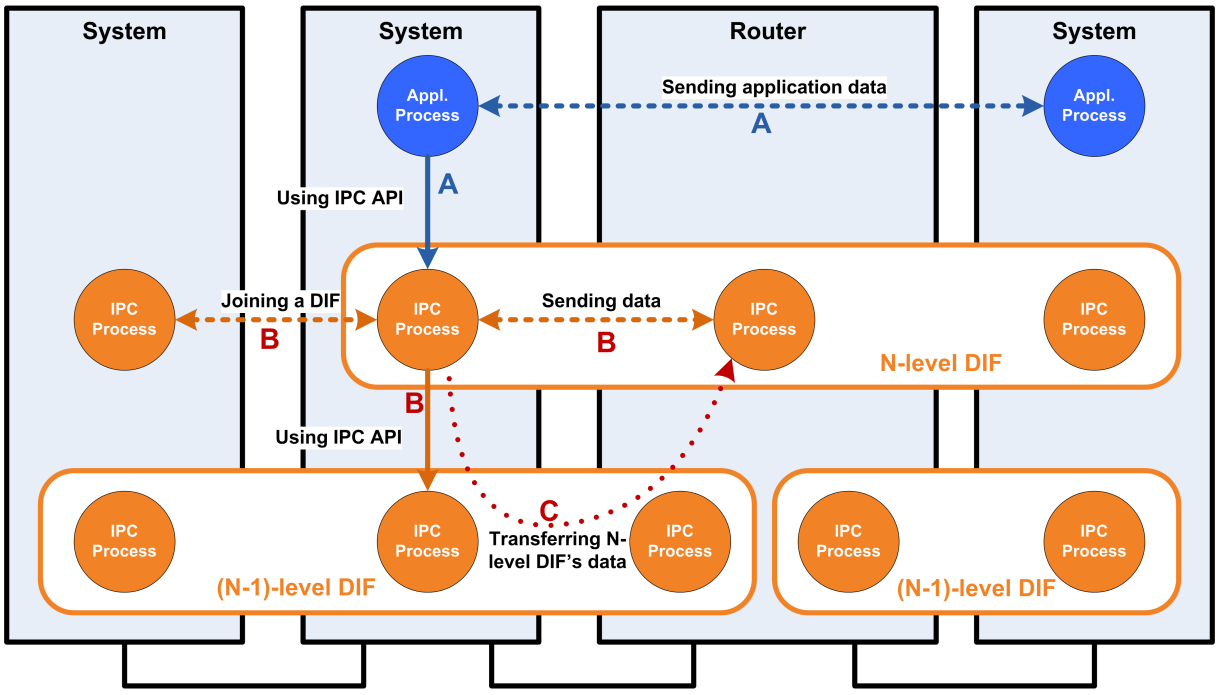

Figure 28. Example RINA network

Also in the scope of the security risk assessment are DIF management functions, such as the Key Manager (KM) and Access Control Manager (ACM), as well as the Management Agent (MA) on each node. However, as these functions are still in the process of being defined, we only consider an initial set of threats to these functions. A more detailed threat assessment of the KM, ACM and MA will be covered in next version of this document.

6.3. Asset Identification

There are two types of assets. Primary assets are the intangible targets of an attack, which are valuable to a RINA network. There are two main types of primary assets: information and services. A successful attack would result in damage to the primary assets and have an impact on the network.

The main primary assets of a RINA network are shown in the table below.

Table 1. RINA Primary Assets

Primary Asset	Type	Description
User data	Information	Any data which is stored on or transmitted to or from an AP or IPC Process, or that can be inferred from such data, e.g. documents, media files
Management data	Information	A subset of data that concerns the operation of the computing system or network, e.g. configurations, addresses, cryptographic keys, policy data
Computing resources	Service	This refers to the computer system itself, e.g. memory storage, processes
Communication Service	Service	The service that transfers data from one system to another

Supporting assets are tangible entities that enable and support the existence of primary assets. Entities involved in storing, processing and/or transmitting primary assets are classified as supporting assets. They may have vulnerabilities that can be exploited by threats targeting the primary assets. Within RINA, the supporting assets are mostly the entities in an IPC Process as shown in Figure 2, “RINA Architecture” as well as in DAF/DIF management functions.

The following table lists the supporting assets that may be targeted by a threat scenario and their related primary assets.

Table 2. Supporting Assets

Supporting asset	Description	Related Primary Asset
EFCP	Protocol that transfers data and controls flow and retransmission	User data, Management data, communication service
SDU Protection	Entity that applies protection to application data	User data, Management data
SDU Delimiting	Entity that performs SDU Fragmentation/ concatenation	User data, Management data
RIB	Logical representation of all information known by the IPC, e.g. configurations, key material, policies, logs	Management Data, Computing resources
RIB Daemon	Broker for RIB that operates on objects stored in the RIB	Management Data
IPC API	Entity that offers a communication service to application processes in the layer above	User data, Management data, communication service
CACEP	Entity that handles setting up application connection	Management data, communication service
Authentic-ation	Entity that handles authentication of the IPC process (plugin of CACEP)	Management data
CDAP	Application Protocol	Management data, user data, communication service

Supporting asset	Description	Related Primary Asset
Enrolment	Entity that coordinates an IPC Process to join a DIF; For joining a DIF or helping other IPC Processes to join	Management data
Flow Allocation	Entity that processes flow allocation and deallocation requests. It allocates a port id, locates applications and negotiates data transfer parameters	Communication service
Resource Allocation	Entity that decides how to allocate IPC resources and monitors their use.	Computing resources
RMT	Real-time scheduling of sending PDUs to (N-1) DIF	Communication service
DAF/DIF related assets		
Key Manager (KM)	Entity responsible for distributing keys in a DAF/ DIF	User data, Management data
Access Control Manager (ACM)	Entity responsible for handling access rights in a DAF/DIF	User data, Management data
Others		
Management Agent (MA)	An agent that has access to the management task of all IPC Processes within a network processing system.	Management data
Manager	The AP that oversees the management of a set of IPC Processes belonging to one or more DIFs, via the Management Agent	Management data

6.4. Threat Scenarios

The PRISTINE project is focussed on intentional threats to a RINA network. Therefore, we do not analyse the complete spectrum of threats (e.g. accidental, natural criminal, terrorist), here. Only the most relevant threats, according to the scope described in section 6.2, have been selected and applied to the secondary assets.

The following table lists potential attacks on the network that originate from an AP in the DAF (labelled 'A' in Figure 28, "Example RINA network"). We consider attacks where either the sending or receiving AP can be malicious.

Table 3. Description of type *A* attacks

Threat ID	Supporting asset	Description
A1	IPC API	Attacker masquerades as another application process when requesting a flow. It calls the Allocate_Request API with source application id of another application, which is used to make access decision whether to grant request. Results in a flow being created from the attacker to an application process.
A2	IPC API	Attacker repeatedly calls Allocate_Request API in attempt to consume resources and perform denial of service attack
A3	IPC API	Attacker masquerades as another application process and calls the Send API to inject PDUs
A4	IPC API	Attacker repeatedly sends messages to another AP to overwhelm it and consume resources
A5	IPC API	Attacker masquerades as another application process and calls the Deallocate API.
A6	CACEP	Attacker spoofs a RELEASE message to disrupt connection to another AP
A7	CACEP	Attacker floods CONNECT messages to overwhelm destination and consume resources
A8	CDAP	Malicious destination AP learns AP name and credentials of legitimate AP and uses these to impersonate the AP and join the DAF.

The following table lists possible attacks originating within the N-level DIF or from an IPC Process joining the N-level DIF (labelled 'B' in Figure 28, "Example RINA network").

Table 4. Description of type *B* attacks

Threat ID	Supporting asset	Description
B1	SDU Protection	Attacker compromises SDU Protection so that it malfunctions
B2	SDU Delimiting	Attacker compromises SDU Delimiting so that it malfunctions
B3	RIB	Attacker accesses objects in the RIB of another IPC, .e.g. key material
B4	RIB	Attacker writes objects to the distributed RIB, e.g. changes the DIF's SDU protection policies so that SDUs are not encrypted
B5	RIB	Malicious IPC Process updates forwarding tables in RIB to route traffic through malicious nodes
B6	RIB	Attacker deletes objects from the distributed RIB, e.g. deletes logs
B7	IPC API	Attacker masquerades as another IPC process when requesting a flow. It calls the Allocate_Request API with source application id of another IPC process, which is used to make access decision whether to grant request. Results in flow being established between malicious IPC process and the DIF it wants to join.
B8	IPC API	Attacker repeatedly calls Allocate_Request API in attempt to consume resources and perform denial of service attack
B9	IPC API	Attacker masquerades as another IPC process and calls the Deallocate API, causing the victim IPC process to lose its connection to the DIF.
B10	IPC API	Attacker masquerades as another application process and calls the Send API, injecting DTP messages
B11	IPC API	Attacker repeatedly sends messages to another IPC to overwhelm it and consume resources

B12	CACEP	Attacker floods CONNECT messages to overwhelm destination IPC process and consume resources
B13	CACEP	Attacker spoofs a RELEASE message to disrupt connection between the targeted IPC and the DIF
B14	Authentic-ation	Attacker bypasses the authentication check and joins the DIF
B15	Authentic-ation	Attacker repeatedly sends false credentials to consume resources
B16	CDAP	Attacker performs a man-in-the-middle attack, intercepting CDAP packets and forwarding them on to the destination IPC.
B17	Enrolment	Attacker masquerades as another IPCP when joining DIF
B18	Enrolment	Attacker repeatedly joins and leaves a DIF, consuming resources
B19	Flow Allocation	Attacker compromises Flow Allocation so that it malfunctions
B20	Resource Allocation	Attacker compromises Resource Allocation so that it malfunctions
B21	RMT	Attacker compromises RMT so that it malfunctions in relaying and scheduling function
B22	MA	Malicious IPC Process compromises MA to gain access to other DIFs within a processing system
B23	ACM	Malicious IPC Process compromises ACM so that it malfunctions
B24	ACM	Attacker accesses objects in the ACM store to change users' profiles
B25	KM	Malicious IPC Process compromises KM so that it malfunctions

The following table lists potential attacks on the network that originate from the N-1-level DIF (labelled 'C' in Figure 28, "Example RINA network"). We do not consider attacks within the N-1-level DIF, as we consider these to be the same as those described above.

Table 5. Description of type *C* attacks

Threat ID	Supporting asset	Description
C1	EFCP	Malicious IPC Process fabricates PDUs from DIF above
C2	EFCP	Malicious IPC Process modifies PDUs from DIF above
C3	EFCP	Malicious IPC Process eavesdrops PDUs from DIF above
C4	EFCP	Malicious IPC Process replays PDUs from DIF above
C5	EFCP	Malicious IPC Process does not forward PDUs from DIF above
C6	IPC API	Malicious IPC Process eavesdrops AP or IPC Process name and credentials and uses them to enrol in a DIF
C7	CDAP	Malicious IPC Process fabricates CDAP messages, e.g. containing routing updates and sends these to the DIF above
C8	CDAP	Malicious IPC Process modifies CDAP messages containing routing updates and sends these to the DIF above
C9	CDAP	Malicious IPC Process eavesdrops CDAP messages e.g. containing key material.

6.5. Security Risk Assessment

For each threat, the impact on the confidentiality, integrity and availability of the network is assessed according to the following scale:

1. No impact / NA
2. Minor – impact is limited to the IPC or AP, but it is still able to function
3. Severe – performance of the AP or IPC is compromised
4. Critical – performance of the DIF or DAF is compromised
5. Catastrophic – RINA network or multiple DIFs are compromised

The overall impact is then calculated as the highest of the three impact values. We then assess the likelihood that the threat scenario is completed successfully according to the following scale:

1. Practically impossible (one in a million)
2. Conceivable but very unlikely
3. Only somewhat possible
4. Quite possible
5. Might well be expected

The following table shows the assessed impact and likelihood of each threat.

Table 6. Impact and likelihood of different threats

Threat ID	Confident-iality	Integrity	Avail-ability	Overall Impact	Likelihood
A1	3	1	1	3	3
A2	1	1	3	3	3
A3	2	2	1	2	3
A4	1	1	4	4	3
A5	2	1	3	3	3
A6	2	1	3	3	3
A7	1	1	3	3	3
A8	4	4	1	4	4
B1	2	2	2	2	2
B2	2	2	2	2	2
B3	3	3	1	3	2
B4	4	4	1	4	4
B5	4	4	1	4	4
B6	1	4	4	4	4
B7	3	1	1	3	3
B8	1	1	3	3	3
B9	2	1	3	3	3
B10	2	2	1	2	3

Threat ID	Confident-iality	Integrity	Avail-ability	Overall Impact	Likelihood
B11	1	1	4	4	3
B12	1	1	3	3	3
B13	2	1	3	3	3
B14	4	4	1	4	2
B15	1	1	3	3	4
B16	3	3	1	3	3
B17	4	4	1	4	4
B18	1	1	4	4	4
B19	2	2	3	3	2
B20	2	2	3	3	2
B21	2	2	4	4	2
B22	5	5	1	5	3
B23	4	4	1	4	2
B24	4	4	1	4	3
B25	4	4	1	4	2
C1	1	5	1	5	4
C2	1	5	1	5	4
C3	5	1	1	5	4
C4	1	5	1	5	4
C5	1	1	5	5	4
C6	4	4	1	4	4
C7	1	4	1	4	4
C8	1	4	1	4	4
C9	4	1	1	4	4

Once the likelihood and impact of each threat has been assessed, the level of risk can be calculated using the following table.

	Impact				
Likelihood	**1**	**2**	**3**	**4**	**5**
5	Low	High	High	High	High
4	Low	Medium	High	High	High
3	Low	Low	Medium	High	High
2	Low	Low	Low	Medium	High
1	Low	Low	Low	Medium	Medium

Figure 29. Risk level calculation definition table

The following table show the risk level for each of the identified threats.

Table 7. Risk level of the threat scenarios

Threat ID	Likelihood	Impact	Risk Level
A1	3	3	Medium
A2	3	3	Medium
A3	3	2	Low
A4	3	4	High
A5	3	3	Medium
A6	3	3	Medium
A7	3	3	Medium
A8	4	4	High
B1	2	2	Low
B2	2	2	Low
B3	3	3	Medium
B4	4	4	High
B5	4	4	High
B6	4	4	High
B7	3	3	Medium
B8	3	3	Medium
B9	3	3	Medium
B10	3	2	Low
B11	3	4	High

B12	3	3	Medium
B13	3	3	Medium
B14	2	4	Medium
B15	4	3	High
B16	3	3	Medium
B17	4	4	High
B18	4	4	High
B19	2	3	Low
B20	2	3	Low
B21	4	2	Medium
B22	5	3	High
B23	4	2	Medium
B24	4	3	High
B25	4	2	Medium
C1	4	5	High
C2	4	5	High
C3	4	5	High
C4	4	5	High
C5	4	5	High
C6	4	4	High
C7	4	4	High
C8	4	4	High
C9	4	4	High

6.6. Security Controls

Threats that have a risk level of Low can be accepted and do not need further security controls. We therefore only consider additional security controls for threats with a risk level of Medium or High. The following table describes the security controls for threat scenarios with a risk level of Medium or High.

Table 8. Security control for medium and high risk threat scenarios

Threat ID	Risk Level	Security Controls
A1	Medium	Authenticate users of the IPC API

A2	Medium	Authenticate users of the IPC API. Monitor and log use of the IPC API.
A4	High	Authenticate users of the IPC API. Monitor and log use of the IPC API.
A5	Medium	Authenticate users of the IPC API.
A6	Medium	Authenticate APs when establishing a CDAP connection and include proof of authentication with CDAP messages, .i.e. persist the authentication for the connection.
A7	Medium	Authenticate APs when establishing a CDAP connection and include proof of authentication with CDAP messages, .i.e. persist the authentication for the connection. Monitor CDAP connections
A8	High	Authenticate the destination AP before sending credentials.
B3	Medium	Control access to the RIB. Monitor and record access to the RIB.
B4	High	Control access to the RIB. Only the Management Agent should be able to change the DIF's policies. Monitor and record access to the RIB.
B5	High	Monitor routing updates. Authenticate routing update messages.
B6	High	Control access to the RIB. Log files must not be deleted. Monitor and record access to the RIB.
B7	Medium	Authenticate users of the IPC API
B8	Medium	Authenticate users of the IPC API. Monitor and log use of the IPC API.
B9	Medium	Authenticate users of the IPC API.

B11	High	Authenticate users of the IPC API. Monitor and log use of the IPC API.
B12	Medium	Authenticate APs when establishing a CDAP connection and include proof of authentication with CDAP messages, .i.e. persist the authentication for the connection. Monitor CDAP connections
B13	Medium	Authenticate IPC Processes when establishing a CDAP connection and include proof of authentication with CDAP messages, .i.e. persist the authentication for the connection.
B14	Medium	Authentication must be non-bypassable and a strong authentication mechanism should be used.
B15	High	Monitor the number of authentication failures
B16	Medium	Authenticate the destination IPC Processes when establishing a CDAP connection.
B17	High	Use a strong authentication mechanism.
B18	High	Monitor IPC Processes enrolling in the DIF
B21	Medium	Monitor RMT functions to detect abnormal behaviour
B22	High	Monitor MA to detect abnormal behaviour
B23	Medium	Monitor ACM to detect abnormal behaviour
B24	High	Control access to the ACM data store Monitor and log access to the ACM data store
B25	Medium	Monitor KM functions to detect abnormal behaviour
C1	High	Protect the integrity of PDUs when sending over an untrusted N-1 DIF
C2	High	Protect the integrity of PDUs when sending over an untrusted N-1 DIF
C3	High	Protect the confidentiality of PDUs containing sensitive data when sending over an untrusted N-1 DIF

C4	High	Use replay protection on PDUs when sending over an untrusted N-1 DIF
C5	High	Monitor the DIF to detect IPC Processes not forwarding PDUs.
C6	High	Protect the confidentiality of PDUs containing authentication data when sending over an untrusted N-1 DIF
C7	High	Protect the integrity of PDUs when sending over an untrusted N-1 DIF Verify the authenticity of CDAP messages
C8	High	Protect the integrity of PDUs when sending over an untrusted N-1 DIF Verify the authenticity of CDAP messages
C9	High	Protect the confidentiality of PDUs containing sensitive data when sending over an untrusted N-1 DIF

From the table above, it can be seen that the majority of threats can be mitigated using existing mechanisms that are provided by IPC Processes and that have been considered in the earlier sections of this document:

- Authenticate both the source and destination APs or IPCs when establishing a CDAP connection, as described in Section 2.2.2
- Authenticate users of the IPC API. Note that this may be implemented in the operating system of the node, as described in Section 2.2.1
- Controlling access to the RIB, as described in Section 3
- Using SDU protection to protect the confidentiality and integrity of PDUs, as described in Section 4.

The remaining threats can be reduced by performing monitoring within a DIF to identify IPC Processes that are not behaving as expected. These threats can be generalised into the categories described in the table below.

Table 9. Security control for high or medium risk threat scenarios

Threat ID	Description	Related Threats

T1	An IPC Process provides false information to other IPCPs	B5
T2	An IPC Process deliberately overwhelming other DIF members or the underlying DIF with messages	A2, A4, A7, B8, B11, B12
T3	An IPC Process not forwarding messages	C5
T4	An IPC Process repeatedly joining and leaving a DIF	B18
T5	An IPC Process repeatedly causing errors when attempting to join a DIF	B15
T6	Compromising the data stored in an IPC Process or DIF	B3, B4, B6, B24
T7	Compromise an IPC Process or DIF function so that it malfunctions	B21, B22, B23, B25

Any further analysis will be reported in the 2nd version of this document. The remainder of this section discusses how to perform monitoring within RINA. It will consider where data for security monitoring can be gathered within RINA, how monitoring is performed and what metrics and tools should be used.

6.7. Monitoring and Counter Measures

There are four distinct phases to combat security threats and implement smart monitoring:

- **Monitor:** the situations are observed for gathering security information from the environment. A range of techniques and a variety of applications are used to monitor and collect information for detecting and assessing vulnerabilities and attacks. This may include probes/agents, vulnerability assessment tools, intrusion detection, etc. A number of metrics must be defined for monitoring purposes. Some of these metrics are listed below:
  - number of events in IPCs joining a DIF
  - number of events in IPCs leaving a DIF
  - latency of the IPC in responding to requests (processing, communications)
  - number of IPC calls to access objects
  - Volume of data transferred by an IPC

- number of DIF authentication events
- number of DIF authentication refusal
- and so on.

- **Analyse and Detect:** to develop an analysis and understanding of the situation based on experience, context and the information gathered during the observe phase. This phase can extract and combine events (using data aggregation, mining, normalisation, classification, correlation, etc. The events are analysed to detect abnormal behaviours. It is important to get this part correct as the later phases are based on the understanding developed in this phase.
- **Plan:** Once a vulnerability or an attack has been detected, a set of potential responses to the perceived situation is produced to mitigate it. These responses should be produced automatically using a decision support system (or by a human operator).
- **Act:** To implement and apply the chosen actions to the environment; the effects of the response are observed and the cycle continues. Self management ability with minimal human intervention is needed to automatically perform actions based on high-level mitigation/security policies.

6.8. Summary

Here we performed an initial security risk assessment to identify and prioritise runtime threats to a RINA network. We proposed additional security controls using internal RINA components for threat scenarios with a risk level of Medium or High and identified threats that require monitoring.

In order to realise the security state of RINA's network system, monitoring should be carried out for observing and gathering data from different indicators and possibly in a coordinated way, processing events, identifying the most feared vulnerabilities, adversary activities, and possible damages, and thus discovery of the needs for any adaptation of IPC/DIF components. With regard to the above, work is initially required to be carried out for identifying existing or developing viable methods and solutions for the stated phases. Further work will be reported in the next version of this deliverable.

7. Resiliency

7.1. State of the Art and Relevance to RINA

7.1.1. Failure detection in packet switched networks

This section provides an overview of existing failure detection protocols for packet-switched networks.

Bidirectional Forwarding Detection

The goal of Bidirectional Forwarding Detection (BFD) is to provide low-overhead, short-duration detection of failures in the path between adjacent forwarding engines, including the interfaces, data link(s), and, to the extent possible, the forwarding engines themselves. BFD is designed to work over any media, at any protocol layer, with a wide range of detection times and overhead [RFC5880].

BFD is a simple Hello protocol running between adjacent systems. A pair of systems transmit BFD packets periodically over each path between the two systems and when a number of consecutive BFD packets are not received, some component in that particular bidirectional path to the neighbouring system is assumed to have failed. A path is only declared to be operational when two-way communication has been established between systems, though this does not preclude the use of unidirectional links. A separate BFD session is created for each communications path and data protocol in use between two systems.

BFD has two main modes of operation. The primary mode is known as Asynchronous mode, where the systems periodically send BFD Control packets to one another, and if a number of those packets in a row are not received by the other system, the session is declared to be down. The second mode is known as Demand mode, where an independent way of connectivity verification is present and BFD control packets are only used to verify connectivity explicitly. Orthogonal is the Echo function, where the BFD control packets are returned back on the incoming link, after processing through the forwarding engine.

Ethernet Connection and Fault Management

Ethernet OAM Link Fault Management (IEEE 802.3ah)

The IEEE 802.3ah standard [IEEE802.3ah] defines an Operations, Administration, and Maintenance (OAM) sublayer (in between Media Access Control and LLC layers),

which provides mechanisms useful for monitoring link operation such as remote fault indication and remote loopback control (a feature where the upstream node is requested to immediately reflect all traffic back on the link) in Ethernet networks. In general, OAM provides network operators the ability to monitor the health of the network and quickly determine the location of failing links or fault conditions. It uses OAM PDU's, whichs traverse a single link and as such, are not forwarded by MAC clients (e.g., bridges or switches).

Ethernet CFM (IEEE 802.1ag)

The IEEE 802.1ag CFM standard [IEEE802.1ag] specifies protocols, procedures, and managed objects to support transport fault management. This allows for the discovery and verification of the path, through bridges and LANs, taken by frames addressed to and from specified network users and the detection, and isolation of a connectivity fault to a specific bridge or LAN. Ethernet CFM defines proactive and diagnostic fault localization procedures for point-to-point and multipoint Ethernet Virtual Connections that span one or more links. It operates end-to-end within an Ethernet network.

802.1ag provides hierarchical network management to cope with (stacked) VLANs in carrier grade ethernet networks, it has a couple of features:

Continuity Check Protocol (CCP) "Heartbeat" messages for CFM. The Continuity Check Message (CCM) provides a means to detect connectivity failures. CCMs are unidirectional multicast messages confined to a "Management Domain".

Link Trace (LT) messages otherwise known as "Mac Trace Route" are Multicast frames that are transmitted to track the path (hop-by-hop) to a destination node, similar in concept to User Datagram Protocol (UDP) Trace Route. Each receiving node sends a Trace Route Reply directly to the Originating MEP, and regenerates the Trace Route Message.

Loopback (LB) messages otherwise known as "MAC ping" are Unicast frames that a node transmits, they are similar in concept to an Internet Control Message Protocol (ICMP) Echo (Ping) messages, sending Loopback to successive nodes can determine the location of a fault. Sending a high volume of Loopback Messages can test bandwidth, reliability, or jitter of a service, which is similar to flood ping. Unlike CCMs, Loopback messages are administratively initiated and stopped.

Failure detection in RINA

A failure detection policy for flows would be a useful thing to have. The benefits of BFD are that it is simple and somewhat follows the mechanism/policy split as advocated in

RINA. BFD checks the continuity of a path, in RINA it could be used for flows between IPC processes. Note that this policy should therefore be implemented as close to the physical infrastructure as possible (i.e. DIFs of lower rank) to have sufficient correlation between flows. The higher the rank of a DIF, the more different flows could be routed differently between two processes. BFD has some features that it does not need to perform explicitly when applied to RINA. E.g. as RINA's core transport protocol is based on delta-t, the three way handshake is unnecessary, and as IPC processes were authenticated at enrolment, no additional authentication would be needed.

A feature that would be interesting is the LoopBack function. In order to cope with the independence of flows, when a failure is suspected, the IPC process can signal its neighbour to go in Loopback mode. In this mode the neighbour will forward all traffic it receives from the originating IPC process's source address back.

Traceroute is a method for locating failures from the end-user perspective, usually across different networks. In RINA this functionality may not be as useful as it would either be violating the DIF boundary (tracing flows in a lower (N-1)-DIFs is not allowed) or trace a route within the application. It could be used as a debugging tool during development, but should not be required in an operational RINA deployment.

7.1.2. Recovery in packet switched networks

Restoration

Link State Routing

A link-state routing protocol is one of the two main classes of routing protocols used in packet switching networks for computer communications (the other is the distance-vector routing protocol). Examples of link-state routing protocols include Open Shortest Path First [RFC2328] [RFC5340] and Intermediate System to Intermediate System (IS-IS) [RFC1142].

Link-state protocols determine adjacency using a reachability protocol (also referred as the "HELLO" protocol), and distribute their local reachability information through Link State Advertisements (LSAs). By combining all the reachability information it received from LSA's, each node builds a map of the network and calculates next hops to each known destination (the Routing Table) using a shortest path algorithm. When all nodes participating in link-state routing have the same view of the network, it is said that the protocol has converged.

Link-state protocols are resilient to failures as reachability is checked periodically and the network will re-converge if changes in reachability are detected. The time (and

stability) of reconvergence is dependent on the size of the network and the interval at which reachability is checked. In general, link-state protocol reconvergence times (order of seconds to minutes) are not fast enough to meet carrier recovery requirements (order of tens of milliseconds).

Multi-homing

IPv4/IPv6 Multihoming

Multihoming has its applications for providing resiliency. If one interface or its attached link fails, the offered service is still available via another interface of a multihomed host. However, in IP or Ethernet some additional configuration is required in order to support multihoming hosts (Autonomous Systems can be multihomed using BGP, but this is considered out of scope for now).

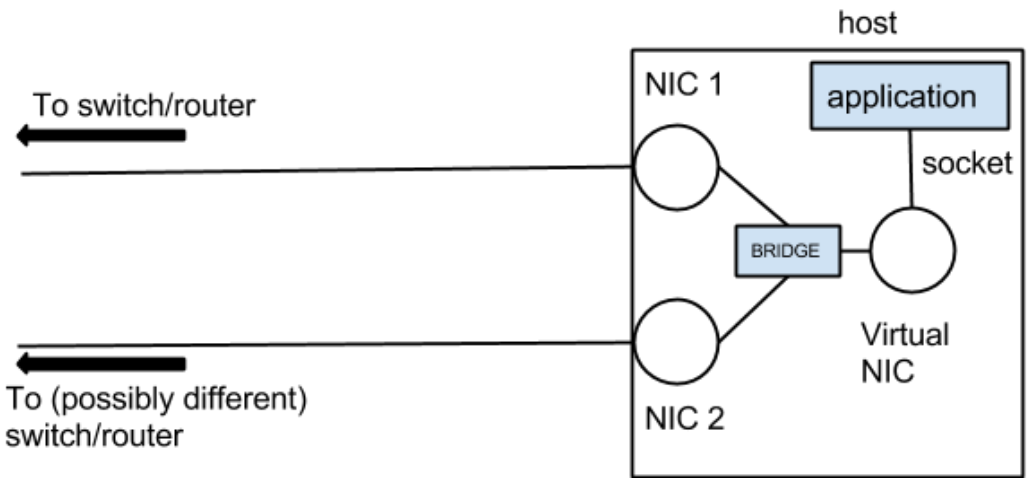

Figure 30. Resilient setup

A multihomed setup of a host in IP is shown in Figure 30, “Resilient setup”. In this setup, NIC 1 and NIC 2 are bridged towards a virtual interface, to which the application is bound. In this case, if either NIC 1 or NIC 2 fails, the application can still be reached (after convergence of the routing protocol). Moreover, since this is done at the network layer, the NICs can be connected to a different router, eliminating one more single point of failure.

Link Aggregation

Link aggregation [IEEE802.3ad] is a technique to aggregate multiple network connections in order to increase the maximum throughput. By doing this, resiliency is also provided. If one network connection fails, the other(s) still deliver(s) traffic.

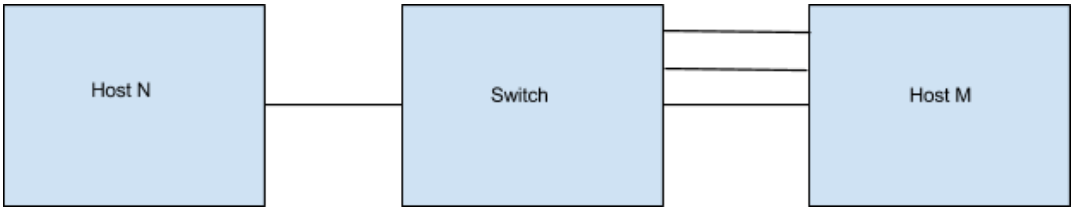

Figure 31. Link aggregation on layer 2

Aggregation is performed between switch ports, either physical or virtual ones. Note that to provide resiliency at least 2 physical NICs are required. An example of link aggregation at layer 2 is shown in Figure 31, "Link aggregation on layer 2". In this case however, the switch is a single point of failure. If it fails, no failover is available.

Multi-path TCP

Multipath TCP (MPTCP) is a set of extensions to regular TCP to enable a transport connection to operate across multiple paths simultaneously. MPTCP assumes that the presence of multiple addresses at a host is sufficient to indicate the existence of multiple paths which do need not be entirely disjoint. However, even in such a situation, making use of multiple paths is beneficial, improving resource utilization and resilience to a subset of node failures.

MPTCP operates at the transport layer on top of (multiple) standard TCP connections (called subflows).

An interesting point is raised in the Architectural Guidelines for MPTCP [RFC6182] relating to sequence numbering. MPTCP uses two levels of sequence spaces: a connection-level sequence number and another sequence number for each subflow. This permit connection-level segmentation and reassembly and retransmission of the same part of connection-level sequence space on different subflow-level sequence space. The alternative approach, using a single connection-level sequence number, which gets sent on multiple subflows, has two problems: first, the individual subflows will appear to the network as TCP sessions with gaps in the sequence space. Second, the sender would not be able to attribute packet losses or receptions to the correct path when the same segment is sent on multiple paths (i.e., in the case of retransmissions).

IP Fast ReRoute

IP Fast ReRoute: Loop Free Alternates [RFC5286] is a mechanism that enables a router to rapidly switch traffic following an adjacent link and/or node failure, towards a pre-computed/pre-programmed loop-free alternative (LFA) path. The goal of LFA FRR is to reduce failure reaction time to tens of milliseconds by using a pre-computed alternate

next-hop, in the event that the currently selected primary next-hop fails, so that the alternate can be rapidly used when the failure is detected. A LFA exists for a certain source node s to a destination node d via neighbour n adjacent to s if:

dist(n,d) < dist(n,s) + dist(s,d)

Where dist(x,y) is the length of the shortest path from x to y.

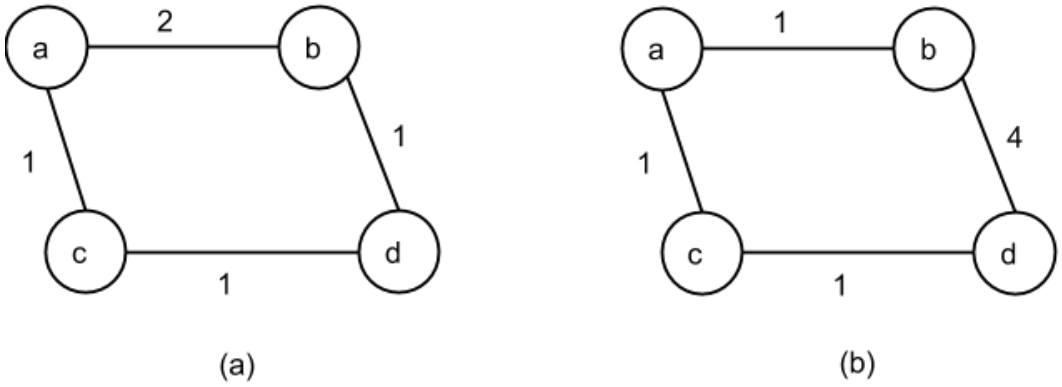

Figure 32. Example of Loop Free Alternates

An example is given in Figure 32, “Example of Loop Free Alternates”. The shortest path from a to d is via c. In Figure 32, “Example of Loop Free Alternates”(a), b is a Loop Free Alternate of a for destination d: if we send traffic to b instead of c the traffic would still reach destination d without passing through a. In Figure 32, “Example of Loop Free Alternates” (b), this no longer holds, since dist(b,d) > dist(b,a) + dist(a,d). The shortest path from b to d is back through a and c.

MPLS Fast ReRoute

Multi-Protocol Label Switching Fast ReRoute is a mechanism that uses ReSerVation Protocol with Traffic Engineering extensions (RSVP-TE) signaling to establish backup tunnels between Label Switched Routers (LSRs). The endpoints are called the Point of Local Repair (PLR) and the Merge Point (MP) respectively. MPLS Fast-ReRoute supports One-to-One Backup, where a backup LSP is created for each protected LSP at the PLR, and Facility Backup, where a bypass LSP is protecting one or more protected LSPs that traverse the PLR, the resource being protected, and the Merge Point in that order. In facility protection, if a bypass LSP is protecting a link, it is called an NHOP (Next-hop) bypass tunnel, if it it is protecting a node, it is called an NNHOP(Next-Next-hop) bypass tunnel. [RFC4090].

Circuit-oriented mechanisms have not yet been explored in RINA. Such DIF designs will be explored at a later stage in PRISTINE.

Recovery in RINA

In RINA, sending packets over multiple paths (multihoming) does not raise additional issues. Routing takes precedence over the flow allocation: packets within an EFCP flow can be routed over different paths in an (N-1) DIF or over different (N-1) DIFs, and will be assembled at the endpoints by the EFCP instance in the N-DIF. Inside each of the different (N-1)-DIFs, the EFCP instance takes care of reordering within the DIF (although this is strictly speaking not needed). In any case, reordering and retransmission should not happen in both DIFs, good design would do retransmission and reordering only in the N-DIF, and provide flow control in the (N-1)-DIF.

A protocol for performing Loop-Free Alternates seems like a good first step to create baseline protection functionality for RINA deployments. A policy to perform this is given in the next section.

Load Balancing

Effective load balancing systems need to consider not only congestion within the network, but also load on the servers. In order to test load balancing on applications we propose a use case based on minimising response time of requests to applications servers (e.g. HTTP) by controlling load on both the network and the servers by using RINA. It is common practice for large web sites to balance load simultaneously over many application servers.

Load balancing techniques may be oblivious (e.g., spreading requests equally over all servers, without consideration for their load), or stateful (e.g., sending the request to the least loaded server). In a data-center or a dedicated web-hosting service, the application servers are connected by a regular, over provisioned network; the load balancer usually does not consider the network state when load balancing across servers which is highly wasteful of resources.

A simulated scenario based on this use case is being developed. It will use the RINA simulator (i.e. RINASim) for demonstrating RINA's load balancing capabilities that involves balancing traffic across 3 application servers with 1,000 clients. The RINASim will implement the necessary DIF components that includes the functionality of FA, FAI, RMT (with static PFT), EFCP (DTP transfer and basic policies) and RA. There will be two methods available to generate traffic for the simulation. The first method is to implement a separate simulation module that will actually "generate" traffic conforming to the traffic class pattern (with preset transport protocol, randomly chosen byte length varying between certain values, etc.). Nevertheless, this approach is feasible

only for inspecting the impact of the traffic on the network because it is meaningless for applications. The second method is to use real captured traffic as a template that will feed into the application simulation module to produce comparable messages. The latter method is more complicated for implementation (compared to the first method described), but produces more relevant results by also considering the application's perspective. From the use case description, the strategy will be to observe how 3 servers handle traffic from 1000 clients no matter what the actual application protocol using it is. Hence, the first method seems more convenient for this kind of simulation.

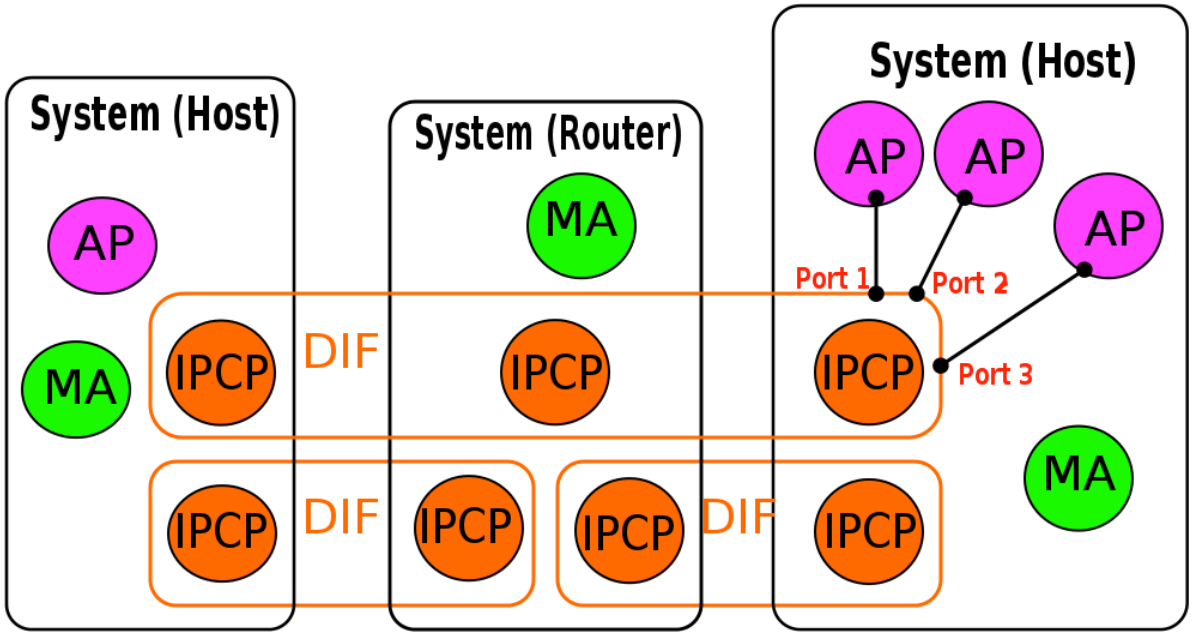

Figure 33. Load-balancing applications

A simple scenario would involve two computer systems where the application in the first node communicates with the application in the second node. In order to demonstrate a realistic load balancing scenario the number of nodes (i.e. clients) communicating will be increased from the initial two nodes to 1,000 or more nodes. The basic steps involved in this scenario would involve the application requesting the IPC manager specifically the IRM to:

1. AE requests IRM for communication.
2. IRM chooses a DIF from the static DA's Directory.
3. IRM passes allocation request to a given DIF's FA.
4. FA creates FAI.
5. FAI allocates ports, creates EFCPI, prepares RMT queues and binds data paths between modules.

6. The exact route is determined, by querying load balancing specific objects within the DAF, to determine an appropriate destination node. (node load balancing)
7. FAI sends create (flow) request to the selected destination node and local processing for the new flow is carried out. One of the tasks is to ensure that the flow is allocated to the most appropriate AP, by selecting the most appropriate local port. (AP load balancing on a single node)
8. AE can then use the flow to behave like a typical client application.
9. AE asks IPC process to de-allocate flow which passes it towards a given FA.
10. Follow up FA cleaning process.

In order to demonstrate that 8) is working, a simple application protocol will be implemented that uses CDAP Read and CDAP Read_R messages to simulate behaviour of ICMP Echo Request/Reply and test connectivity across DIF.

RINASim will include all the desired functionalities required to demonstrate the scalability of this type of scenario. Nevertheless, scalability will depend on the types of "RINA" devices required and the topology size for actual simulation execution where a larger topology means longer time to simulate all events and more memory/disk space and processor power. Computer resources pose the limits for the scale of such a simulation where it is common to simulate computer networks with hundreds of nodes within OMNeT++. Currently under development for RINASim are extensions that include a host-like simulation module that can communicate with directly connected peers and support for simulating Interior Routers and Border Routers.

7.2. Policies for Failure Detection

This section documents two policies for failure detection that apply to the Flow Allocator (FA) and its related components. Flow Liveness Detection (FLD) is a policy that is executed to detect whether a flow is up or down by keeping remote watchdogs. Flow Loopback Request (FLR) is a policy that is only executed upon request (e.g. when error conditions are suspected). If FLR is activated, all PDUs sent on that flow are looped back as soon as they arrive. In this way, a flow is also checked for liveness and packet loss.

Due to the additional functionalities added to the Flow Allocator (FA), we also propose to rename the Flow Allocator Instance to the Flow Manager (FMGR). For simplicity, this renaming is assumed in the following. We also suggest a new component, the Flow Monitor (FMON), which monitors the flow during its lifetime. The FMON described in this section only targets FLD and FLR for recovery purposes thus advanced functions

- e.g. such as average/min/max bandwidth, delay, jitter characteristics - are not considered at the moment.

Therefore, the following two figures represent the current Figure 34, "Current situation" and proposed Figure 35, "Proposed situation" situation respectively.

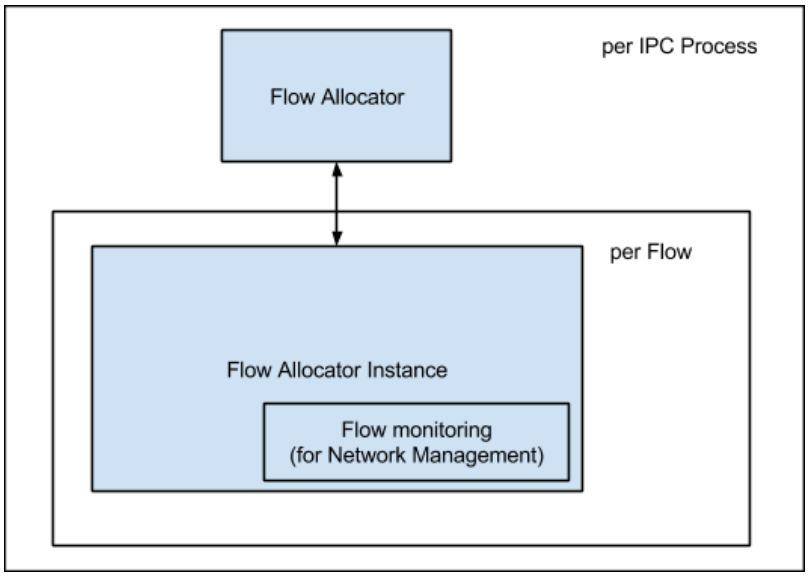

Figure 34. Current situation

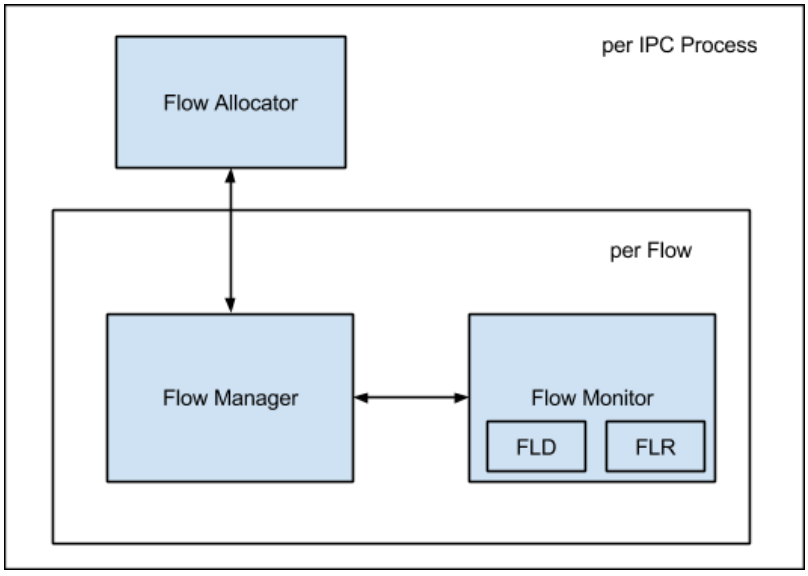

Figure 35. Proposed situation

Therefore, the definitions that follow are assumed in the remaining text [8]:

- Flow Allocator - The component of the IPC Process that responds to Allocation API invocations from Application Processes by optionally performing authentication,

[8] Note that the Flow Allocator definition has been updated while the Flow Manager substitutes the Flow Allocator Instance

assigning a port-id and creating a Flow Manager to manage the flow during its lifetime.

- Flow Manager - A Flow Manager is created for each allocation request to manage the flow for its lifetime. It will translate the QoS requested by the Application Process into specific policies and find the destination Application and determine if the allocation can be honored. The Flow Manager keeps state of the flow. A Flow Manager Identifier (FMGRI) or port-id is returned to the application as a handle for referencing the allocation.

7.2.1. Flow Liveness Detection

Flow Liveness Detection detects if a flow between IPC processes is alive or not by sending periodic messages. When FLD is present, the Flow Manager keeps two additional states for the flow - i.e. UP and DOWN. FLD maintains a timer that is reset upon reception of such a periodic message. The flow is declared DOWN if the timer expires, otherwise it is declared UP.

Common elements

The procedures described in the remaining sections, rely on the following common elements:

FLD elements:

```
Watchdog:
    Timeout : Timer

FLD data:
    port-id : Port-id
    watchdog : Watchdog
    interval : Int (milliseconds)

RIB objects:

../fld/<neighbour-address>-<address>/<connection-id>
Timeout : Double

../fld/<address>-<neighbour-address>/<connection-id>
Timeout : Double
```

A RIB object containing a timeout value - i.e. ../fld/<neighbour-address>-<address>/<connection-id> - is periodically updated with a new timeout value on each

corresponding CDAP M_WRITE. FLD subscribes to changes to this object and is thus notified when it has been changed. The timer - i.e. the Watchdog - is then restarted with the new timeout value. If the Watchdog expires the FLD notifies the FMGR that the flow is DOWN.

Initialization

The Timeout value for Watchdog has to be chosen depending on the DIF. Most likely it will be a function of the Round Trip Time (RTT). For initialization of the FLD, the following steps are followed:

- Firstly, FLD will subscribe to changes to the RIB object ../fld/<address>-<neighbour-address>/<connection-id> through the RIB Daemon, where <connection-id> is the connection-id that identifies the flow with the peering IPC process.
- Secondly, FLD will ask the RIB Daemon to periodically, every Interval milliseconds, replicate ../fld/<neighbour-address>-<address>/<connection-id> to the peer's RIB.
- Finally, the Watchdog timer is started.

FLD Behaviour

Watchdog_Timer.expire

When invoked

Whenever the Watchdog timer expires.

Action upon invocation

The FMGR is notified that the flow should be declared DOWN.

Timeout_Changed.receive

When invoked

Upon changes to ../fld/<address>-<neighbour-address>/<connection-id>

Action upon receipt

The Watchdog timer is re-armed with the communicated timeout value. Communicating a 0 timeout is allowed and implies declaring the flow as DOWN

immediately. This could be used for interrupting incoming traffic without deallocating the flow.

7.2.2. Flow Loopback Request Policy

The Flow Loopback Request (FLR) should be executed only under error conditions. The procedure is activated by sending a CDAP M_START message to a neighbour IPC process containing the connection-id to identify the connection of the flow to test. When an M_START_R is received back with a positive answer, all PDUs sent on that flow are looped back by the peering IPC process in order to asses the QoS level of the flow. After the monitoring traffic, an M_STOP CDAP message is sent to the neighbour IPC process. When it replies with a positive M_STOP_R, normal operations continue.

Common elements

```
State:
    ON
    OFF

Role:
    INITIATOR
    LOOPBACK

FLR data:
    port-id : Port-id
    role : Role

RIB object:
../flr/<connection-id>

State : State
```

Initialization

Upon initialization, a role is set to Role, SERVER if it will be the sender of PDUs and client if it will receive them. If the Role SERVER was set, a PDU size is set in Size, and the amount of PDUs to send is set in Amount, Returned is set to zero.

FLR Behaviour

Loopback_Start.receive

When invoked

This is invoked upon receipt of an M_START<> to ../flr/<connection-id>

Action upon receipt

If Role is INITIATOR, FLR is to be started. FLR blocks all reads/writes from/to Port-id. Next, FLR will change the state of ../flr/<connection-id> to ON and send an M_START message on Port-id in order to set the LOOPBACK role on the peer. If no M_START_R is received back, an error is reported, ../flr/<connection-id> is set to OFF. If Role is LOOPBACK, all writes/reads to/from Port-id are blocked and a positive M_START_R is returned. If after 2 MPL no PDU is received, an error is reported and ../flr/<connection-id> is set to OFF. If LOOPBACK state is not accepted a negative M_START_R is returned. [9]:

M_START_R.receive

When invoked

Upon receipt of an M_START_R reply with regard to ../flr/<connection-id>

Action upon receipt

Loopback_send.submit is called. If a negative reply is received, it is reported and ../flr/<connection-id> is set to OFF.

Loopback_Send.submit

When invoked

upon a positive M_START_R reply with regard to the ../flr/<connection-id> RIB object.

Action upon receipt

If ../flr/<connection-id> is set to ON, FLR writes the testing traffic to Port-id. 2 MPL after the last PDU has been sent, Loopback_Stop.submit is called.

PDU_Receive.receive

When invoked

This is invoked when a PDU is received on Port-id.

[9] Loopback_Start.submit is invoked by the FMGR, not FLD, which starts the procedure with an M_START message to ../flr/<connection-id>

Action upon receipt

If ../flr/<connection-id> is set to OFF, the normal data path processing is followed. If ../flr/<connection-id> is set to ON, the following alternate processing is followed: If Role is LOOPBACK, PDU_Resend.submit is called.

PDU_Resend.submit

When invoked

This is invoked upon receipt of a PDU when ../flr/<connection-id> is set to ON.

Action upon receipt

The received PDU is re-sent on Port-id.

Loopback_Stop.submit

When invoked

This is invoked when all PDUs have been sent.

Action upon receipt

Upon receipt, if. ./flr/<connection-id> is set to ON, ../flr/<connection-id> is set to OFF and an M_STOP is sent on the ../flr/<connection-id> RIB object to replicate the state. If no M_STOP_R is received after 2 MPL, an error is reported.

Loopback_Stop.receive

When invoked

Upon receipt of an M_STOP to ../flr/<connection-id>

Action upon receipt

If ../flr/<connection-id> is set to OFF and a positive M_STOP_R reply is sent on Port-id. Port-id is no longer blocked for reads/writes. if ../flr/<connection-id> was OFF already, this should report an error.

M_STOP_R.receive

When invoked

Upon receipt of a positive M_STOP reply with regard to ../flr/<connection-id>

Action upon receipt

Normal operations continue, Port-id is no longer blocked for reads/writes. Statistics should be gathered and returned.

7.3. Policies for Resilient Routing

This section describes a per-hop link-state resilient routing policy, based on Loop-Free Alternates, to generate the PDU forwarding tables for the IPC processes in a Distributed IPC Facility (DIF). When a link-state routing policy is used, the following components are present to implement the PDU Forwarding Function:

- FSDB: The subset of the RIB that contains all the Flow State Objects known by the IPC Process.
- PFTG: Takes the Flow State Database as input to generate a Routing Table. With this Routing Table, a PDU Forwarding Table is generated.
- RT: Contains a set of routes towards all destinations.
- PFT: A table that maps a destination address to one or more port-ids of N-1 flows. The RMT uses this table to decide where to forward EFCP PDUs to.

Loop Free Alternates (LFAs) are based on the observation that, on a (positive weighted) graph, when a source S has a neighbour U for which the triangle inequality

```
d(U,T) < d(U,S) + d(S,T)
```

with destination T holds, the shortest path to the destination T as calculated on the graph from U will never pass through S. All such IPC processes U adjacent to S are called Loop-Free Alternates of S for T.

In RINA, multiple (N-1)-flows can exist between IPC processes. The condition can thus translate to

```
min( d(U,T) ) < max( d(U,S) ) + max( d(S,T) )
```

where min and max are taken over the distances of all (N-1)-flows between the IPC processes.

This link-state routing policy subscribes to certain events that are affecting the local N-1 flows (local N-1 flows are flows that have the IPC Process as source or target), such

as the allocation, deallocation and changes in the status of these flows (N-1 flow up/ N-1 flow down) due to failures.

The Flow Allocator Instance and the Flow Monitor [see spec] components of the IPC Process are responsible for throwing these events.

This specification assumes the usage of a flat addressing scheme, which means no hierarchy or partitioning is imposed on the addresses.

7.3.1. Definition of Terms

Connection - the shared state between EFCPMs. The endpoints of a connection are identified by CEP-ids.

Connection Endpoint Id (CEP-id) - An identifier unambiguous within the scope of an IPC Process that identifies an EFCPM-instance.

Distributed IPC Facility (DIF) - A distributed application consisting of at least one IPC process in each participating processing system. The DIF provides IPC services to applications via a set of API primitives. The cooperating IPC processes manage the Distributed IPC Facility.

Error and Flow Control Protocol (EFCP) - The data transfer protocol required to maintain an instance of IPC within a DIF. The functions of this protocol ensure reliability, order, and flow control as required.

EFCPM - The Error and Flow Control Protocol Machine is the task that instantiates an instance of the EFCP for a connection. An EFCPM consists of two state machines loosely coupled through a single state vector: one that performs the tightly bound mechanisms, referred to as the Data Transfer PM; and the other that performs the loosely coupled mechanisms, referred to as the Data Transfer Control PM.

Flow State Database (FSDB) - The subset of the RIB that contains all the Flow State objects known by the IPC Process. It is used as an input to a link-state routing policy.

Flow State Object (FSO) - An object describing the state of an N-1 flow between two IPC Processes. It contains source and destination addresses, QoS-id, status and associated data to avoid stale, duplicated and obsoleted data in the flow state database.

IPC Process - An application process that is a member of a DIF.

Flow - The binding of a connection to source and destination ports.

(N)-DIF - The DIF from whose point of view a description is written.

Protocol Data Unit (PDU) - The string of octets exchanged among the Protocol Machines (PM). PDUs contain two types: Protocol Control Information (PCI), which is understood and interpreted by the DIF, and User-Data, that is incomprehensible to this EFCP and is passed to its user.

PDU Forwarding Function (PFF) - The function that is consulted by the RMT to decide whereto a PDU should be forwarded.

PDU Forwarding Table - An instantiation of the PDU Forwarding Function in the form of a table that maps a destination address to one or more port-ids of N-1 flows. The RMT uses this table to decide where to forward EFCP PDUs to. The PDU Forwarding Table generator is the task in charge of creating and updating the PDU forwarding table.

PDU Forwarding Table Generator (PFTG) - Takes the Flow State Database as input to generate a Routing Table. With this Routing Table, a PDU Forwarding Table is generated.

Relaying/Multiplexing-Task (RMT) - This task is an element of the data transfer function of a DIF. Logically, it sits between the EFCP and SDU Protection. RMT performs the real time scheduling of sending PDUs on the appropriate (N-1)-ports of the (N-1)-DIFs available to the RMT.

Resource Information Base (RIB) - The logical representation of information held by the DIF, necessary for the operation of the DIF

RIB Daemon - A local Application Process participating in a Distributed Application may have several sub-tasks or threads. Each of these may have requirements for information from other participants in the distributed application on a periodic or event driven basis. Hence, a common component of the Application Process is a RIB Daemon, which other tasks may request to get information from other members of the DAF either on demand, on an event, or periodically. The RIB Daemon can then optimize these requests for information.

Routing Table (RT) - Contains a set of routes towards all destinations.

7.3.2. Narrative description of the Loop Free Alternates policy

The Flow State Database

The Flow State Database is the subset of the RIB that contains all the Flow State Objects (FSOs) known by the IPC Process. It is used as an input to calculate the Routing Table. The FSDB consists of the operations on FSOs received through CDAP messages.

RIB Objects:

Flow State Object (FSO)

The object exchanged between IPC Processes to disseminate the state of one N-1 flow supporting the IPC Processes in the DIF. This is the RIB target object when the PDU Forwarding Table Generator wants to send information about a single N-1 flow.

```
../fsdb/<address>/<neighbour_address>/<QoS> : flowstateobject
    address          /* The address of the IPC Process */
    neighbour_address /* The address of the neighbour IPC Process */
    QoS-cube         /* The QoS of this N-1 flow */
```

Routing Table

Based on the FSDB, a graph of the connectivity in the DIF is constructed. From this graph, a routing table can be calculated for every QoS cube in the DIF. However, in this specification, only the shortest route is calculated using Dijkstra, using hop count as the metric for distance. Apart from this, for every node, the Loop Free Alternates are also calculated. Node Protecting Loop Free Alternates are preferred over Link Protecting Loop Free Alternates. An example connectivity graph is shown in Figure 36, “An example connectivity graph”, and its corresponding routing table as calculated by A is shown in Table 10, “Routing table of IPC process with address A”. Note that from A to B there are 2 N-1 flows with different QoS.

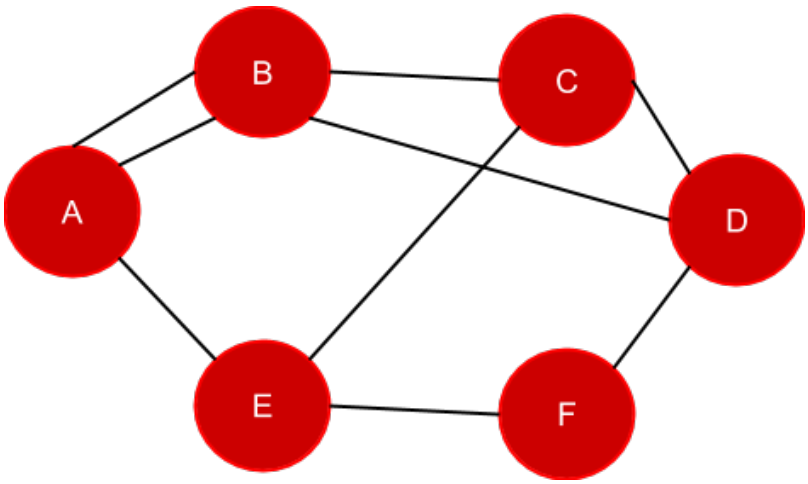

Figure 36. An example connectivity graph

Table 10. Routing table of IPC process with address A

Destination Address	Next Hop	LFA
B	B	B

Destination Address	Next Hop	LFA
C	B	E
D	B	E
E	E	B
F	E	B

PDU Forwarding Table

Based on the routing table, the PDU forwarding table is calculated in each node. In essence, this is the mapping of the next hop on a port-id. In the example, suppose there are 2 flows to B from A, with port-id 1 and 2, and there is one flow from A to E with port-id 3. Then a generated forwarding table could look as follows:

Table 11. Forwarding table of IPC process with address A

Destination Address	Port-id	LFA
B	2	1
C	2	3
D	1	3
E	3	1
F	3	2

This table is then consulted by the Relaying and Multiplexing Task (RMT) to decide on what port-id the PDU should be written.

Subscription and reaction to events

Upon initialization of the PFT, the PFT subscribes to certain events of the RIB daemon. This makes the PDU Forwarding Table Generator completely event based. The cooperation between these tasks in the IPC process is depicted in Figure 37, “Cooperation of tasks in the IPC process”. These events are:

- N-1 flow allocated
- N-1 flow deallocated
- N-1 flow up
- N-1 flow down
- Flow State Database has changed

Apart from subscribing to these events, the PFT marks all objects in the FSDB to be replicated upon changes.

N-1 flow allocated

When invoked

This is an event that indicates a new N-1 flow to a neighbour was allocated.

Action upon receipt

A Flow State Object is created, containing the address of the IPC process and the address of the neighbour IPC process where the flow is allocated to. The QoS is set to the QoS of the flow. The FSO is added to the FSDB unless there is already an FSO present with the same addresses and the same QoS.

N-1 flow deallocated

When invoked

This is an event that indicates an N-1 flow to a neighbour was deallocated.

Action upon receipt

Upon receipt, the FSO corresponding to the addresses and QoS of the flow is removed, unless there is another neighbour flow with the same addresses and QoS present in the IPC process. If the port-id of the flow is present in the forwarding table, the LFA is used until a new forwarding table is generated.

N-1 flow up

When invoked

This is an event that indicates an N-1 flow is up again.

Action upon receipt

If there is a Delete_FSO timer corresponding with this flow, it is stopped. Else, a Flow State Object is created, containing the address of the IPC process and the address of the neighbour IPC process where the flow is allocated to. The QoS is set to the QoS of the flow. The FSO is added to the FSDB unless there is already an FSO present with the same addresses and the same QoS.

N-1 flow down

When invoked

This is an event that indicates an N-1 flow to a neighbour is down.

Action upon receipt

The Delete_FSO timer is started on this flow. Note that this time should be chosen reasonably small.

Delete_FSO expires

When invoked

This is invoked when the Delete_FSO timer fires.

Action upon receipt

The Flow State Object corresponding with this flow is deleted, unless there is another neighbour flow with the same addresses and QoS present in the IPC process. If the port-id of the flow is present in the forwarding table, the LFA is used until a new forwarding table is generated.

Flow State DB has changed

When invoked

This is an event that indicates there was a change to the Flow State Database.

Action upon receipt

Upon this event, the routing table is re-calculated. If there is already a calculation on-going it is stopped and restarted. After the routing table has been calculated, the forwarding table is generated from it.

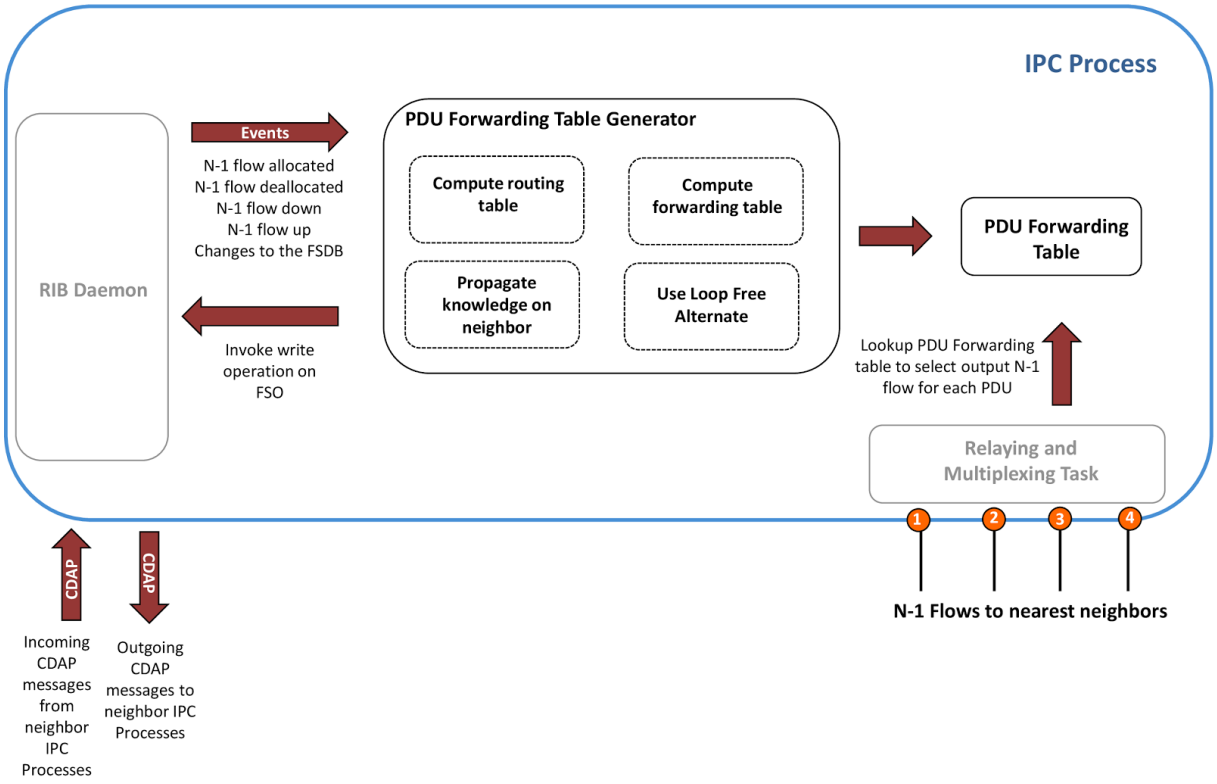

Figure 37. Cooperation of tasks in the IPC process

8. Summary and Conclusions

The primary aim of RINA is to provide a reliable communication service to end-users and providers. To this end, the security objectives are to protect the network and its resources (i.e., user data, management data and computing resources) from failures or attacks that are targeted to disrupt this communication service. This deliverable provided details of the RINA security solution, the functions and the relevant enablers to achieve the above objectives. These functions and enablers include: Authentication, Access Control, Secure Channel and SDU Protection, Key Management functions, monitoring and countermeasures for reducing the security risks and combating the threats. This deliverable also looked at network resiliency and availability in RINA.

Six different authentication policies have been proposed in this deliverable, ranging from a simple policy with no authentication, trusting the application names exchanged in the connect messages, to a policy requiring the support of PKI mechanisms. The proposed authentication policies have been discussed and sequence diagrams have been used to show how authentication can be achieved in RINA.

Access control policies define the rules specifying the conditions under which authorised subjects can have access to objects. A Capability Based Access Control model was explained and selected for design and implementation in PRISTINE. With regard to Multi-Level Security, different MLS architectures and practical scenarios have been identified and the ways that we can achieve MLS for content security given the RINA networking model have been thoroughly discussed. It was shown that RINA is a promising architectural framework for allowing the enforcement of multi-level content security in a more managed way through the configuration of RINA internals.

The requirements for setting-up a secure channel in RINA have been given. This is needed to: protect the messages exchanged when an IPC Process joins a DIF; to allow keys to be negotiated per session; to enable application data to be protected prior to joining a DIF; to provide mechanisms to manage keys; and to enable the destination IPC Process to be authenticated, preventing man in the middle attacks. An example of an existing secure channel protocol has been used to extract all of the functionality that is required. This functionality has been mapped to the RINA specifications to identify how RINA components could be used and what additional functionality is needed to achieve a secure channel.

SDU protection is used to protect the integrity and confidentiality of DTP traffic when passed as an SDU to an underlying IPC Process. The required algorithms that should be used and how protection is applied are defined in the SDU protection

policy has been discussed. An example of an existing data protection protocol has been analysed to extract all of the functionality that is required with some suggestion for modifications so that more robust security methods can be applied to protect application communication from eavesdropping and tampering.

A trusted entity is needed to generate, maintain and distribute keys to relevant processes within the RINA infrastructure. These keys are used to authenticate requests and encrypt/decrypt data. Two architectural options have been suggested for assuming the role of this security sensitive entity (Key Server): the Centralised and Distributed Key Management Architectures. The functionality and pros/cons of both architectures have been explained and discussion is still under way for selecting the preferred architecture for design and implementation.

We introduced a risk assessment methodology for combating threats and vulnerabilities in RINA. We have identified a comprehensive set of threats to the RINA assets, their impacts, the threat scenarios, the likelihood occurrence of each scenario, and the associated security risks. We then defined a set of security controls to reduce the risks to an acceptable level and mitigation actions to be put in place.

Maintaining the network resiliency in the case of failures and attacks and ensuring high-availability of the network for providing assumed services are set as the main objectives for RINA. In this deliverable methods for improving resiliency have been fully explained, specifically how to deal with IPC and link failures and exploitation of vulnerabilities.

We will advance our research investigation, especially on the subjects identified, and provide some of the relevant specifications to be used for the implementation of security functions and controls in the next version of this deliverable.

9. References

[ABAC] http://www.axiomatics.com/solutions/role/business-managers/abac-beyond-rbac.html

[Anderson01] R. Anderson et al., “Security Policies”, Advances in Computers: 55, Academic Press, Volume 55, pages 186-237, 25 July 2001.

[Benantar2006] M. Benantar, Ed., “Access Control Systems – Security, Identity Management, and Trust Models”, Springer Book, 2006.

[CC] “Common Criteria for Information Technology Security Evaluation – Part 3: Security assurance components”, ISO/IEC 15408, Version 3.1, Revision 3, Final, July 2009.

[Close2009] T. Close. ACLs don’t. HP Laboratories Technical Report, February 2009.

[D4.1] PRISTINE Consortium. Delivrable-4.1. Draft conceptual and high-level engineering design of innovative security and reliability enablers. September 2014.

[Day2007] Day, J. (2007). Patterns in Network Architecture: A Return to Fundamentals (p. 464). Pearson Education, Inc.

[Dilloway2008] Dilloway, C., & Lowe, G. (2008). Specifying secure transport layers. In Proceedings - IEEE Computer Security Foundations Symposium (pp. 210–223).

[Gollmann05] D. Gollmann, Computer Security, Second Edition, John Wiley & Sons, November 2005.

[IEEE802.1ag] "IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks Amendment 5: Connectivity Fault Management," IEEE Std 802.1ag - 2007 (Amendment to IEEE Std 802.1Q - 2005 as amended by IEEE Std 802.1ad - 2005 and IEEE Std 802.1ak - 2007) , vol., no., pp.1,260, 2007

[IEEE802.3ah] "IEEE Standard for Information technology-- Local and metropolitan area networks-- Part 3: CSMA/CD Access Method and Physical Layer Specifications Amendment: Media Access Control Parameters, Physical Layers, and Management Parameters for Subscriber Access Networks," IEEE Std 802.3ah-2004 , vol., no., pp.1,640, Sept. 7 2004

[IEEE802.3ad] "Amendment to Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications-Aggregation of Multiple Link Segments," IEEE Std 802.3ad-2000 , vol., no., pp.i,173, 2000

[IPsec] "Security Architecture for the Internet Protocol", IETF Network Working Group, RFC 4301, December 2005.

[LINK] “Interactive Link Data Diode – Connectivity Without Compromise”, Datasheet, downloaded from http://www.baesystems.com/ProductsServices/bae_prod_serv_aus_interactive.html

[Keith2012] Keith, M. (2012). Everyday Cryptography: Fundamental Principles and Applications. Oxford University Press. p. 114.

[Neuman2005] C. Neuman, T. Yu, S. Hartman, and K. Raeburn, “The Kerberos Network Authentication Service (V5)”, RFC 4120 (Proposed Standard), July 2005. Updated by RFCs 4537, 5021.

[Oppliger2009] Oppliger, R. (2009). SSL and TLS: Theory and Practice. Artech House.

[Pouzin2013] Pouzin Society, 2013. The RINA specification handbook.

[RFC1142] D. Oran (February 1990). "OSI IS-IS Intra-domain Routing Protocol", RFC 1142. Internet Engineering Task Force (IETF).

[RFC2104] Krawczyk, H., Bellare, M. & Canetti, R. (1997). HMAC: Keyed-Hashing for Message Authentication, RFC 2104.

[RFC2328] J. Moy (April 1998). "OSPF Version 2", RFC 2328. Internet Engineering Task Force (IETF).

[RFC2712] A. Medvinsky and M. Hur (October 1999). “Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)”. RFC 2712. Internet Engineering Task Force (IETF). Retrieved 9 October 2014.

[RFC3749] Hollenbeck, S. (2004). "Transport Layer Security Protocol Compression Methods". RFC 3749. Internet Engineering Task Force (IETF).

[RFC4090] P. Pan, G. Swallow and A. Atlas (May 2005). "Fast Reroute Extensions to RSVP-TE for LSP Tunnels". RFC 4090. Internet Engineering Task Force (IETF).

[RFC4279] P. Eronen and H. Tschofenig (December 2005). “Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)”. RFC 4279. Internet Engineering Task Force (IETF). Retrieved 9 October 2014.

[RFC5054] D. Taylor et al. (November 2007). “Using the Secure Remote Password (SRP) Protocol for TLS Authentication”. RFC 5054. Internet Engineering Task Force (IETF). Retrieved 9 October 2014.

[RFC5246] T. Dierks and E. Rescorla (August 2008). "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246. Internet Engineering Task Force (IETF). Retrieved 9 October 2014.

[RFC5286] A. Atlas and A. Zinin (September 2008). "Basic Specification for IP Fast Reroute: Loop-Free Alternates", RFC 5286. Internet Engineering Task Force (IETF).

[RFC5340] R. Coltun, D. Ferguson, J. Moy and A. Lindem (July 2008). "OSPF for IPv6", RFC 5340. Internet Engineering Task Force (IETF).

[RFC5746] E. Rescorla et al. (February 2010) “Transport Layer Security (TLS) Renegotiation Indication Extension” RFC 5746. Internet Engineering Task Force (IETF). Retrieved 9 October 2014.

[RFC5880] D. Katz and D. Ward (June 2010). "Bidirectional Forwarding Detection (BFD)", RFC 5880. Internet Engineering Task Force (IETF).

[RFC6182] A. Ford, C. Raiciu, M. Handley, S. Barre, J. Iyengar (March 2011). "Architectural Guidelines for Multipath TCP Development", RFC 6182. Internet Engineering Task Force (IETF).

[RFC6347] E. Rescorla and N. Modadugu (January 2012). “Datagram Transport Layer Security Version 1.2”. RFC 6347. Internet Engineering Task Force (IETF). Retrieved 9 October 2014.

[RFC6520] R. Seggelmann et al. (February 2012). "Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS) Heartbeat Extension". RFC 6520. Internet Engineering Task Force (IETF). Retrieved April 8, 2014.

[RFC7366] P. Gutmann (September 2014). “Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)”. RFC 7366. Internet Engineering Task Force (IETF). Retrieved 9 October 2014.

[SecRAM] SESAR ATM SecRAM Implementation Guidance Material - Project 16.02.03 D03, SESAR Joint Uundertaking, www.sesarju.eu, 2013

[SecureChannel] Wikipedia, Secure channel, http://en.wikipedia.org/wiki/Secure_channel, updated July 2014.

[SELinux] Security Enhanced Linux Project Page, http://selinuxproject.org

[Shamon] J. M. McCune et. al., “Shamon: A System for Distributed Mandatory Access Control“, 22nd Annual Computer Security Applications Conference 2006, ACSAC '06, IEEE, pp. 23-32, 26 December 2006.

[SP800-56B] NIST SP 800-56B: Recommendation for Pair-Wise Key Establishment Schemes Using Integer Factorization Cryptography, August 2009

[Ulgen1994] Ulgen, O. M., Black, J. J., Johnsonbaugh, B., & Klungle, R. SIMULATION METHODOLOGY - A PRACTITIONER’S PERSPECTIVE. International Journal of Industrial Engineering, Applications and Practice, 1(2), 1994.