Specific Targeted REsearch Project

PRISM

D2.2.1: High level System Architecture Specification

Project acronym: PRISM
Project full title: Privacy-Aware secure Monitoring
Contract No.: 215350
Project Document Number: IST-2007-215350-WP2.2-D2.2.1-R1
Project Document Date: January 16, 2009
Workpackage Contributing to the Project Document: WP2.2
Deliverable Type and Security: Public
Author(s): CNIT: G. Bianchi, A. Di Pietro, A. Detti, M. Pomposini, G. Procissi, S. Teofili
TLS: S. Rao, M. Matachowski, S. Kh avtasi
BAK: F. Gaudino
FHG: C. Schmoll
FTW: E. Hyytia, I. Gojmerac
HIT: B. Trammell, E. Boschi
ICCS: G. Lioudakis, F. Gogoulos, A. Antonakopoulou, D. Kaklamani, I. Venieris
NET: E. Stinco, A. Mancini
SRFG: F. Strohmeier, P. Dorfinger

Abstract:

This deliverable D.2.2.1, high-level system architecture specification, is the first of a series of two produced within Work Package 2.2. It describes and motivates the design principles of the PRISM project, provides a comprehensive system architecture and operation specified in terms of functions supported by system components and their interfaces, and provides a preliminary specification of individual components and functions, which will be complemented and finalised in the final architecture deliverable D.2.2.2.

Keyword list: PRISM, IST-2007-215350, Front-end, Back-end, architecture, interfaces, protocols, functionalities, system components.

Table of Contents

1 Introduction 7
2 Architectural principles 8
2.1 Two-stage approach 8
2.1.1 Flow isolation and extraction 8
2.1.2 Per-flow protection 9
2.1.3 Implications on the PRISM architecture and emerging trade -offs 10
2.2 Access control 10
2.2.1 Need for semantic specification 10
2.2.2 From semantic to authorisation 11
2.3 Monitoring flow chart 12
3 Architecture overview 14
3.1 Example scenario 14
3.2 Components of the PRISM architecture (FE, BE, PPC) 15
3.3 BE-FE interaction and characterisation of the roles 16
3.4 PRISM ontology 17
4 Front - end 19
4.1 Front – end control interface 20
4.2 Data plane manager (DPM) 21
4.3 Capturing unit 22
4.3.1 NP board 22
4.3.2 Host PCs 22
4.4 Processing units 23
4.5 Internal communication network 23
5 Back- end and privacy-preserving controller 25
5.1 Roles management and authorisation 26
5.2 Back-end internal architecture 27
5.3 Privacy-preserving controller 28
6 Semantic model 30
6.1 Description of PRISM ontology 30
6.1.1 `PersonalData` class 30
6.1.2 `Services` class 31
6.1.3 `Roles` class 32
6.1.4 `ExclusiveCombinations` class 32
6.1.5 `Conditions` class 33
6.1.6 `Rules` class 33
6.1.7 `Components` class 34
6.1.8 `DataTransformations` class 35
6.2 PRISM access control model editor 35
7 Data plane export protocol 38
7.1 Brief introduction to IPFIX 38
7.2 Data plane 39
7.3 Data export 39
7.4 Summary of data and control plane connections 39

8 Monitoring framework 41
8.1 Processing of modified source data 41
8.2 Hybrid pre-processing at the front-end 43
8.3 Full PRISM conversion 43
8.4 Proposed usage of the Measurement Infrastructure for Network research (MINER) in the PRISM prototype 43
9 Conclusions 46
References 47

Abbreviations

BE	Back end
DoS	Denial of Service
DPM	Data Plane Manager
FE	Front end
IANA	Internet Assigned Numbers Authority
IDP	Intrusion Detection System
IP	Internet protocol
IPS	Intrusion Prevention System
IPFIX	Internet Protocol Flow Information eXport
IXP	Internet eXchange Point
MA	Monitoring Agent
MPLS	Multi Protocol Label Switching
PKC	Public Key Certificate
PKI	Public Key Infrastructure
PMI	Privilege Management Infrastructure
PPC	Privacy Precerving Controller
PRISM	PRivacy-aware Secure Monitoring
PSAMP	Packet Sampling
PU	Clustrer Processing Unit
SLA	Service Level Agreement
SoA	Source of Authority
QoS	Quality of Service

List of Figures

Figure 1: Monitoring flow chart 13
Figure 2: Example scenario 14
Figure 3: Interaction channels of FE and BE 16
Figure 4: Front-end architecture 19
Figure 5: Back-end architecture 27
Figure 6: PRISM Ontology – Personal Data Subgraph 31
Figure 7: PRISM Ontology – Example of Access Control Rule 34
Figure 8: PRISM Access Control Model Editor: Personal Data Subgraph View 36
Figure 9: PRISM access control model editor: rules administration view 36
Figure 10: PRISM access control model editor: rule creation/e diting wizard. 37
Figure 11: Summary of schematic diagram of interactions in the PRISM architecture 40
Figure 12: Usage of MINER in the PRISM project 44

Terminology

Below is a list of frequently used terms together with a brief explanation.

actor	Human being running a traffic monitoring application.
analysis	Set of functions which process packet data into some result. Composed of one or more analysis functions, which are split between the front-end and back-end.
back-end analysis function	Function carried out at the back-end which takes as input some intermediate or final analysis result, as output by a front-end function or from the database, and returns some final analysis result, for later post-processing for presentation, export, or alert-generation purposes. A back-end analysis function is driven by the semantic access control system on the back-end.
front-end analysis function	Function which takes as input packet data and produces as output some intermediate or final analysis result; the output should be protected as necessary to meet the privacy requirements. A front-end analysis function may maintain internal state in order to derive its output from multiple packets. A front-end analysis function is driven largely by arriving packets, but may be driven by other events (e.g. link state changes on the front end or configuration) or by a timer (e.g., a periodic state clean up).
monitoring application	Software application or software agent that is used by an actor to submit a request to the back-end, and which also fetches the result.
monitoring application execution	The lifecycle of an application's execution; it may include several requests (e.g., an IDS application runs non-stop (execution); when triggered by an event, it submits requests). See also "request", below.
purpose	Purpose that a given monitoring application serves.
request (by a monitoring application to the back-end)	Autonomous, transactional request submitted by a monitoring application to the back-end; a monitoring application execution may include multiple requests.
requesting entity	The {actor, monitoring application} pair that submits a request, i.e., requesting entity $\in$ actors $\times$ monitoring applications.
role	Role that is assigned to an actor (e.g., network administrator) or to a set of actors (e.g., if we define the role *executive*, the *board of executives* constitutes another role that can be assigned to the logical/virtual actor comprised of all the executives)
service	Kind of abstraction for a monitoring application and/or purpose served; for example, PasTMon is a performance monitoring application, while the "measurement of SLA performance" is a service.

1 Introduction

The goal of the PRISM project is to design a general-purpose network traffic monitoring framework that enforces strong protection of personal data in monitoring applications. This is accomplished through a two-stage architecture, which separates monitoring tasks between a front-end, which observes traffic in the network, and a back-end, which stores and processes the results of the monitoring. The architecture effectively separates trust between these stages, applying new cryptosystems and data protection techniques to captured data as early in the monitoring process as possible, and using a fine granularity semantic access control scheme to the protected information. The access control is based on 1) the wider context of the information requested, 2) the entity making the request, and 3) the purpose for which the request was made. The core system provides the basic support for the development and deployment of privacy-aware monitoring applications; the project will develop a pilot implementation of this core system and, in addition, adapt selected monitoring applications to operate in concert with this core.

This document specifies the high level system architecture for the PRISM system. Section 2 introduces and justifies the basic principles upon which the PRISM system architecture is built. Section 3 presents a system-wide comprehensive overview of the proposed architecture, and discusses the interaction among the various architecture components. Sections 4 and 5 provide a first level of specification for the key system entities, namely the front-end, the back-end, and the privacy preserving controller). Section 6 introduces the key ideas behind the semantic model that manages most of the system's operation in terms of authorisation permissions and related policies. Section 7 describes the protocol interface (the so-called IPFIX) chosen for delivering data across the front-end and back-end entities, and for exporting data to third-parties. Section 8 describes how to interface with an external monitoring application on top of the PRISM architecture. The initial architecture is summarized in Section 9.

2 Architectural principles

The PRISM project aims to define a general-purpose, privacy-preserving network traffic monitoring system architecture. As such, the fundamental question we must answer before defining the principles of such architecture is ***what does privacy preservation mean in the context of network monitoring***?
We have considered this issue at length through an extensive discussion of the privacy issues arising from network monitoring [PRISM-D2.1.1], and the impact of these issues on the requirements for the system [PRISM-D2.1.2]. With the goal of establishing basic architectural principles in mind, we summarise this analysis as follows: ***The PRISM system achieves privacy preservation if it provides the means to effectively control access to monitoring data and monitoring results such that only the data and/or monitoring results strictly necessary to the operation of a monitoring task are available to that task***. This principle highlights the following key design aspects to be used as reference guidelines in the specification of PRISM's architecture:

1. We consider any approach wherein raw data is collected without provision for protection or concern for subsequent processing at collection time to be vulnerable. In other words, the separation between the data collection and data processing functions common in existing measurement applications inherently risks disclosure of the raw data, thereby potentially compromising privacy protections. Instead, PRISM views data collection and processing as a joint task, where data collection and protection is performed in the specific context of the usage of the data.
2. As a consequence of this design philosophy, the collection of data requires supplementary capabilities with respect to this context: it must be able to filter and pre-process data in order to extract the information that is strictly necessary for the monitoring task at hand out of all network traffic observed.
3. In order to achieve this discrimination, the architecture must provide for a formalisation of what "strictly necessary" means. This requires us to specify i) which specific subset of data must be accessed and/or which monitoring task shall be performed; ii) by a user with which rights, responsibilities, and roles with respect to the data and the monitoring location; iii) for which specific and lawful monitoring purposes; and iv) depending on which contextual information. All these aspects may vary over time during the monitoring process.

These principles can be met by designing PRISM as a two-stage system (as detailed in Section 2.1), and by providing a generalised access control framework (as detailed in Section 2.2)
.

2.1 Two-stage approach

PRISM's architecture is split into two broad stages; a front-end first stage which captures traffic and isolates, extracts and protects traffic data for further processing at a back-end second stage. In this section we discuss the implications of and rationale behind PRISM's two-stage architecture.

2.1.1 Flow isolation and extraction

Network traffic monitoring is generally devised to extract small pieces of useful information from a potentially very large volume of data. As such, reducing the amount of data to process and focusing the data processing on what is really meaningful is in itself a fundamental requirement before considering privacy issues, solely in terms of performance and scalability

issues. This is well illustrated by the following quote [1]: "*If we're keeping per-flow state, we have a scaling problem, and we'll be tracking millions of ants to track a few elephants* ". Privacy requirements merely strengthen the need to focus data collection and processing activities on a subset of the observable data, namely that which is strictly necessary to perform a specific monitoring task.

This observation leads to a system architecture composed of two stages: a front-end devised to collect, filter, and pre-process only the data strictly necessary for performing a specific monitoring task, and a second stage devised to process such a subset of pre-filtered data. This arrangement is a first step towards privacy preservation; as a consequence it additionally provides significant performance and scalability benefits.

In this arrangement, the front-end is placed as close as possible to the source of the data, on or very close to the traffic probe which collects packets from the observed network. The front-end isolates the traffic data of interest from all observable traffic, and extracts the information actually needed for a given measurement task. Many measurement tasks require only access to some small portion of the observed traffic (e.g. all flows of a given size), or some small portion of information about that traffic (e.g. a subset of the flow header fields). This isolation can be performed within the front-end, with no data about irrelevant flows being sent to the second stage. This isolation reduces the risk to end-user privacy though it does not eliminate it, as the isolated subset of data may still contain sensitive information and as such might require further control in the second stage; we address this in the next section. Nor is flow isolation necessarily an easy task to perform "on the fly"; we will address this challenge within the scope of PRISM work packages WP3.2 and WP4.1.

2.1.2 Per-flow protection

It is not always possible to completely eliminate traffic irrelevant for a given measurement task within the front-end through isolation, or to significantly reduce the information reported for each flow through extraction. In this case the information passed to the second stage requires additional protection to reduce risks to privacy.

Consider a monitoring task which needs to analyse all of the information collected for a given flow, but where flow isolation is technically possible only after the flow has been partially processed, or compared to information derived from other flows. A simple example of this situation would be traffic thresholding based on flow volume percentile. Here the operation requires access to some information (flow volume distribution) from every flow. The state required to support this operation as part of flow isolation at the front-end is prohibitive, especially on resource-constrained devices.

To address this issue, the front-end may in addition selectively or completely protect, through encryption, measured data on a per-flow basis, delivering this encrypted data to the second stage. According to the requirements of the specific measurement task, the second stage may then selectively decrypt *only the flows for which a detailed analysis is deemed necessary* . This selective encryption is to be enabled by using distinct encryption keys on a per-flow or per-class-of-flows basis, so that the second stage will be able to access only a subset of the data delivered by the front-end. A single encryption key for the whole traffic delivered to the second stage would yield an all-or-nothing approach.

Note that this issue may also be addressed in a slightly different way: not by decrypting the data selectively within the second stage, but by selectively projecting the operations to be done for the specific monitoring task into encrypted space, thereby extracting less privacy-sensitive information from the measured data without decrypting it. We note that this approach will require the application of novel cryptosystems, and must be applied in a case-by-case way to each specific task or set of primitive operations .

[1] Van Jacobson, End-to-end Research meeting, June 2000; quote reported in C. Estan, G. Varghese, "New Directions in Traffic Measurement and Accounting", SIGCOMM 2002.

Note that both of these approaches require some method for conveying keys to the second stage. Though this detail is not relevant from the standpoint of architectural principles, two potential solutions to this problem appear possible. i) A key repository that can be selectively accessed only if certain conditions emerge can be added to the PRISM architecture, or ii) key material may be embedded within the data delivered to the second stage, so that the key can be reconstructed only if certain conditions emerge. These approaches may be applied in a complementary fashion.

2.1.3 Implications on the PRISM architecture and emerging trade-offs

Implicit in the requirement to isolate flows, extract data from them, and selectively protect them within the front-end is the fact that, in contrast to other monitoring systems, there is ***no single front-end traffic capturing component valid for all situations***, but that the first stage must adapt to the specific monitoring task pursued. This further implies that when multiple measurement tasks are performed by the same system over the same observed traffic, there may be up to one logical[2] first stage per measurement task.

2.2 Access control

A two-stage approach predicated around the isolation, extraction, and protection of data ensuring that the measurement tasks access only the data that is strictly necessary requires a formal definition of "strictly necessary". This section addresses this requirement, and outlines how the system actually guarantees enforcement of access restrictions to the data.

2.2.1 Need for semantic specification

Network traffic monitoring is not a single, well specified activity. A monitoring task can be considered as a means to answer some single question about the activity of the network under monitoring. The thorough specification of which data is "strictly necessary" thus depends on what question is being asked, about what aspects of the network, and by whom, requiring an in-depth clarification of the following issues:

- What is the *purpose* of a monitoring task (e.g., SLA enforcement, intrusion detection, traffic profiling, accounting and billing, etc.)?
- Which *specific traffic information* is required to accomplish that purpose? This can be either information explicitly present in the observed packets (e.g., header fields, payload sections) or derived from one or more analysis functions performed on the observed packets (e.g. flow size distributions). Information considered strictly necessary must have the minimum possible information content for the purpose. For example, raw IP address information from the packet header is not strictly necessary if the given purpose could use IP address information aggregated by network (as is often the case for traffic engineering purposes) or anonymised IP address information (as in any task not requiring any form of host attribution) without loss of functionality.
- What *entity* (a system administrator, a division in the operator's organisation, the public research community, etc) runs the monitoring task and has access to its input data as well as the results?

These aspects are in general not static, but may depend on intermediate results of the monitoring application itself, and hence change over time. For instance, a network intrusion

[2] We say "logical" here because each of these first stages may be deployed on the same physical device or even within the same process context on that device. The mapping is not necessarily one-to-one because certain measurement tasks may be able to share identical first-stage processing operations. An analysis of the tradeoffs between resource efficiency in sharing first stages and additional protection risks implied by this arrangement is beyond the scope of this document, but will be addressed in subsequent documents.

detection system residing at the perimeter of a network may not strictly need to access IP le vel information in order to detect attack activity. However, once an attack is detected it may need to precisely attribute the attack activity to a set of source addresses and targets in order to respond to the attack. The dynamic nature of this necessity implies there is another aspect which must be addressed:

- What is the *context* in which the information is collected and processed, and how do variations in context affect the three previous aspects?

A thorough classification of monitoring tasks with respect to these aspects is a daunting undertaking, however a complete ontology of monitoring tasks and subtasks is out of scope of the PRISM project; nevertheless we aim to provide significant contribution by setting the basic structure of such a comprehensive, "proof-of-concept" classification, and demonstrate its effectiveness through selected specific monitoring scenarios and use cases.
In that respect, our proposal consists of ***the specification of a semantic model which captures and integrates all of the nec essary aspects of a monitoring task***. This semantic model not only accomplishes the goal of specifying and formalizing such knowledge, but can be also used as the basic engine for the enforcement and control of the access to traffic data and monitoring primitives and resources.

2.2.2 From semantic to authorisation

The architecture of the PRISM system shall specify an access control system conforming to the specified semantic model. This access control system will guarantee the enforcement of derived policies in line with the security and privacy requirements specified in [PRISM-WP2.1.2].
A naive approach to the design of this access control system, by simply centralizing all access control decisions within a "super-entity" embracing both front-end and back-end would be reductive and unrealistic. Such a system would be undeployable in the real world, taking into account the technical and social separations between administrative domains. Therefore, PRISM adopts a decentralised and distributed approach. In this conte xt and in order to enable the flexibility of the authorisation infrastructure, PRISM exploits the paradigm of the ***Privilege Management Infrastructures*** (PMIs). Indeed, the goal of PRISM is to guarantee that only entities with a given privilege level, depend ent on the specific monitoring purpose and context, may ultimately access some system resources. These resources may be stored data, meta-data, or monitoring subtasks, analysis functions deployed within either the front-end or back-end components. And, whi le some access decisions rely on ad hoc information that is very dynamic in nature and, therefore, must be evaluated in real-time by policy-based mechanisms, there are cases where access depends on, or is affected by, contextual attributes that remain stat ic for a given period (e.g., a role). This information can be evaluated "off-line", and it can be expressed by means of PMI attribute certificates, resulting in a reduction of the effort needed for its "on-line" processing and evaluation.
Privilege manage ment infrastructures are centered on an entity known as the Source of Authority (SoA), which generates and manages authorisation profiles, permissions, and policies. These services are provided offline, such that entities using the infrastructure can independently verify authorisation permissions without the need to interact with the SoA. In other words, PMIs are to authorisation what Public Key Infrastructures (PKIs) are to authentication. PMIs use attribute certificates (ACs) to hold user privileges, in t he form of attributes, instead of public key certificates (PKCs) to hold public keys. PMIs have Sources of Authority (SoAs) and Attribute Authorities (AAs) that issue ACs to users, instead of Root Certification Authorities (Trust Anchor) or Certification A uthorities (CAs) that issue PKCs to users. The SoA is then analogous to the Root CA, which has no run-time role in the operation of the infrastructure.

These considerations suggest the inclusion of a Source of Authority (which we call a ***Privacy-Preserving Controller*** or PPC for short), which provides and manages [3] authorisations through offline "contracts" with the various PRISM system components, for example in the form of X.509 certificates, in conformance with PRISM's semantic model. Since the semantic model is designed in accordance with legal and regulatory requirements, the system will comply with the context of the legal and regulatory environment in which it is deployed.
The complete PRISM access control system will complement this PMI-based model with the run-time evaluation of access requests according to the role of the accessing entity and application, as well as the context in which the access request occurs. The PPC may also be more than an elementary SoA, as it may also be necessary to involve the PPC at runtime in certain exceptional conditions, e.g., when a request to decrypt protected data arrives in the setting of lawful interception.

2.3 Monitoring flow chart

From the viewpoint of the monitoring tasks is performs, an example PRISM instance is composed of the processes shown in the flow chart illustrated in Figure 1. The PRISM capturing device in the FE captures all packets with full payload. In the next step the traffic is isolated according to the goals and the needs of the monitoring purpose. The intention is to identify in the FE the flows relevant to a particular monitoring application, as explained in Section 2.1.1, in order to make the analysis on the data easier. This can, for example, mean just flows the packets of which match a predefined filter string, or flows which's rate exceeds a given threshold. The specific FE analyses for the monitoring purpose are performed in real time mainly on the network processors. After the flows' isolation and extraction analysis the packets are further processed before being delivered to the BE. All the information collected in the FE requires additional protection to reduce risks of privacy violation as described in Section 2.1.2. The encryption key will only be delivered to the BE if a particular flow has to be further analysed in the BE or in an external monitoring application. Depending on the results of the FE analysis (e.g. some threshold was exceeded), the system decides whether further and deeper analysis at the BE is required.
The BE receives the monitoring data and, if necessary, the decryption key(s) (e.g. in an IDS scenario, if the flow is suspicious) and stores them into its database (DB). If further analysis is required and the key was received, the relevant data are taken from the DB and decrypted. If the information can be still reduced to fulfil the monitoring purpose it is reduced to the minimal information needed for the monitoring purpose. Some data may be anonymised and filled with empty or scrambled information to match the format expected by the monitoring application. Finally, the data are handed over to the monitoring application to perform the analysis and to receive the requested results.
Finally it is important to underline that between different monitoring purposes the exact actions for each block may vary and some blocks may not even be necessary (e.g. the flows sent from the FE to the BE in order to be analysed by a specific monitoring application are not encrypted). The behaviour of every block depends, in the end, on who (e.g., a police officer or an ISP employee) is running the particular traffic monitoring application and why (e.g., for tapping or for network performance monitoring).

[3] We recall that, even if off-line, PMIs do not lead to a static (i.e. once for all) authorisation model. As in the case of PKIs, PMI certificates can be revoked and have expiration dates. New authorisation models emerging in scenario changes (such as a change in regulation) can hence be enforced in the system by properly managing the already issued authorisation certificates, and by issuing new certificates or policies conforming to the updated scenario.

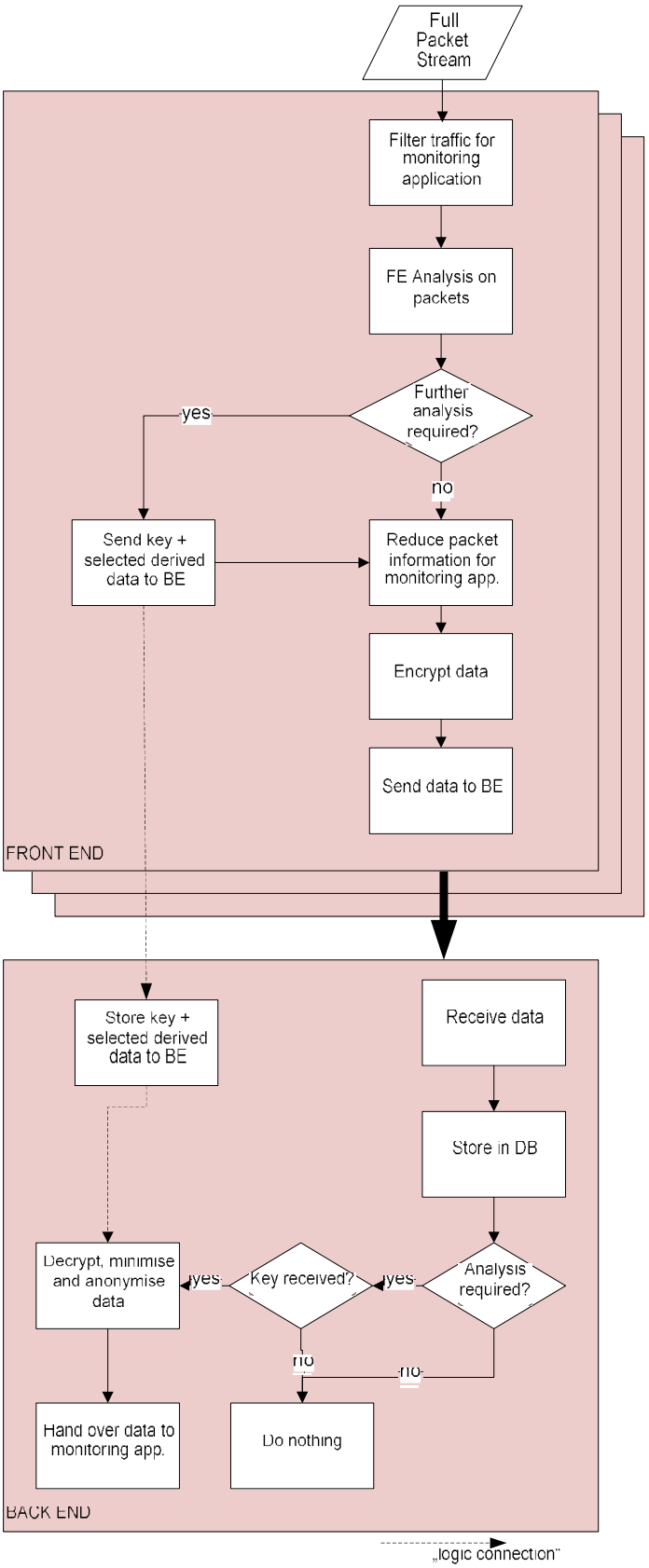

Figure 1: Monitoring flow chart

3 Architecture overview

The realisation of a general-purpose monitoring framework providing strong protection for personal data poses a number of problems that influence the design of the architecture. In this section, we give an overview of the proposed system and try to highlight the important aspects in the chosen architecture.

3.1 Example scenario

Before describing in detail the three main components realizing the PRISM architecture, let us start by introducing a simple scenario (Figure 2) that stresses the main challenges related to the design of such a privacy preserving traffic monitoring framework.
To this end, let us assume that three ISPs share the same Internet Exchange Point (IXP), and that for a particular reason (e.g., data retention, QoS monitoring, DoS detection) all three ISPs need to monitor the traffic departing from and arriving to their specific networks. In this scenario, illustrated in figure below, it makes sense that the traffic is collec ted by the IXP manager on behalf of the ISPs. The IXP manager's task is to deliver to each ISP the proportion of traffic related to the network of the particular ISP. To this end, without yet taking into account the users' privacy, the IXP first has to ide ntify and authenticate each ISP in order to ensure that each ISP requests only the traffic relevant to them. Moreover, also during the traffic monitoring activity the IXP has to pre-classify the data flows accordingly in order to deliver each ISP the corre sponding part of the traffic. Note that within the PRISM system, in line with the IPFIX standard, the term flow is used in a wide sense to mean any kind of information flow driven by the arriving packets. For example, a number of packets observed during a time period of 1 minute constitute a flow in this sense (time series).

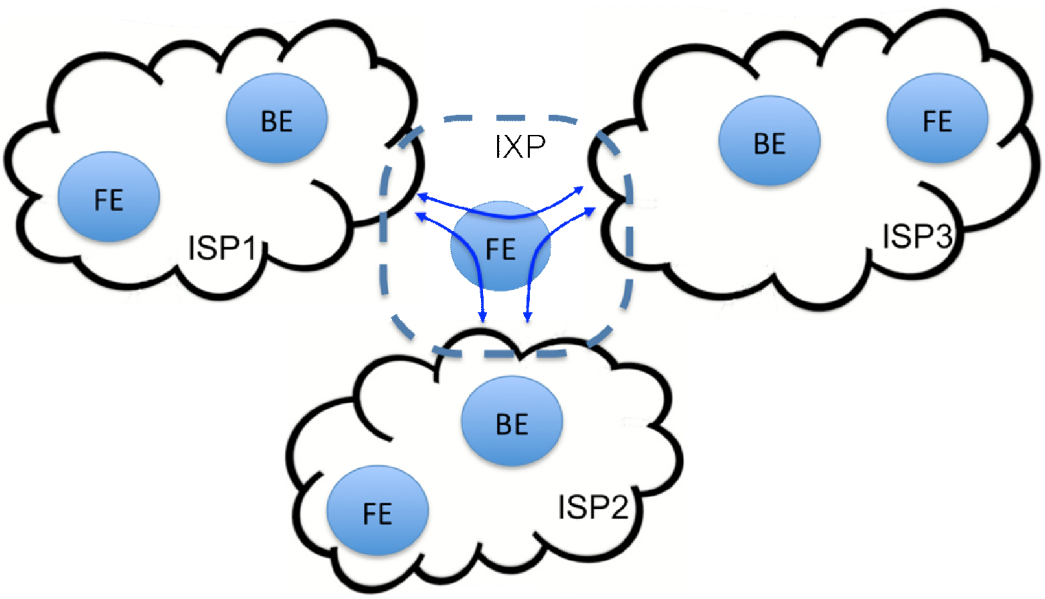

Figure 2: Example scenario

The monitoring framework proposed by the PRISM project is based on a two-stage architecture, which separates the actual monitorin g tasks between i) a front-end (FE), which observes traffic in the network, and ii) a back-end (BE), the task of which is to store and (further) process the results of the monitoring activity. The architecture is complemented by a

third entity, referred to as iii) the Privacy Preserving Controller (PPC), the role of which it is to act as the source of authority (SoA).
In the context of the example scenario, when an employee of an ISP needs to execute a monitoring application that needs to have data collected in the IXP, he or she first needs to authenticate at the BE. After the employee's credentials (i.e., password, X.509 certificates, etc.) have been verified by the BE, the BE then provides its credentials to the FE.
The FE first verifies that the BE is allowed to receive the requested set of data for this particular application. If this is the case, then the FE starts to isolate the flows relevant to this monitoring activity and delivers them to the BE.
Note that the information required depends completely on the particular traffic monitoring application. In particular, the underlying philosophy of the PRISM architecture is that only the information strictly needed to accomplish a specific traffic monitoring task should be provided to the BE by the FE, and nothing else. For example, if the flow data consists of packets, then the corresponding packets relevant to the specific traffic monitoring activity are first processed by appropriate protection mechanisms (e.g., anonymisation and encryption) before transmission to the BE.
Finally, we recall that an entity belonging to the PRISM architecture (i.e., the FE for example) in order to verify the credentials (i.e., an X.509 credential certificate) presented by another entity (i.e., the BE for example) or by an employee of some organisation (if needed) first requires a cryptographic verification of the signature of the credential certificate and then a verification (based on the policies set forth by the PPC) that the presented credentials are suitable or sufficient to be authorised to perform a specific operation.

3.2 Components of the PRISM architecture (FE, BE, PPC)

As already mentioned, the PRISM architecture consists of three components, the front-end (FE), the back-end (BE), and the privacy preserving controller (PPC).
The FE provides the basic functionalities necessary to capture, isolate, and protect the flows. In the FE the traffic captured by a Capturing Unit is processed in order to both i) filter the flows interesting to a specific monitoring application, and ii) to protect the flows in order to disclose to the monitoring application only the data it strictly needs to accomplish the task. In other words, the data filtering process at the FE from one hand ensures protection of the user's privacy by preventing an application to analyze all the traffic collected in the network. On the other hand, the filtering process can also greatly reduce the amount of data the monitoring application has to analyze to complete the task, thus improving the performance and scalability of the monitoring application itself.
The BE constitutes the system's entity that mediates between the information sources (i.e., the FEs) and the information consumers (i.e., the monitoring applications), controlling the access of the latter to the traffic data sent by the former. In that respect, the BE provides an interface for the monitoring applications, which can be seen as the "end-users" of the information collected, while it provides the corresponding mechanisms for the authentication and authorisation of the applications and their users. These mechanisms include the verification of identities, roles, credentials, and access rights. On the other hand, the BE has to interact with the FE in order to initialise and manage the data capturing activities.
Note that even if a pilot implementation of the PRISM architecture is targeted for deployment at an SME (i.e., where both FE and BE are managed by a single entity), this is not the case in general. Instead, there clearly are situations where these two entities are managed by different authorities, as already indicated in the example scenario of three ISPs sharing a single IXP. One can further anticipate a scenario where more than one BE has to interact with more than one FE. The PRISM architecture will be able to cope with this kind of scenario easily. For example, the FE should be seen, to some degree, as an element that can be triggered by any

entity which is in possession of the appropriate credentials. These credentials or authorisations as well as the corresponding policies are provided by the third element of the PRISM architecture referred to as Privacy Preserving Controller (PPC). In daily operations, the PPC has a relatively passive role. The PPC acts as the source of policies and creden tials for both the FE and the BE, which, by some means, must obtain them from the PPC before any other operation can take place. Both the credentials and the policies issued by the PPC should take into account commercial agreements (e.g., between the ISPs and the IXP in the example scenario), the current legislation, and the internal decisions of the ISP .

3.3 BE-FE interaction and characterisation of the roles

The interaction between BE and FE consists of control and data channels as shown on Figure 3. The control channel is used by the BE to provide the FE with credentials, information about the monitoring application requesting the data, etc. The data channel, on the other hand, is used to exchange the actual data resulting from t he monitoring. As already stated, the data captured by the FE are filtered and protected according the specific monitoring applications' needs.

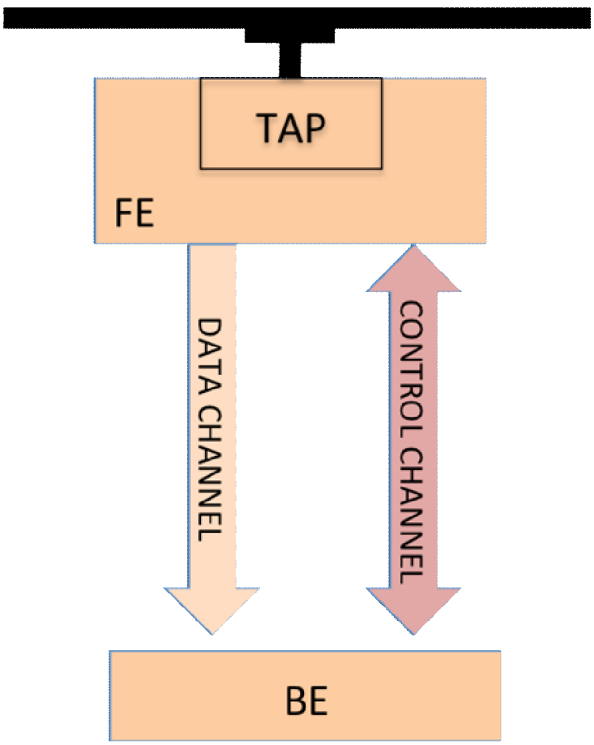

Figure 3: Interaction channels of FE and BE

It should be emphasised that the use of certificate based identity and permission management between the FE and BE allows the PRISM architecture to be employed also in very large scale network monitoring scenarios (e.g., a large ISP), where several BEs and FEs could be deployed. In general, each BE is provided with credentials or certificates that allow it to obtain specific type of flow or flows with a specific level of protection (i.e. encrypted and/or anonymised) from the specific FE, while at the same time, the roles of the persons using the particular applications is taken into account by the given BE.

From the PRISM requirements the data channel from the FE to the BE is a unidirectional IPFIX session, secured by TLS. While IPFIX can be used to export raw flow data from the FE to the BE, raw packet data using PSAMP, or unprotected summary data via flexible flow keys, we do not anticipate this will often be the case, given data protection requirements. Instead, the flexible data model provided by IPFIX will be used to export processed and prot ected data, while inlining metadata (e.g. anonymisation metadata, or secret -sharing key fragments required by the cryptosystem in use by a given analysis). Note that the use of TLS implies that X.509 certificates will be used for transport security identit ies, which allows the integration of identity management between the transport layer and access control system. More details on this interaction appear in Section 7.

The control channel between the FE and the BE may also be used by external components to control the behaviour of the FE. This allows the usage within an existing monitoring framework (e.g., MINER) to allow for an easy integration in operational networks. The monitoring framework has to interact with the PPC in order to receive the appropriate credentials to control the FE.

The task of the FE is to collect the data required by monitoring applications, which may include also some elementary operations needed to properly filter or elaborate the flows of interest (i.e., the information strictly nee ded by the monitoring application task).

However, the FE is not supposed to keep track of the "history" of the data flows, but instead one should consider that the operations carried out by the FE have to be fast, implementable also in hardware (for fast r eal-time operation), and above all, such FE operations should not be based on the packets previously collected belonging to the same (or some other) flows. This implies, e.g., that the FE has no access to the stored traces, and that all the operations are triggered by arriving packets. Packet encryption is performed per packet. The element capable of analyzing the overall situation based on more complicated and extensive information, in off-line fashion if needed, is the BE (as well as the traffic monitorin g applications operating through the BE). Note that while the FE is in possession of the full payload (including privacy sensitive parts therein), the (final) analysis carried out at the BE is limited by the protection mechanisms applied to the packets by the FE.

.

3.4 PRISM ontology

As described in the previous chapters, the need for the semantic representation of several concepts underlying the PRISM framework has been clearly identified. These concepts include the particular data types that are collected and processed, the purpose behind the targeted analysis, the roles of the entities involved in the collection and processing chain and any other information that determines the context of a monitoring application's execution. The PRISM system's behaviour will strongly depend on these parameters and, therefore, all these concepts along with the rules that regulate access and specify additional behavioural norms are integrated in a semantic model, implemented by means of an OWL ontology [OWL].

The different concepts that comprise the PRISM knowledge base, and thus they affect PRISM operation and constitute the classes of the PRISM Ontology, include:

1. Types of personal data, both native, e.g., header fields, and derived after processing.
2. Types of monitoring service s, representing monitoring tasks and purposes.
3. Types of roles that characterise the different actors.
4. Contextual information, such as the specific conditions that characterise the specific execution of an application, the history of data retrieval, etc.
5. PRISM system's components that correspond to analysis functions.
6. Workflows that are executed for the transformation of a data type to another.

7. Rules that define access rights and complementary actions that are performed in association with some access to data and depend on all the parameters above.

The PRISM Ontology serves for the expression of regulatory provisions into concrete rules and its scope in terms of feeding the system wide decision engines. One thing that is worth to be noted here is that the use of ontology has been preferred over a legacy policy language, such as OASIS XACML [XACML], because of its expressive power. The use of an ontology enables the specification of complex concepts, data types, hierarchies, relations, etc., while providing significant advantages in terms of rules consistency evaluation, reasoning capabilities, as well as integration with other semantic models .

4 Front - end

This section describes a high level architecture of the front -end component, which is responsible for observing the traffic in a link in real -time.
In order to fulfil its role in the PRISM architecture, the front -end block must provide some basic functionalities, performed by several devices that communicate, e.g., through a LAN network which is not physicall y accessible from the outer network. Note that the front -end is, by definition, directly attached to the observed link, and thus it must be protected physically the same way as the corresponding link.

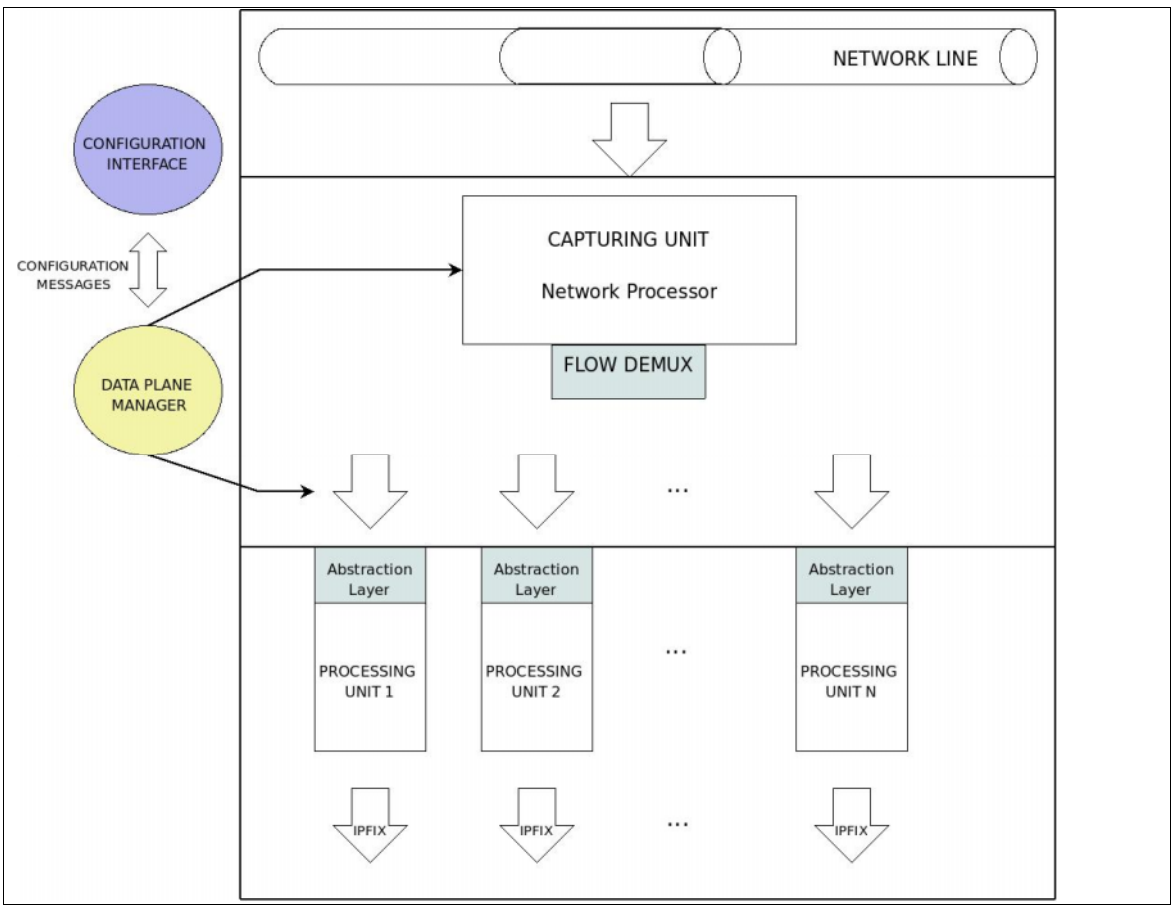

Figure 4: Front-end architecture

As shown in Figure 4, the FE consists of several types of functional blocks:

- ***Capturing Unit*** (CU) can be implemented to consist of one or several standard PCs equipped with a board hosting a network processor connected to t he local PCI bus. Such a board is typically be equipped with up to three Gigabit Ethernet interfaces, and are capable of capturing traffic flowing over a gigabit link with precise timestamps for each packet. Moreover, such boards can classify each packet b ased on a rule set according to the canonical 5 -tuple. Once the capturing phase is performed, the classified flows are demultiplexed to a cluster of processing units. A classification rule is associated with one or more processing units and flows are sent to the selected devices. To execute these operations, the CU must support on -line configuration of

such a demultiplexing table, upon interaction with the FE internal control plane. For the demultiplexing data can be sent to different processing units (PU, see below), e.g. by sending to different MAC addresses; then the actual physical demultiplexing task is performed directly by the Ethernet NICs.

- ***Data Plane Manager*** (DPM) is a logic unit responsible for communicating, by means of proprietary protocols, which flows have to be sent to which processing unit and what kind of processing has to be carried out by each of them.
- ***Cluster Processing Unit*** (PU) are in charge of actually implementing the data protection techniques. Since this kind of data protection strictly depends on the application requiring the traffic traces, different tasks must be performed concurrently within these blocks. Since different PUs can be considered as being totally independent, each of them has to implement its own stack of protocols to communicate with the BE. In addition, PUs must support communication with the internal FE control plane in order to receive the configuration messages.
- *Configuration Interface* (CI) is in charge of receiving the configuration transactions from the BE and of authenticating and authorizing them. Once a configuration transaction has been authorised, the task of actually enforcing it is performed by the DPM .

Such devices communicate the data related to different traffic flows captured on the monitored link.

4.1 Front – end control interface

Since packet capturing and first stage of data analysis have to be configured on-demand, the FE block must be provided with a control interface for on-line configuration
A typical configuration transaction will involve a Back-End Monitoring Agent (BE-MA) issuing to the FE interface a request for accessing the output of a specific FE analysis function (possibly performed over specific traffic flows).
The FE will therefore decide what kind of sensitive data it is allowed to disclose and, hence, apply a proper data protection policy. Such a decision will be based on the authorisation certificate (issued by the PPC or by a proper delegated authority) that will be provided by the BE-MA.
Several solutions for specifying permissions are available (X.509 RBAC, SAML); the choice of the proper protocol will be then considered in a later phase.
In order to allow for a precise description, we will use a conceptual model of processing functions of the FE. In particular, we assume that the FE supports a finite set of analyses $\alpha(\varphi,\pi)$, where φ is a vector of traffic flows and π is a vector of application-dependent configuration parameters (such as packet snapshot length).
Subsequently, each authorisation certificate will specify:

- a set of analysis tasks that can be set up,
- an (optional) range of values that each parameter of each analysis can assume (e.g. in some cases the analysis can be allowed to process the whole packet payload, while in other cases its scope can be limited to the header section),
- an (optional) specification of the traffic flows which the authorisation is limited to.

After the interface has received an authorisation certificate, the FE interface has to check the identities of both the BE-MA and the authority which generated the authorisation; this can be easily achieved by using Public Key Certificates (PKC) generated by a trusted authority.

Authorisation certificates can also have an expiration time; when such a time limit is reached, the authorised analysis function is torn down.
After the identities of the parties involved in the transaction have been verified, the complete specification of the analysis to be set up is sent from the BE to the FE control interface. In particular, such a specification will include:

- identifier of the specific analysis function which has to be set up
- actual values of the configuration parameters in π
- specification of the flows which will be processed by the requested analysis
- address of the host which will receive the processed data

Naturally, all of these specifications must be checked against the authorisation certificate which has been submitted in the previous phase. After that, the interface will convey the configuration data to the DPM, which, in turn, will issue the proper configuration messages to the different components of the FE block. A connection will be established directly between one of the processing units of the FE and the selected host in the BE, and the output of the analysis function will flow through such a channel.
Limited on-demand reconfigurability of the FE analysis functions can be provided by allowing the BE-MA to update the configuration parameters in π while an analysis is running; in this case, the BE-MA will issue an "update" message without repeating the whole configuration transaction. Therefore the interface will only have to verify the identity of the issuer and to check that the new parameters are conforming to the authorisation certificate. In order to speed up such a transaction (which can be part of a looped control cycle of a monitoring application), the FE may cache both the identity and authorisation certificates which were retrieved during the configuration phase.
In addition, the FE interface has to check the expiration date of its cached certificates and, in case of expiration, to stop the associated analysis.

4.2 Data plane manager (DPM)

The data plane manager is in charge of configuring and coordinating the devices involved in the captured data processing. In particular, after a request for setting up a new analysis is validated by the FE control interface, the DPM:

- selects the processing units in charge of performing the new processing task,
- sends appropriate configuration messages to such a processing unit, including the flow to be processed and the values of the analysis operational parameters,
- sends appropriate configuration messages to the capturing unit, including the description of the flow to be monitored and the address of the chosen processing unit.

In particular, the DPM is responsible for configuring pre-filtering within the FE capturing unit: such a function performs a preliminary selection of the packets belonging to a given flow based only on the canonical 5-tuple. A more refined filtering process, which will select the packets belonging to a flow based on arbitrary rules (e.g. packet length, presence of a pattern) will be performed by the PUs. Notice that such a classification activity can turn out to be very processing-intensive and can raise very challenging performance issues. As an example, let us assume that a BE analysis requests a flow composed of all the packets containing a given string: such a request involves a PU performing deep packet inspection at gigabit speed. Although this is feasible with a proper hardware platform, it has to be carefully taken into account during the FE design.
The DPM is also responsible for keeping updated the state information concerning the association of each analysis to a selected Processing Unit and for maintaining a coherent

version of the capturing unit classification table. Based on such information, a new analysis can be allocated, according to performance optimisation criteria and already set-up analyses can be torn down (e.g. in case of an authorisation revocation) or updated with new parameters.

4.3 Capturing unit

The capturing unit is essentially made up of one or several PCs equipped with a board hosting a network processor through the local PCI bus.
The board can be equipped with up to three Gigabit Ethernet interfaces, which can be used both to capture traffic and to deliver data to the processing units

4.3.1 NP board

The software running on the NP board must capture the incoming traffic and either classify it into flows or, if no match with any classification rules is found, discard them.
Notice that at this stage flow classification is based only on the value of the canonical 5-tuple (IP addresses, ports, protocol). Classification based on other values has to be performed by the PUs. In the definition of a flow, wildcard values are allowed, thus guaranteeing a fair degree of flexibility.
If the traffic rate is so high that a single probe turns out to be insufficient, the whole incoming stream can be mirrored to several NP boards in charge of running non overlapping sets of classification rules so as to balance the overall load among different units.
Packets belonging to the same flows will be compressed and placed into batch frames, which will be delivered to the associated PUs.
A batch frame consists of a collection of data structures whose fields may include:

- *packet snapshot*
  (e.g. the first n bytes - a different value n can be defined for each flow),
- *flow id,*
- *timestamp.*

Flow demultiplexing is performed at the MAC level through the association among batch frames and the MAC addresses of the network cards of the PUs that will be in charge of processing packets of that flow.
In addition, the NP keeps track of some cumulative statistics for each flow (basically packets and bytes count).
The NP board communicates with the host CPU through the PCI bus. The classification table used by the NP is computed and updated by the host CPU, which loads it into the internal memory of the NP board. Optionally, the CPU can periodically read the flow statistics collected by the network processor.

4.3.2 Host PCs

The host PC constitutes the interface between the capturing unit and the FE control plane. It is in charge of creating and updating the classification table and to write it to the NP memory banks. For each flow, such an application receives from the control plane the following information:

- definition of the flow (in terms of range of values in the domain of the 5-tuple),
- corresponding flow id,
- length of the packet digest to be included in the batch frames,
- one or more (layer 2) destination addresses (packets belonging to the same flow can be included into multiple batch frames).

Optionally, the host PC can provide the PUs with cumulative per-flow measurements, read directly from the NP board. Although, from the functional point of view, extracting per-flow measurements looks more suitable for a PU, executing it directly on the capturing device can be very profitable in terms of performance.

4.4 Processing units

Processing units consist of a cluster of possibly heterogeneous devices (commodity PCs, NPs) that actually run the most complex anonymisation and preprocessing algorithms before delivering the resulting data to the PRISM back-end. In particular, the PUs are in charge of executing the application specific FE analysis functions, which are triggered and configured by proper configuration messages issued by the DPM. Each processing unit can execute one or more FE analyses, depending on performance issues.
The network card(s) of such devices will receive the batch frames sent by the capturing devices through a proprietary protocol over Ethernet.
The use of such a communication paradigm requires the implementation over each unit of a software compatibility abstraction layer in order to allow running analyses to actually retrieve packets through standard data interfaces. Furthermore, the abstraction layer limits the packets processed by a FE analysis to those selected by the CU, thus guaranteeing both a performance improvement (fewer packets to be processed) and information segregation (the FE analysis cannot access packets belonging to other flows).
A daemon-like process runs on each processing unit and is in charge of handling the configuration transactions issued by the DPM and to manage the FE analyses accordingly. In particular, such a daemon will take care of starting up a requested analysis and updating the parameters of an already running one.
The isolation of such a flow is performed by the PU itself if the flows which an FE analysis has to process are defined over a domain which encompasses parameters other than the classical 5-tuple.
The processing units are also responsible for communicating with the BE agent: a connection will be established directly between one of the processing units of the FE and a selected host in the BE. According to the IPFIX protocol, the processing unit will initiate the connection and will transmit an IPFIX template message in order to describe the format of the processed data. IPFIX considers a flow to be any number of packets observed in a specific timeslot and sharing a number of defined properties. The FE is free to use user-defined data types in its messages: the protocol is indeed freely extensible and the used fields can be adapted to different needs.
Since the whole data flow has to be encrypted, a secure communication protocol has to be implemented by the two end-points of the connection. A broadly diffused protocol that allows a secure connection is the IPSec suite, which represents the standard for ensuring a secure way for data communication at the network layer. By providing the security services at network layer, IPSec is more flexible than TLS, which depends on the transport layer protocols.
Mainly, IPSec defines two distinct protocols, which can also be used together: Authentication Header (AH) that provides data authentication and Encapsulating Security Payload (ESP) that basically provide data encryption and can optionally provide data authentication. Moreover, we suggest the tunnel mode of IPSec that encrypts both the header and the payload.

4.5 Internal communication network

Since the different units of the FE block have to exchange both configuration messages and data to be processed, both a control plane network and a data transfer network are needed;

since they have different functionalities and requirements, the two networks are logically isolated, although they can share the same physical layer.
The data plane network has to convey the captured data to the proper processing units and is likely crossed by high bit rate traffic. Since all the units of the FE are assumed to be physically adjacent, correct demultiplexing of captured data can be provided by the layer-2 forwarding mechanisms, thus avoiding the processing and bandwidth overhead related to upper layer protocols. Data plane messages can be Ethernet frames (jumbo frames have to be supported in order to allow the transport of maximum size captured packets) containing captured data formatted according to a proprietary protocol. Since performance is the main concern in the data plane, such a protocol must imply a low overhead (small headers and no state information). In addition, since it has to be handled by network processors, which cannot rely on existing libraries and have a limited programming flexibility, such a protocol must be easily implemented and parsed (few fixed size fields and no complicated options).
On the other hand, the control plane network has no particular performance concerns (since it basically operates in a slow data-path) and implementation concerns (since it is accessed by standard PC-like devices), and therefore it can be designed foremost with flexibility and reliability of communications in mind. To this end, several solutions appear to be suitable (HTTP, XML messages); the choice of the specific protocol is an implementation-time determination.
Notice that, for the two networks, no security and authentication issues have been taken into account due to the following assumptions:

- all units of a single FE reside in the same physical location and administrative domain, which is not accessible by untrusted entities,
- only communication points with the external network are the control interfaces (which are designed for authentication and security) and the data interfaces of the processing units (which only push IPFIX reports to a set of collectors at the BE).

5 Back- end and privacy-preserving controller

The PRISM Back-End (BE) provides a storage for the measured data and carries out both data analysis and access control computations. Privacy Preserving Controller (PPC), on the other hand, is the ultimate source of authority providing, among other things, the semantic rules for the system, as well as, the (root) cryptographic keys. Together they constitute a policy-based access control system. Note that term "access" does not mean "access to data", but rather access to execute an analysis. In essence, when an actor running a monitoring application submits a request to the PRISM system, it triggers the execution of a sequence of analysis functions; this sequence is dynamically created depending on the "privacy context". That is, the actor is granted access to the execution of the sequence of analysis functions and, consequently, to the resulting data sets. This is something that essentially differentiates the PRISM access control model from the other privacy-aware access control models that exist in the literature and has its roots in the fact that, in contrast to the other approaches, PRISM uses raw network traffic as first material instead of well-defined data structures.
Similarly to other policy-based systems, in the context of this joint BE – PPC operation, the access control functionality follows the Policy Decision Point / Policy Enforcement Point (PDP/PEP) abstract model [RFC3198], with the BE clearly playing the role of access control enforcement point. Since the analysis functions executed at the Front-End (FE) are transparent to the BE, what the PDP provides is the definition of the BE's functional behaviour, i.e., the sequence of analysis functions that need to be executed in the context of a request. This BE Analysis Functions Sequence (BAFS) is enforced by the PEP.
In that respect, several things are clear:

- the PEP resides at the BE;
- the underlying policies are specified in and provided by a semantic model, i.e., the PRISM Ontology (Section 6);
- the system's architectural entity in charge of the definition and lifecycle management of the PRISM Ontology is the PPC.

An issue that possibly needs further investigation is where the access control reasoning takes place, that is, where the "privacy context" is evaluated following the rules defined in the PRISM Ontology in order for the BAFS to be specified on the dynamic, ad hoc basis that PRISM puts in place. In other words, where the PDP resides remains an open issue.
There are three possible approaches:

a) ***PDP is hosted by the BE***: under this scenario, the PPC disseminates the PRISM Ontology to the corresponding associated BE(s) and every time a request reaches a BE, the BE reasons taking into account the "privacy context" and takes the necessary decisions regarding the specification of the BAFS. Immediately after, the BE proceeds with the enforcement (execution) of this BAFS.

b) ***PDP is hosted by the PPC and the BE "asks" the PPC about the BAFS:*** in this case, when a request reaches the BE, the BE provides the PPC with the privacy context (i.e., a set of attributes {data, purpose, role, conditions, …}) and the PDP functionality of the PPC determines the BAFS on this basis. This BAFS is communicated to the BE in order to be enforced.

c) ***PDP is hosted by the PPC which issues ACs specifying the BAFS***: this case is similar to the previous one, with the fundamental difference that after the reasoning, the PPC issues an X.509 Attribute Certificate (AC) assigned to requesting entity, specifying the BAFS in the AC. Evidently, this scenario presupposes that the requests are submitted not only to the BE but to the PPC as well.

Among the three approaches described above, the more reasonable and convenient one is the first approach (a). The second approach (b) has the significant disadvantage that it requires communication and functional interaction between the BE and the PPC on a continuous basis. Consequently, it implies a need for a specification of a proprietary BE $\leftrightarrow$ PPC protocol, as well as, an introduction of communication delays to the overall operation.
Similarly, the third approach (c) suffers from the fact that the requests must be first submitted to the PPC for issuing an AC, and then this AC must be submitted to the BE. Additionally, the management of the ACs becomes a very complex and onerous task, since the ACs must be issued for every request, be used only once for the given request, and be immediately revoked after having been used. As a matter of fact, the burden imposed by the ACs management has been frequently cited in the literature (e.g., [CHADWICK2003], [KNIGHT2002]), while the dynamic, "privacy context"-based nature of access permissions ("privileges", following the terminology of [ITU-T2005]) of PRISM makes the situation even harder.
Another important point that should be noted here is that, in both approaches (b) and (c), there is an implicit need for state maintenance in the context of every monitoring application execution. This is because every request is state-dependant with respect to the previous requests that took place during the life cycle of a monitoring application execution.
Therefore, in what follows, the description provided adopts the first approach, that is, the coexistence of the PDP and PEP at the BE. However, only a few changes would be needed in the case of the adoption of any of the other two approaches.

5.1 Roles management and authorisation

Each actor that participates in the PRISM operation is assigned with a role. As explained in Section 6, each role has its semantic definition in the PRISM Ontology, while the different roles are organised hierarchically. That is, one aspect of the PRISM access control model is that it constitutes an Hierarchical Role-Based Access Control model (H-RBAC) [FERRAIOLO2001], actually a very extended one.
The assignments of roles to actors are hold by Role Assignment ACs (RAACs) [ITU-T2005]. In that respect, a fundamental responsibility of the PPC is to play the role of the Source of Authority (SOA) in the PRISM Privilege Management Infrastructure. The PPC is ultimately the system entity responsible for issuing ACs to trusted holders, which can be either actors or subordinate Attribute Authorities (AAs), such as associated entities at each department of the operator so that distributed and hierarchical roles' management to be possible. In this context, PRISM implements a Privilege Management Infrastructure.
When an actor participates in a request, the actor provides the BE with the RAAC holding the corresponding {actor, role} assignment. The BE (i.e., its PDP functionality) takes into account the nominated role as part of the "privacy context", in order to specify the BAFS. However, as already has been analyzed, the BE and FE implementation design sets no explicit restriction regarding the number of entities being in charge of these two components. Indeed, in general the BE and FE components are managed by two different authorities. Hence, the flexibility of the PRISM architecture implies for a preliminary authorisation procedure, this time concerning the BE's interoperation and collaboration with the FE. Consequently, when an actor introduces himself to the BE through his RAAC, the BE will not be capable of serving that actor until the FE co-operator recognizes it and acknowledges its credibility.
At this preliminary authorisation proceedings the BE is responsible for forwarding to the FE a different RAAC which encompasses a respective {role, purpose} assignment. This RAAC denotes at the FE that:

1. The BE collaborator is a valid one and hence trustworthy.
2. The specific indicated actor representing a specific role participates in a request in order for a specific purpose to be fulfilled. That should lead to a first level of FE

configuration, according to the purpose which is to be processed and concluded.

It is in the second stage of the authorisation procedure that the BE reviews and evaluates the corresponding {actor, role} pair assignment to specify its prospective behaviour

5.2 Back-end internal architecture

Figure 5 provides a high-level view of the BE internal architecture.

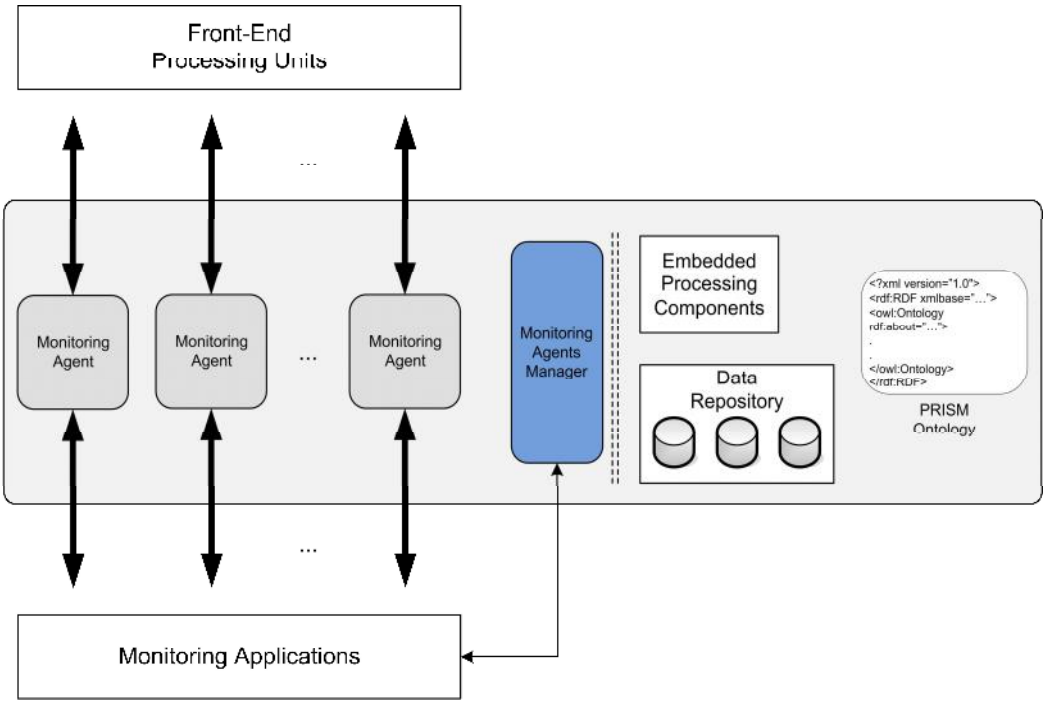

Figure 5: Back-end architecture

As illustrated in the figure, the fundamental components of the BE are the Monitoring Agents (MAs). A MA comprises a PDP/PEP pair and is in charge of mediating between a monitoring application and the Processing Unit of the FE that provides the data structures that are subject of the different BAFSs that are executed in the context of a monitoring application execution. That is, a MA is the peer entity of a FE Processing Unit on the one hand and of the monitoring application on the other. It is a stateful component that is initialised when the execution of a monitoring application begins. For the effective management of the data it receives by the FE, each MA uses a data buffer, based on a relational database and enabled to handle the incoming data as a stream.

It should be also noted that each MA of the BE constitutes an autonomous entity, kind of an independent BE instance. After its initialisation and until the termination of the associated monitoring application, each MA is acting autonomously to what concerns the communication with its peer entities in terms of exchanging control messages and data, as well as, the PPC. Therefore, each MA implements the full communication stacks. Additionally, each MA logs during its lifetime all the actions it performs and any data disclosure, along with the associated metadata that reflect the requesting entities, the context, etc.

The Monitoring Agents Manager (MAM) is the BE component that manages the pool of available MAs. In the beginning of a monitoring application's lifecycle, the monitoring application contacts the MAM once in order to establish an association with the MA that will

constitute the serving MA for this application. The MAM initialises an idle MA from the pool and from this point onwards the MA acts autonomously.
All the MAs comprising the BE share three different types of components:

a) Data Repository, which implements the data storage infrastructure. It is noted that the Data Repository constitutes in essence a separate layer of the system; in that respect it can be distributed and it may be shared not only by the MAs comprising the BE, but also by other BE instances.

b) PRISM Ontology, that is, the semantic model of the system. The PRISM Ontology is encapsulated by a software component which:

   i. provides an API to the MAs in order for the latter to retrieve the necessary information;

   ii. is being contacted by the PPC when an updated version of the PRISM Ontology must be installed at the BE.

c) Embedded Processing Components, which constitute a library of software tools. More specifically, this library offers to the MAs:

   i. Tools for analyzing the incoming IPFIX data structures, as well as for creating the outgoing ones.

   ii. Tools for data encryption and decryption.

   iii. Tools for verifying the RAACs.

   iv. Rich library of data anonymisation functions, such as the ones described in [KOUKIS2006].

   v. Tools for performing transformations between different types of data.

   vi. More advanced transformation tools, such as data aggregation.

   vii. Means for execution tasks imposed by the legislation, such as the notification of the authority and the information of the data subject.

   viii. Components for the internal execution of several parts of the monitoring applications' logic, i.e., parts that are characterised as privacy-sensitive.

Note also that the embedded processing components have their semantic representations in the PRISM ontology, as described in Section 6.

5.3 Privacy-preserving controller

The privacy preserving controller (PPC) holds multiple roles of administrative nature:

- It is the Source of Authority (SoA) of the PRISM PMI.
- It incorporates the PRISM Access Control Model Editor, a user-friendly software tool devised for the specification and management of the semantic model. This tool is not absolutely necessary, meaning that since the access control model is based on an OWL ontology, any corresponding platform (such as Protégé [PROTÉGÉ]) can be used for the specification and management of the model. However, the use of a user-friendly editor hides the technical details of the underlying model, requiring no particular technical expertise by its users.
- It is a monitoring application itself, in the sense that it constitutes the entry point for the authorities for the execution of law enforcement tasks.

- It maintains the (root) crypto secrets which are necessary to retrieve the per-flow encryption keys when not provided through other means (e.g. escrow approaches); this for instance occurs under specific conditions such as legal action/inspection, etc.

It is also worth mentioning that as the role of the PPC is administrative, it is not typically involved in real-time operation. In fact, in a small scale application the minimal set of PPC tasks can be fulfilled by manually configuring, e.g., static cryptographic keys to both FE and BE. Such a system has the obvious drawbacks including tedious and error-prone configuration, as well as, potential security risks.

6 Semantic model

In order for the access control framework to duly include the regulatory provisions, our approach is to use a semantic information model that associates personal data, services and roles with explicitly defined access rules. In that respect, the approach taken is to express any related information by means of an ontology, which is implemented using the W3C Web Ontology Language (OWL) [OWL].
The vision is that the ontology should be as detailed as possible in terms of the various types of personal data monitoring services and roles, so that the widest range of services and situations when personal data are involved can be covered. This is similar to what the Common Procurement Vocabulary (CPV) [CPV] represents for public procurement in Europe; it provides an exhaustive –almost semantic– list of several thousands of products that can constitute subject of public procurement.

6.1 Description of PRISM ontology

In order to associate the personal data with specific processing tasks, the identification of the particular type of each data item is necessary. Moreover, in order to define the appropriate rules that will regulate the disclosure or processing of a personal data item with respect to the purpose for which the information is requested by the data processor, a similar taxonomy of the monitoring-related services must be present. These taxonomies constitute separate subgraphs of the ontology. Therefore, the ontology provides a detailed vocabulary of data types and services' types, structured in an hierarchical way with well defined inheritance rules, which enables the system to associate all privacy related decisions to semantically specified notions. An equivalent taxonomy is needed for the roles of the involved actors and therefore a corresponding roles' subgraph is defined.
These three subgraphs constitute the base of the semantic model, being the domain for the specification of the rules, which comprise another subgraph. However, several additional concepts are modelled by the PRISM Ontology, such as the software tools included in the libraries of the Back-End.
In the following, the description of the ontological classes is provided. It should be noted that this is a characteristic, descriptive but yet preliminary version of the PRISM Ontology.

6.1.1 `PersonalData` class

The types of data that PRISM deals with are defined as instances of the `PersonalData` OWL class. Inheritance hierarchies, as well as other relationships between data are defined using OWL properties. The instances of this class are organised using three hierarchies:

i. The first hierarchy specifies the inheritance of characteristics, referring to the rules that regulate the collection and processing of data. The "root" data type of the subgraph is the `AllPersonalData` type, from which all the other types inherit, while "first level" descendants of the `AllPersonalData` type include `PersonalInformation`, `LocationData`, `BillingData`, `CommunicationData`, etc. These types constitute general data types, in essence categories of data types. This hierarchy is implemented by means of the `inheritsFromData` object OWL property.

ii. The second hierarchy defined inside the `PersonalData` class deals with the detail level of data types. For this purpose, two properties are defined, `lessDetailedThan` and `moreDetailedThan`, being the one inverse to the other. In that respect, the `IPv4SourceAddress` personal data type is

`moreDetailedThan` the `IPv4SourceAddress1stOctet`, while with respect to location data (which according to the legislation are subject to monitoring activities, e.g., [2006/24/EC]) the `Country` data type is `lessDetailedThan` the `GPRSCellID`.

iii. The last relationship between the instances of the `PersonalData` class is the one that defines complex types resulting from simpler ones. In that respect, the `IPv4ProtocolHeader` of an IP datagram contains the `IPv4TTL`, `IPv4DestinationAddress`, `IPv4SourceAddress`, etc. data types. The corresponding relationships are implemented by means of the `containsType` and its inverse `isContainedToType` OWL properties, which in essence define an AND-tree hierarchy.

It should be noted here that the `PersonalData` class of the PRISM ontology is not meant to only include data types that are explicitly contained in the monitoring flows or are part of the monitoring procedure (e.g., date/time of the monitoring, location data, etc.); data types derived from processing procedures (e.g., information related to billing) and information otherwise associated with the data subjects (e.g., personal information of the operator's subscribers) are considered for inclusion in the model.
The following Figure 6 illustrates part of the personal data subgraph, along with the OWL properties that implement the three types of relationships between the personal data instances, as described above. The following figure is provided as a explanatory preliminary snap shot of the `PersonalData` class of the PRISM ontology.

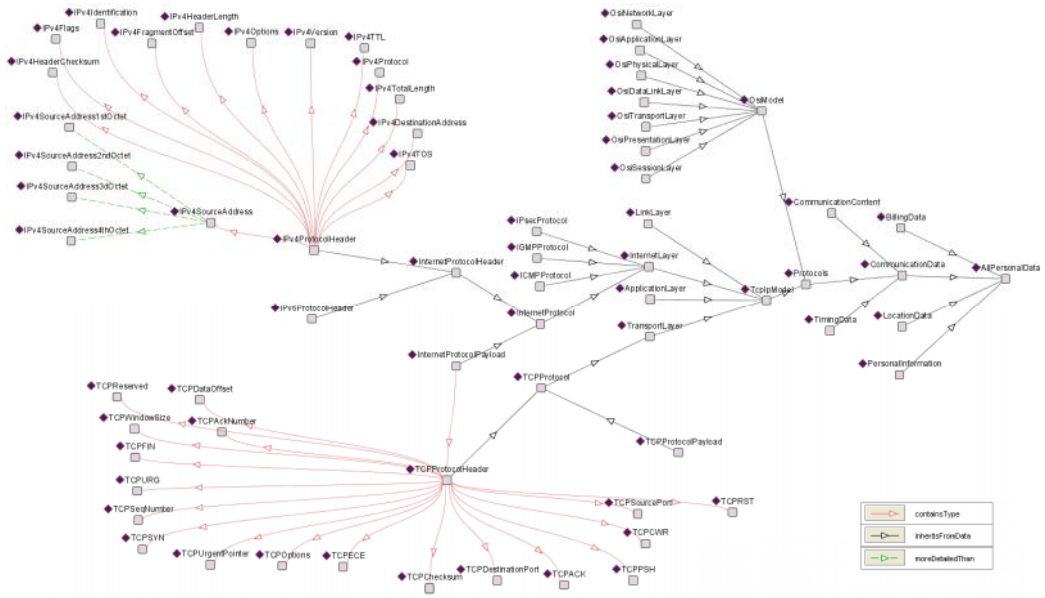

Figure 6: PRISM Ontology – Personal Data Subgraph

6.1.2 `Services` class

The different monitoring services' types are organised as an hierarchy that defines inheritance of characteristics. All the defined types constitute instances of the `Services` OWL class. The "root" service type is the `AllServices` type, from which all the other services' types inherit, while "first level" children of `AllServices` type instance include `PerformanceMonitoring`, `LawEnforcement`, `Billing`,

TrafficClassification, etc. These types constitute general monitoring services' types. This hierarchy is implemented by means of the inheritsFromService object OWL property. It is noted that multiple inheritance is possible. A second hierarchy defined for the monitoring services' subgraph is an AND-tree hierarchy which denotes the decomposition of a service to subservices. It is implemented by means of the consistsOf OWL object property. As an example, a Billing service can be decomposed to VolumeOfExchangedDataMetering and VolumeOfExchangedDataAccounting subservices.
.

6.1.3 Roles class

As described in Section 5, the different actors of the PRISM system are assigned specific roles. These roles find their semantic representation as instances of the Roles OWL class. The roles fall into three main categories: data subject, data processor and Privacy Authority. Therefore, the "root" instance of the class, denoted as AllRoles, has three first-level descendants (DataSubject, DataProcessor and PrivacyAuthority) which constitute root instances of discrete subgraphs. The subgraph corresponding to the service provider includes all the members of the organisation that performs the network monitoring (e.g., the ISP), as well as potential subcontractors, third-party organisations, the research community that uses aggregated monitoring data, etc. On the other hand, the Privacy Authority subgraph concerns all entities involved in law enforcement. Regarding the data subject, it can be argued that it cannot be further decomposed. However, considering the case where the data subject is an organisation the communications of which constitute a traffic flow, the members of the organisation (e.g., the different departments or humans) can be regarded as "sub- data subjects" that differentiate the policies that apply. This topic is currently under investigation.
The data processor and Privacy Authority subgraphs are characterised by two kinds of hierarchies. The first one is an OR-tree hierarchy which serves as the means for specifying the inheritance of characteristics between the members of each subgraph and is implemented by the inheritsFromRole OWL object property and its inverse. Such hierarchies are quite typical in role-based models. On the other hand, an innovative feature of the PRISM model is the specification of an AND-tree hierarchy for the roles. This hierarchy defines the explicit membership of some roles' types to other types. As an example, PrivacyAuthorityLawfulInterceptor is a logical role which consists of the well-defined physical roles' types that must all consent and interact in order for a Lawful Interception procedure to be executed. This hierarchy is implemented by means of the isPartOfRole OWL object property.

6.1.4 ExclusiveCombinations class

The ExclusiveCombinations class of the PRISM Ontology is meant to describe how data types are mutually excluding the one the others from disclosure. As an example, each of the bytes of an IP address by itself is normally harmless from a privacy protection point of view. However, altogether they compose the actual IP address, which is certainly a personal data item and therefore the disclosure of all four of them to the same entity in a way that is able to recompose the IP address should be prevented. In that respect, each instance of the ExclusiveCombinations class is related with the Excludes OWL object property to the different data types that constitute an exclusive combination.

6.1.5 Conditions class

The Conditions class of the PRISM Ontology is purposed to specify conditions that must meet in order for some access rights to be enforced. As an example, certain access rights may be granted to an intrusion detection application if and only if there is an evident intrusion attempt, while the same access rights may be redundant in normal situations

6.1.6 Rules class

Access control rules are defined as instances of the Rules class of the PRISM Ontology, in order to regulate the execution of services. Every instance of the Rules class is associated with a {personal data type, service type, role type} triad, using the corresponding refersToData, refersToService and refersToRole OWL object properties, and defines one or more properties that specify the permitted/forbidden actions of the *role* over the *personal data type*, in the context of the execution of the *service type* under consideration, possibly along with certain complementary actions that must be additionally performed by the system. Moreover, the OWL object property appliesUnderCondition links the rule with the conditions (if any) that must meet in order for the rule to apply, with respect to the Conditions class of the PRISM Ontology.
With the use of OWL annotation properties, every rule contains the following information:

- DisclosureOfData: it defines whether the data of the specified type should be disclosed or not to the specified role in the context of the execution of the specified service.
- RetentionPeriod: it specifies the period for which the data of the type under consideration should be retained.
- ModificationPermission: it defines if the specified role should be granted with write/modify rights on the data of the specified type.

While the information above defines the "core" of the rule, additional properties specify the complementary actions that should be potentially executed:

- DataSubjectInformation: it refers to the right of the user to be informed when the rule is applied (i.e., when in the context of the specified service, the personal data of the specified type are disclosed to the specified role, or their modification takes place). Naturally, in order for this to be realised, the data subject must be an identifiable user, such as a customer of the network operator.
- DataSubjectConsent: it enables the user to be asked about explicit consent, prior to enforce the body of the rule. Similarly to the case above, the identification of the data subject must be possible for this provision to be feasible.
- AuthorityNotification: it forces the notification of the Privacy Authority when the rule is applied.

Finally, a rule may be characterised by certain meta-properties that serve for resolving conflicts between contradictory rules:

- appliesToPersonalDataDescendants: this binary property specifies whether the rule is inherited to the descendants of the specified data type, with respect to the corresponding subgraph of the ontology and the inheritance relationships.
- appliesToServiceDescendants: similarly to the case above, this binary property specifies the inheritance of the rule to the service type descendants.
- appliesToRoleDescendants: it refers to the inheritance of the rule to the

descendants of the role's type.

- `OverrideDataSubjectPreferences`: in certain cases, the user may have specified privacy preferences that contradict with the rules of the ontology; this property serves for defining which rule dominates over the other. As an example, a lawful enforcement related rule may override a confidentiality preference expressed by the user.
- `OverrideExclusions`: when two or more requested data types are mutually exclusive when about to be disclosed or processed, this binary property defines whether possible exclusions defined in the Conditions class for the data type under consideration are ignored in the context of this rule's enforcement.

Finally, each instance of the `Rules` class is optionally characterised by a textual description, that is, a human-readable description of the rule. In that respect, the `rdfs:comment` property as specified in the W3C RDF Schema [RDFS] is used.
In Figure 7, an example of an access control rule is illustrated. What this rule states, i.e., its `rdfs:comment` property textual description, is that "*for purposes related to QoS provision, a Network Administrator may have read access to the Differentiated Services Code Point (DSCP) of a traffic flow. However, no modification is allowed, while such data should not be retained by the corresponding systems. This rule is not inherited to the descendants of the specified data, service or role, while it doesn't override the privacy p references of the data subject, if any*".

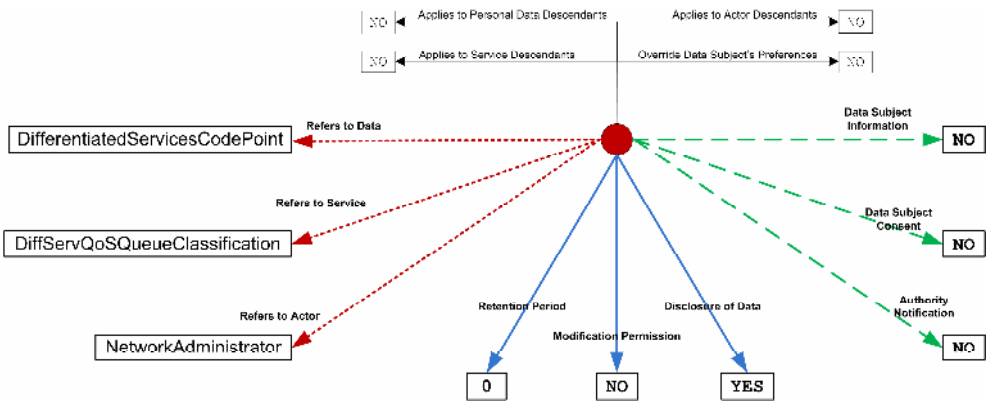

Figure 7: PRISM Ontology – Example of Access Control Rule

That is, an application run by a recognised and authenticated `NetworkAdministrator` devised for the `DiffServQoSQueueclassification` service is authorised to access the data of type `DifferentiatedServicesCodePoint`. Note that no instance of the `Conditions` class is specified for this rule, while the `DifferentiatedServicesCodePoint` data type is not a member of some instance of the `ExclusiveCombinations` class.

6.1.7 `Components` class

As described earlier, the back-end tier of the PRISM architecture incorporates several embedded processing modules for the internal execution of various tasks. These components have their semantic signature in the `Components` class of the PRISM ontology

6.1.8 DataTransformations class

Regarding the transformations between different data types that these components are able to perform, they are described by the instances of the `DataTransformations` class of the ontology. As an example, the component semantically defined as `IPv4AddressHeaderHandler` is able to extract from the `IPv4ProtocolHeader` a discrete field, such as the `IPv4SourceAddress`. This concept is implemented by means of the `transformsDataFromType`, `transformsDataToType` and `usesTransformationTool` OWL object properties that link the `DataTransformations` class with the `PersonalData` and `Components` classes of the ontology.

6.2 PRISM access control model editor

Ontology management tools, such as the very popular Protégé [PROTÉGÉ], have the disadvantage of being difficult to be used by people that lack some relevant technical expertise and understanding of certain technical notions. Therefore, a graphical software tool devised for the specification and management of the semantic model will be incorporated to the PPC. The PRISM Access Control Model Editor targets for its use people that are not coming from the worlds of computer science or engineering, such as the members of the legal department of an operator. It will hide all the technical details of the semantic model from the user, translating transparently the input provided by the user through the graphical interface to OWL code that will be consequently disseminated to the Back-End.
The PRISM Access Control Model Editor will provide –at least– three different views to its users:

- A view for the management of the different subgraphs of the PRISM Ontology.
- A view for administering the rules contained in the PRISM Ontology.
- A wizard-like view for the step-by-step creation and editing of the rules.

Since several functionalities have already been implemented to some extent, the following three figures (Figure 8, Figure 9 Figure 10) illustrate respective screenshots of the afore-mentioned views.

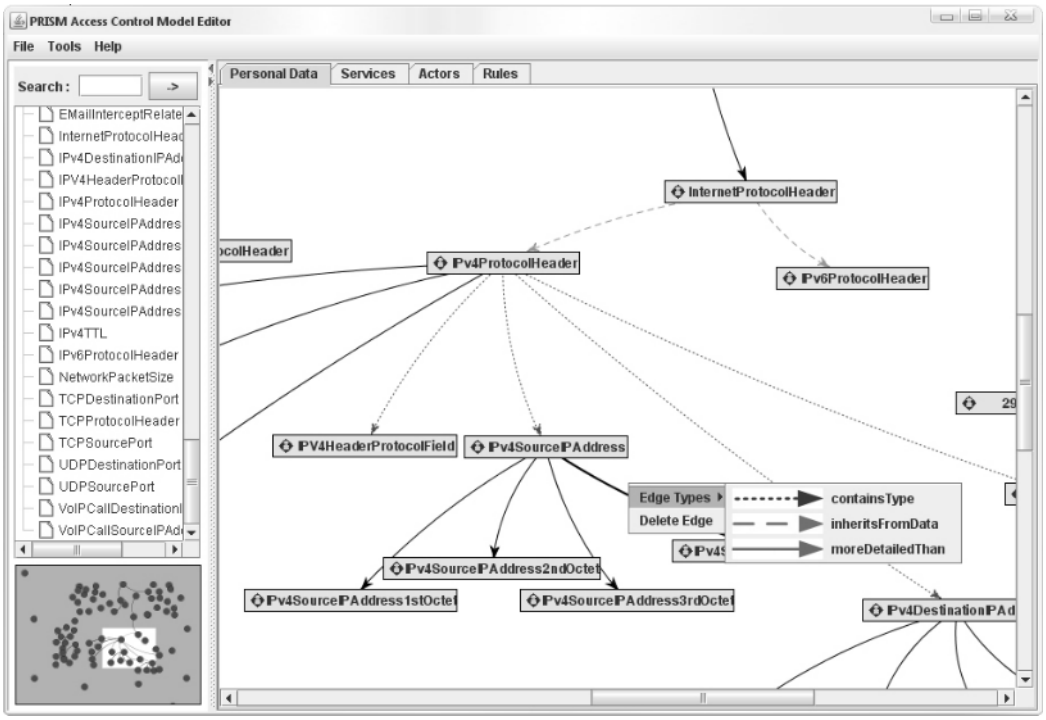

Figure 8: PRISM Access Control Model Editor: Personal Data Subgraph View

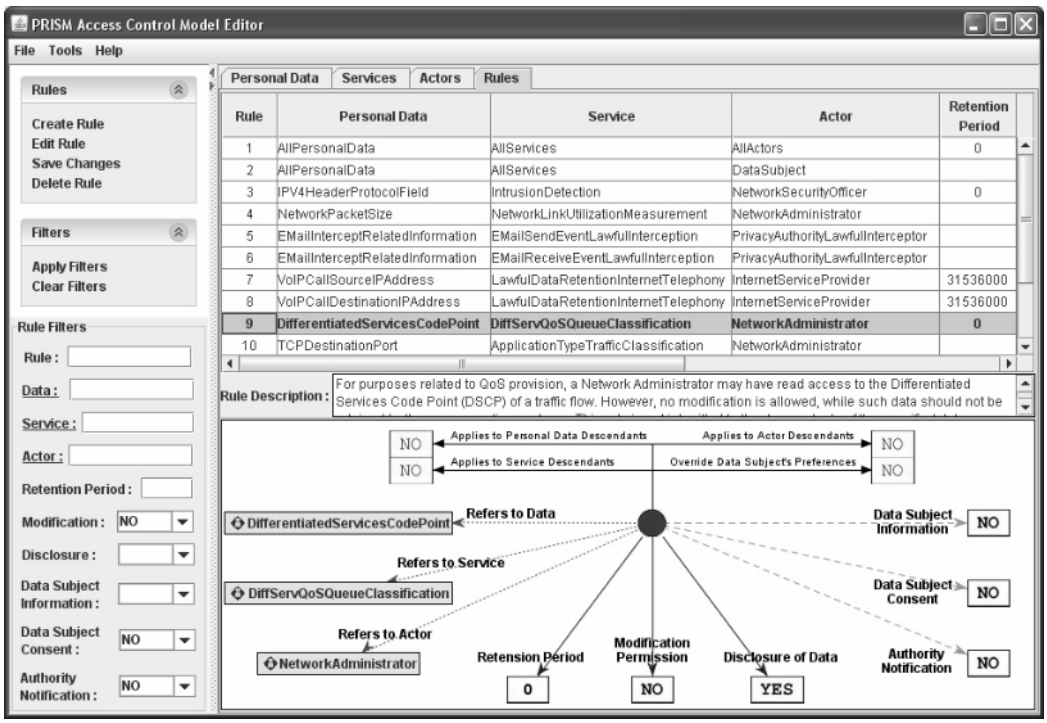

Figure 9: PRISM access control model editor: rules administration view

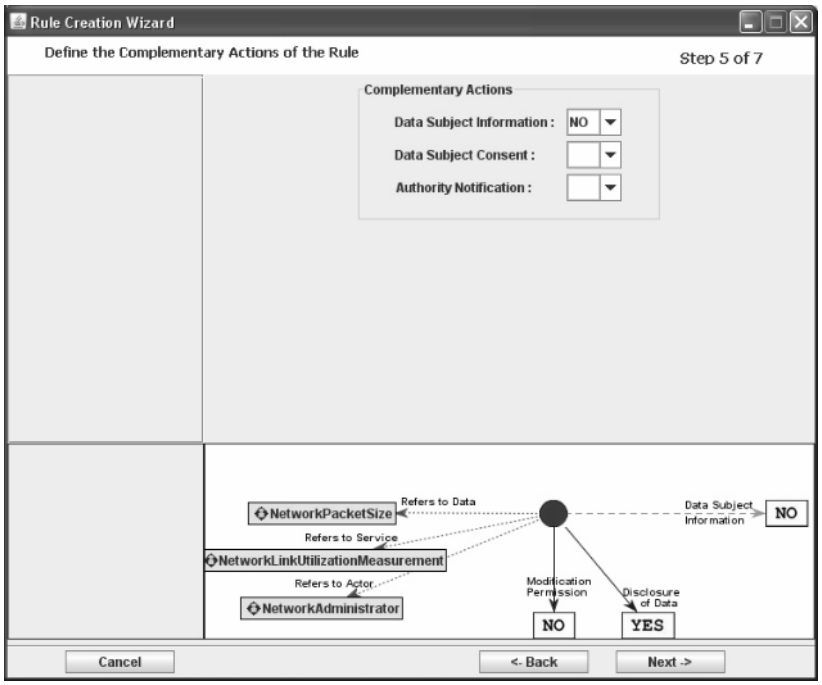

Figure 10: PRISM access control model editor: rule creation/editing wizard .

7 Data plane export protocol

There are two primary data paths to consider within the PRISM architecture. First is the data plane connection between the front-end and the back-end, by which the front-end sends preprocessed or protected information resulting from front-end analysis functions to the back-end for storage and further processing. Second is the data export connection, by which aggregated or anonymised flow or packet data may be sent on to analysis applications external to the PRISM system for further analysis or presentation.

These data paths have similar requirements. Both should be oriented toward common types of traffic data (e.g. packets, flows, aggregates) yet flexible enough to transport diverse types of traffic data and derived information, as the wide variety of analysis functions possible within the PRISM architecture cannot be bound to a single data model. Both should be based upon defined or de facto standards in order to facilitate both the substitution of components within a measurement ecosystem and to ensure the widest possible utility of the exported data. The front-end to back-end data plane, especially, should be able to send meta-data in-line to support the data protection infrastructure.

The IETF's IP Flow Information Export (IPFIX) protocol [RFC5101] meets all of these requirements, and as such we have selected it as the basis for the data plane and data export within the PRISM architecture. This section briefly describes IPFIX and explores the applicability of particular features of IPFIX to both the front-end to back-end data plane connection and the data export to external measurement applications.

7.1 Brief introduction to IPFIX

IPFIX is a unidirectional flow export protocol specified for sending flow information from an exporter to a collector with a flexible data model. It describes the data records it exports inline via templates. These templates describe record formats in terms of a standard, extensible set of information elements tailored to network traffic data export: timestamps, packet header fields, counters, and so on. The information model is extensible on the fly using a facility called "enterprise-specific Information Elements"; this allows any organisation to specify its own private information element registry.

While IPFIX is designed with flow data export in mind, it has a very flexible definition of what a "flow" is; this definition covers essentially any traffic data which share some set of common properties. A flow, for example, could be as described by the "traditional five-tuple" of source and destination IP address and port and protocol, or all the traffic sent by a specific host, or all the traffic consisting of 40-byte packets observable by a specific observation point.

IPFIX supports a mechanism called Options for the inline representation of non-flow data. Options are a special type of record. They are bound to a specified "scope" and provide additional information about this scope. Example scopes included entities within the measurement infrastructure (e.g. an Exporting Process) or within the protocol itself (e.g. a Template). Options are widely used by the protocol for interoperable extension of the export mechanism and for metadata export.

IPFIX can be adapted to a wide variety of network traffic measurement applications given these features. A specific adaptation of IPFIX to sampled packet measurement and export is specified by PSAMP. PSAMP defines a set of sampling techniques, additional standard information elements for representing packet-specific information and sampling technique metadata, and an Options-based mechanism for exporting this sampling technique metadata inline.

7.2 Data plane

For the data plane connection, it is assumed that the front-end will act as an exporter, and the back-end as a collector. The front-end should be configurable to send data to any IPFIX collector, and the back-end to accept from any IPFIX exporter that is capable of producing the information required of the specific analysis functions running on it. In other words, the PRISM architecture allows for the replacement of components, provided that they implement the necessary interfaces.

Since the front-end generally does not export raw, unprotected flow data, the data plane connection will use IPFIX less for its standard information model and more as a standard transport protocol and framing mechanism. While IPFIX's flexible flow key mechanism does allow the export of aggregate or other summary information using standard information elements, data protected by per-field encryption schemes, or information specific to a front-end/back-end analysis function pair, should instead use IPFIX enterprise-specific Information Elements to represent this information. Any inline metadata required to support per-field encryption schemes or any other protection scheme that requires direct information sharing between the front-end and the back-end should use IPFIX Options.

If a front-end analysis function yields non-encrypted but anonymised data, then standard information elements should be used, along with the catalogue of techniques and the Options-based metadata export specified in [IPFIX-ANON][4].

Similarly, if a front-end analysis function yields sampled packet data, then the information model and Options-based metadata export specified by PSAMP should be used.

The IPFIX Template mechanism provides a natural means to describe the data-plane interfaces between front-end and back-end analysis functions. Specific front-end analysis functions may specify their interfaces in terms of their output templates, and back-end analysis functions in terms of their input templates.

7.3 Data export

For the data export connection, it is assumed that the back-end will act as an exporter. The collector on the other end of the connection belongs to the external application, and is not specified by the PRISM architecture. Any information exportable via this data export should also be exportable via IPFIX Files as specified in [IPFIX-FILE]; this allows for a more natural usage of exported data for public dissemination for research studies, and for interoperability with external measurement applications with which an on-line transport connection is not feasible.

Data export should, to the extent possible, use standard, IANA-registered IPFIX information elements in order to improve interoperability with external applications. Export of information not specified by the standard IANA registry should be exported by enterprise-specific Information Elements, described inline by the Options mechanism specified in [IPFIX-ET]. As with the data plane, anonymised exported data should use the Options mechanism specified in [IPFIX-ANON] to describe the properties of the anonymisation techniques used, and sampled packet data should use PSAMP.

7.4 Summary of data and control plane connections

A summary of the arrangement of the data plane, control plane, and export interactions in the PRISM architecture is shown in Figure 11 below. Data plane interactions are shown in black

[4] Note that the authors of this in-progress standards effort are PRISM project partners. The development of this standard will proceed inline with PRISM's requirements for it.

along the left side of the diagram, and control plane interactions are shown in red along the right.

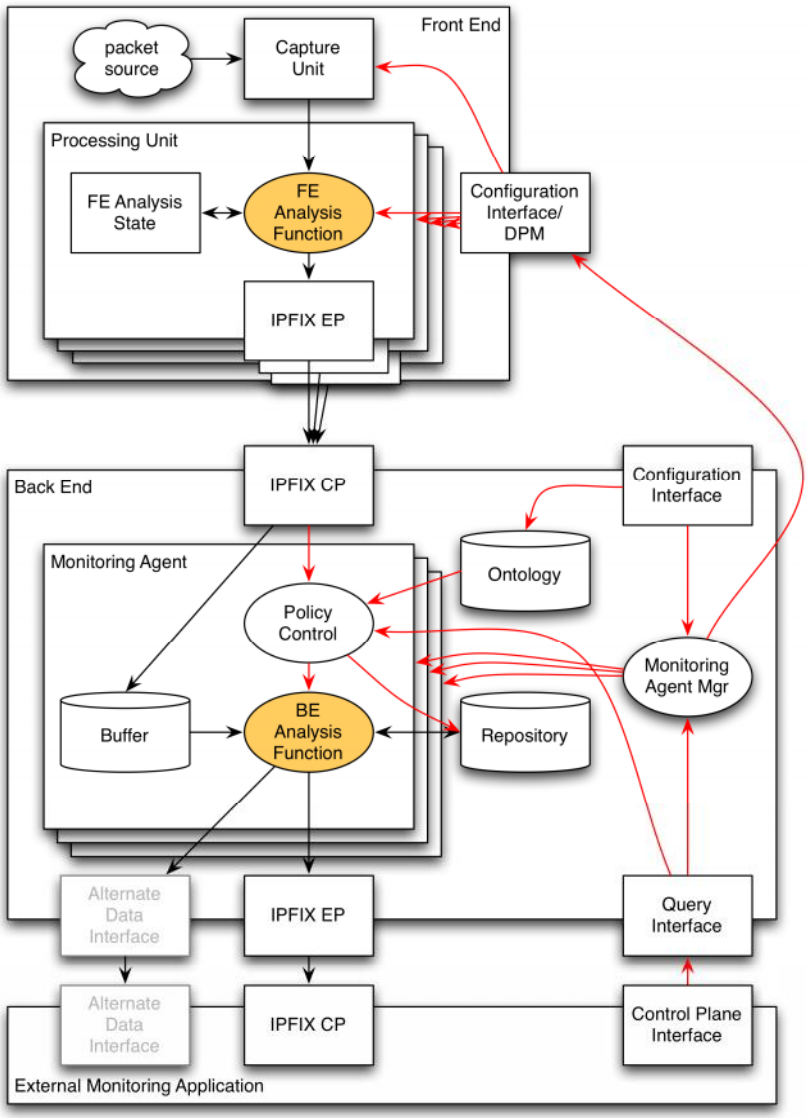

Figure 11: Summary of schematic diagram of interactions in the PRISM archit ecture

8 Monitoring framework

In this section we describe at high level how the traffic monitoring activities should be carried out within the PRISM architecture and why.

Many traffic monitoring applications use full packet information (e.g., IP headers and potentially also the user payload) for processing, even though a given task (purpose for monitoring) actually depends only on a small subset of the information available in the trace files or in live traffic. Full trace files or live traffic naturally pose a clear privacy risk as the available information can be used by a malicious monitoring application (or user) to extract far more information from the data than necessary for a given monitoring task, including legally-protected, privacy-sensitive data. In summary, this means that many current monitoring applications may tend to compromise the privacy of the end-users.

It is clear that the vast majority of common monitoring purposes can be achieved using much less information than involved in the processing of full trace files.

PRISM separates raw packet capture from additional monitoring analysis tasks. Only the information required for the actual monitoring purpose is analysed further. Therefore in many cases, the data given to the monitoring application does not include any privacy-sensitive information.

A monitoring activity receives data for processing or presentation from the PRISM back-end, the external access point towards the PRISM architecture. There are three main possibilities how a monitoring application can be adapted to be compatible with the PRISM framework:

a) Modified source data: no adaptation of application necessary; PRISM provides modified or reduced packet traces, flow data, or other traffic summaries to the application; this source data is specifically modified or reduced to mitigate privacy risk.

b) Hybrid pre-processing: splitting the application into a pre-processing stage to handle privacy-sensitive data (which may itself be further split between the front-end and back-end within PRISM) and a post processing-stage "main part" of the application.

c) PRISM conversion: full processing is done within PRISM, and only results are returned to the monitoring application (e.g., raw result data as input to the monitoring application's visualization process.)

The appropriate choice depends on the information required to obtain the desired results, and the types of data that a particular monitoring application can take as an input. Consider the following examples: A monitoring purpose requiring detailed IP address processing should by default be implemented via full conversion, while a monitoring purpose measuring the packet size per TCP port distribution can be handled by reducing packet data at the front end and passing only packet size summary or synthetic packet data to the application.

8.1 Processing of modified source data

The simplest adaptation of monitoring applications to preserve privacy is to modify or reduce the packet traces, flows, or flow aggregates within PRISM, and provide this modified data as input to the monitoring application. In this case, no modification of the application is necessary. The focus in this case is on the adaptation of the traces. This can mean on the one hand to isolate the "suspicious" flow as described in Section 2.1.1 and on the other hand to just filter the information required for the monitoring purpose and anonymised or synthesize the rest of the data (e.g. with random numbers). The decision whether information is handed over to the monitoring application can be based on the results of pre-processing in the front-

end. The monitoring application then operates as it would without PRISM, treating PRISM as a sensor or data source as any other.

This approach allows deployment of suitable monitoring applications without modification. This has a positive impact on end-user privacy with minimal effort, as currently most monitoring applications observer full packet payload regardless of the monitoring purpose. Three examples below clarify how PRISM could modify or reduce the traces before handing them over to the monitoring applications.

As a first example, getting the distribution of traffic per destination or source port addresses usually uses the captured packet headers or full packets including payload. In this case, the only information actually required is the packet size and the port numbers. For such a monitoring scenario PRISM ensures the privacy of all users. The FE deletes the payload and all header fields except the packet size and port numbers, and sends summary information to the BE via IPFIX. The BE then generates synthetic anonymised packet header fields, adds the real packet size and port numbers, and hands over the packets to the monitoring application.

As a second example, let us consider a traffic planning and network dimensioning scenario, where, in general, the key input parameter is knowledge about the traffic volumes between different source-destination pairs. Assume further that one is given a network planning tool that accepts some type of traffic trace file(s) as input parameter. This scenario can be implemented several ways within the PRISM framework. Let us discuss two options briefly.

The most straightforward method is to extract the source and destination IP addresses and the size of each packet. As shown in [D3.1.1], a one-to-one mapping of IP addresses for anonymisation is vulnerable to rather trivial attacks. Assuming that network addresses are not considered to be personal information, the IP addresses are reduced to the network portion of the address. Thus, for each packet the front-end extracts two network addresses, *dst* and *src*, together with the packet size, and hands this information to the back-end. Note that this information is clearly sufficient for the network planning purposes. At the next stage, the back-end generates synthetic anonymised packet header fields, adds the real packet size and network portion of IP addresses to the packet, and hands the packet over to the network planning application in appropriate trace file.

A somewhat more sophisticated solution to this problem is as follows. Instead of using per packet time series, for network planning purposes knowledge about observed traffic volumes between different networks should be sufficient (with appropriate time bins if the dynamic nature of the traffic is taken into account). Hence, the front-end can already aggregate several packets and provide the average byte and packet rates to the back-end. The back-end then generates artificial traffic according to the measured rates. This method has the added advantage of being more scalable, as it greatly reduces the data volume between the FE and the BE.

A third example is an IDS/IPS scenario, where it is obvious that the information cannot be reduced as much as in above. In particular, full captured packets with payload are used in applications such as SNORT, e.g., to detect malicious traffic. In other words, it is not possible to identify malicious traffic based only on some header fields and it might be necessary to provide the actual payload to the traffic monitoring application. The advantage of the PRISM system in this context is the fact that it can be used to reduce the amount of traffic inspected by the external application, which also improves privacy because less sensitive information is exposed to the third party application. To this end, the PRISM system is used to classify suspicious traffic at the front-end based on algorithms such as that presented in [BIANCHI2008]. One main criterion for the initial classification is the accuracy of the classification. Depending on the case, we can tolerate some normal traffic to be inspected by an IDS such as SNORT, while no malicious traffic should escape the preliminary detection. In the ideal case, the traffic analysed by SNORT consists solely of suspicious traffic, thus ensuring that the privacy of “normal” users is not violated.

8.2 Hybrid pre-processing at the front-end

If some privacy-sensitive data must be processed for a given monitoring task, and when the monitoring application is easily separable into stages where the privacy-sensitive work can be isolated in a first step, then a hybrid solution for integration is to pre-process the privacy sensitive information at the front-end. The results of this pre-processing, together with privacy-insensitive information from the traffic stream, is then sent to the back-end and stored for further processing. The main part of the monitoring application is outside of the PRISM system; i.e., the application retrieves the requested information from the back-end. The monitoring application performs the primary processing steps and presents the results.
An example of such an application could be an MPLS-based route optimiser. In particular, consider a task where one has set a criterion that in an MPLS network the load on the monitored link(s) should be less than 80%. If the traffic load is greater than 80% (say, in 1 minute interval), then the administrator should get information about the load per MPLS label (actual interval + five intervals before) to reroute traffic and reduce the load on the corresponding link. If the load is below 80%, then the administrator should not get any information as MPLS labels may be bound to users (privacy risk). The front-end filters the packet size and the MPLS label. For a one minute interval the packet size per MPLS and of all packets is summed up. If the load is above 80% the key is sent to the BE and stored there. The information is decrypted and sent to the monitoring application. Information that is older than 5 intervals is deleted from the database.

8.3 Full PRISM conversion

Full conversion of an application into PRISM front-end and back-end analysis functions is recommended when complex or resource-intensive processing on privacy-sensitive data is necessary. This is the case when the processing cannot be performed at the front-end due to limited storage, memory or processor power. As privacy-sensitive data cannot be given to components outside the PRISM system, the processing normally performed in the monitoring application itself has to be done within the PRISM back-end. The final results of the processing are handed over to the monitoring application. In this case the monitoring application is used for presentation of the results or for further processing.
As an example, consider a simple billing application that is based on the amount of data that have been downloaded during the period of a month by a registered customer of a provider. In such a case, the back-end can do the processing of the corresponding metadata and return to the billing application only the aggregated volume of data.

8.4 Proposed usage of the Measurement Infrastructure for Network research (MINER) in the PRISM prototype

MINER[5] is a programmable measurement infrastructure that integrates existing tools and provides higher-level services on top of the existing tools. It enables users to specify measurement scenarios, schedule (repeated) executions and retrieve the results. The services are accessible via a tool-agnostic and unified programming interface on top of which measurement applications can be developed.
The usual process for a MINER user is to use the MINER API with the following three steps:

1. Specification of scenarios (i.e. select/configure tools and monitoring agents)
2. Schedule executions of scenarios
3. Retrieving results and/or alarms

In an integration with PRISM, MINER mainly covers the specification and execution of scenarios, while the third step only provides a very limited set of results (if any, e.g. status

[5] http://miner.salzburgresearch.at

information or alarms). The main results of the monitoring tasks (i.e. the packet or flow traces) would go directly into the PRISM storage in the back-end to avoid strong dependencies between MINER and PRISM.
Figure 12 below shows the potential interaction of MINER with the PRISM FE and BE. The back-end control interface would be used by a so called "MINER Tool", which allows to control the back-end and consequently also the front-end by the MINER infrastructure via the Tool API.

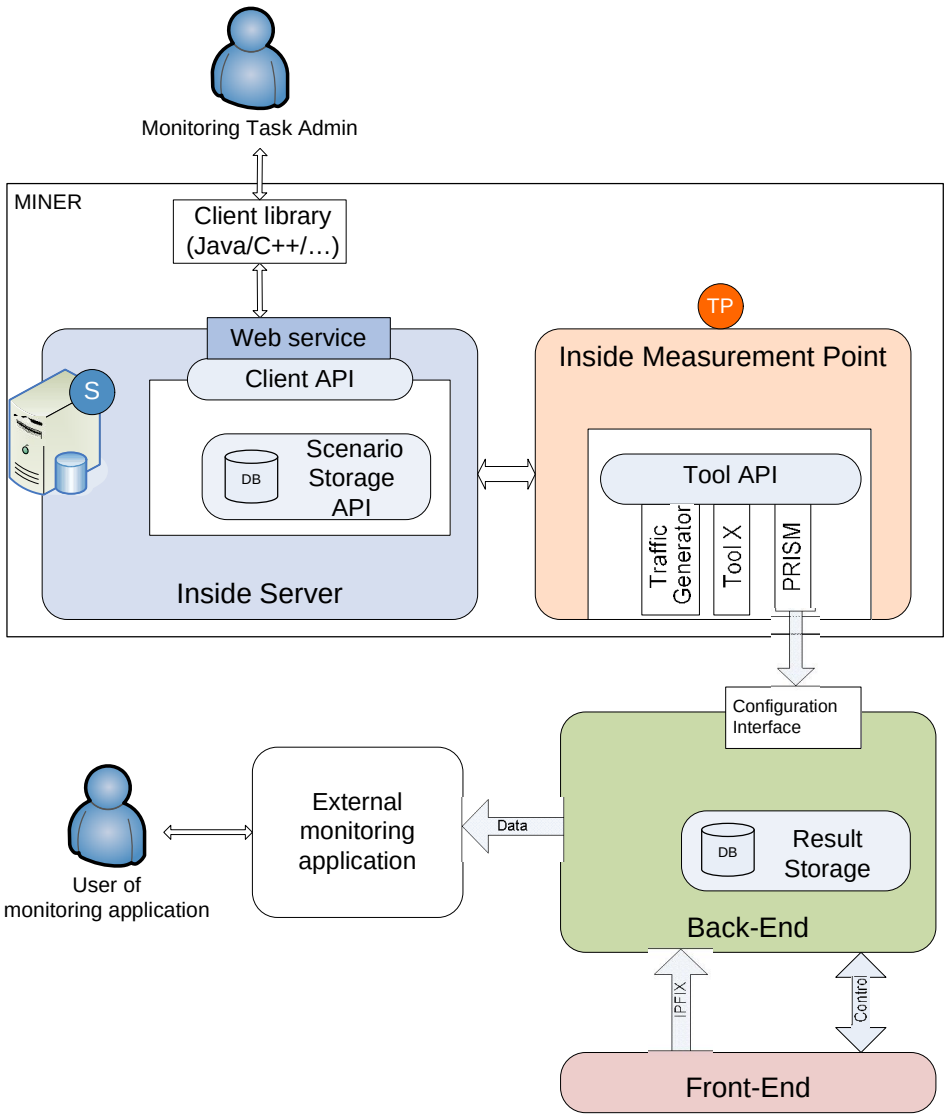

Figure 12: Usage of MINER in the PRISM project

Having this clear separation between MINER, the front-end and the back-end, the PRISM prototype has several advantages:

- MINER can be very helpful in carrying out tests
- MINER does neither store privacy-sensitive data, nor it has any point of access to this data
- Administrator of the measurement tasks cannot directly access privacy-sensitive data, as it is separated from the users of the monitoring applications
- Monitoring scenarios are stored for documentation

- Addition of new monitoring tasks to one or even multiple front-ends can be done via a simple API
- If some artificial traffic is required for an evaluation scenario, traffic generators can also be controlled by the MINER platform
- Further information during the monitoring process (e.g. SNMP information) can be collected by MINER

MINER therefore comes with a lot of features, while it reduces the implementation effort and allows easy control of the demonstration scenarios. In summary, this approach exploits functionality of MINER that can be helpful in PRISM while at the same time there is a very clear decoupling between MINER and PRISM components and no unwanted dependencies are created.

9 Conclusions

This deliverable provides a first level of specification for the PRISM system architecture. The specified system is a general-purpose network traffic monitoring system that provides strong protection for personal data in a wide variety of monitoring applications. The architecture that emerges from this document has several features unique in network traffic monitoring systems. First is the separation of trust between network observing and traffic analysis components, allowing the storage and analysis of data without allowing the storage and analysis components full access to the data; this is achieved through novel cryptographic data protection algorithms. Second is semantic access control, which provides access to information at each step of an analysis based on the wider "privacy context" of each information request. Third is the treatment of the applicable legal and regulatory environment for monitoring, not just during the design of the system but at runtime as well, by predicating access control decisions in part on the provisions of the law. While the core system is in effect a framework, providing a set of services from which privacy-aware monitoring applications can be built, the project will also adapt selected monitoring applications to operate in concert with this core.

References

[2006/24/EC] European Parliament and Council, "Directive 2006/24/EC of the European Parliament and of the Council of 15 March 2006 on the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks and amending Directive 2002/58/EC", *Official Journal of the European Communities*, No. L 105, pp. 54 – 63, April 2006.

[RFC5101] B. Claise, S. Bryant, S. Leinen, T. Dietz, B.Trammell. *Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information.* IETF Request for Comments 5101, January 2008.

[IPFIX-FILE] B. Trammell, E. Boschi, L. Mark, T. Zseby, A. Wagner. *An IPFIX-Based File Format.* IETF IPFIX Working Group work in progress, draft-ietf-ipfix-file-02.txt, July 2008.

[IPFIX-ET] B. Trammell, E. Boschi, L. Mark, T. Zseby. *Exporting Type Information for IPFIX Information Elements.* IETF IPFIX Working Group work in progress, draft-ietf-ipfix-exporting-type-02.txt, July 2008.

[IPFIX-ANON] E. Boschi, B. Trammell. *IP Flow Anonymisation Support*. IETF IPFIX individual submission work in progress, draft-boschi-ipfix-anon-02.txt, January 2009.

[RFC3198] A. Westerinen, et al., "Terminology for Policy-Based Management", *IETF RFC 3198*, November 2001.

[CHADWICK2003] D. W. Chadwick, A. Otenko, and E. Ball, "Role-Based Access Control With X.509 Attribute Certificates", *IEEE Internet Computing*, Vol. 7, No. 2, pp. 62 – 69, March 2003.

[KNIGHT2002] S. Knight and C. Grandy, "Scalability Issues in PMI Delegation", in *Proceedings of the 1st Annual PKI Workshop*, Gaithersburg, Maryland, U.S.A., April 24 – 25, 2002.

[ITU-T2005] International Telecommunication Union (ITU) – Telecommunication Standardization Sector, "Information technology – Open Systems Interconnection – The Directory: Public-key and Attribute Certificate Frameworks", *ITU-T Recommendation X.509*, August 2005.

[FERRAIOLO2001] D.F. Ferraiolo, R. Sandhu, S. Gavrila, R.D. Kuhn, and R. Chandramouli, "Proposed NIST Standard for Role-Based Access Control", *ACM Transactions on Information and System Security*, Vol. 4, No. 3, pp. 224 – 274, August 2001.

[KOUKIS2006] D. Koukis, S. Antonatos, D. Antoniades, P. Trimintzios, and E.P. Markatos, "A Generic Anonymization Framework for Network Traffic", in *Proceedings of the 2006 IEEE International Conference on Communications (IEEE ICC 2006)*, Istanbul, Turkey, June 11 – 15, 2006.

[OWL] The World Wide Web Consortium (W3C), "Web Ontology Language (OWL)", *W3C Recommendation*, February 2004, home page: http://www.w3.org/2004/OWL/.

[CPV] European Parliament and Council, "Regulation 2195/2002/EC of the European Parliament and of the Council on the Common Procurement Vocabulary (CPV)", *Official Journal of the European Communities*, No. L 340, pp. 1 – 562, December 2002.

[PROTÉGÉ] The Protégé Ontology Editor, home page: http://protege.stanford.edu/.

[RDFS] The World Wide Web Consortium (W3C), "RDF Vocabulary Description Language 1.0: RDF Schema", *W3C Recommendation*, February 2004, home page: http://www.w3.org/TR/rdf-schema/.

[XACML] Organization for the Advancement of Structured Information Standards (OASIS), "OASIS eXtensible Access Control Markup Language (XACML) TC", 2004, http://www.oasis-open.org/committees/xacml/.

[BIANCHI2008] G. Bianchi, S. Teofili, and M. Pomposini, "New directions in privacy-preserving anomaly detection for network traffic", in *Proceedings of the 1st ACM Workshop on Network Data Anonymization (NDA '08)*, Alexandria, Virginia, USA, October 31 - 31, 2008.