



Fachhochschule Hannover
Universidad Politécnica de Madrid

Master Thesis

# Analysis and evaluation of property-based NAC policies within the context of TNC

*Author:*
Álvaro Cebrián Benito

*Examiners:*
Prof. Dr. rer. nat. Josef von Helden
Prof. Dr. rer. nat. Stefan Wohlfeil

October 14, 2009

## Participants

**First examiner**
Prof. Dr. rer. nat. Josef von Helden
Fachhochschule Hannover, Fakultät IV,
Ricklinger Stadtweg 120, 30459 Hannover, Germany
E-Mail: josef.vonhelden@fh-hannover.de

**Second examiner**
Prof. Dr. rer. nat. Stefan Wohlfeil
Fachhochschule Hannover, Fakultät IV,
Ricklinger Stadtweg 120, 30459 Hannover, Germany
E-Mail: stefan.wohlfeil@fh-hannover.de

**Supervisors**
M.Sc. Ingo Bente
Fachhochschule Hannover, Fakultät IV,
Ricklinger Stadtweg 120, 30459 Hannover, Germany
E-Mail: ingo.bente@fh-hannover.de

M.Sc. Jörg Vieweg
Fachhochschule Hannover, Fakultät IV,
Ricklinger Stadtweg 120, 30459 Hannover, Germany
E-Mail: joerg.vieweg@fh-hannover.de

**Author**
Álvaro Cebrián Benito
Universidad Politécnica de Madrid & Fachhochschule Hannover
E-Mail: acbenito@gmail.com

**Thesis Declaration**

I hereby certify that this thesis is my own and original work using the sources and methods stated therein.

**Selbständigkeitserklärung**

Ich versichere, dass ich diese Masterarbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfasst habe.

**Declaración Jurada**

Por la presente declaro que esta propuesta es mi propio trabajo y hasta donde yo sé y creo, no contiene material previamente publicado o escrito por otra persona, ni material que de manera substancial haya sido aceptado para el otorgamiento de premios de cualquier otro grado o diploma de la universidad u otro instituto de enseñanza superior, excepto donde se ha hecho reconocimiento debido del texto.

Hannover, October 14, 2009

.....

Álvaro Cebrián Benito

# Contents

# Chapter 1

# Introduction

### Introduction

The objective of this master thesis is to perform a feasibility analysis of a **property-based policies** approach to the problem of attesting platforms that attempt to be a part of a corporate network where a **Network Access Control** (NAC) technology called **Trusted Network Connect** (TNC) is deployed. NAC, as it will be explained in detail afterwards, is a technology which aims to control, through defined policies, the access of devices to the network.

### Network Access Control (NAC) & NAC policies

Although there are several vendors in the market and different architecture specifications that define how to deploy NAC, all of them share one objective: determine whether the devices (and its users) that attempt to gain access to the network **satisfy the corporative requirements** that in each case could have been established in a set of policies with the purpose of restricting or granting the access to the network and the resources within, limitating therefore some security and privacy risks. This verification is performed by checking some aspects of the devices and then comparing this information with local policies, that have to be previously defined by the corporation. In particular, a NAC policy should contain some important elements [11] regarding to aspects like identity and authentication criteria, resource control access or user communications among others.

### Trusted Network Connect (TNC)

Specifically, and as the one of the main elements of the context in which this thesis has to be developed, one of this NAC technologies is the so-called Trusted Network Connect (TNC), which is an open specification of an architecture to perform NAC focused on the interoperability and the unforgeability. So that a TNC compliant deployment will, with the help of a piece of hardware named Trusted Platform Module (TPM), perform a secure and remote attestation of the devices which intend to be part of the network. The TNC architecture has been proposed by the Trusted Computing Gruop (TCG), which is according to themselves [32] an international industry standards group, that after developing the specifications amongst its members, publishes these specifications for the industry to implement and use them.

### TCG Binary Attestation Mechanism

A trusted computing platform participates, with the help of a TPM, in the remote attestation mechanism. This activity consists in reporting the configuration and status of a remote machine and checking its integrity afterwards. From the point of view of

the TCG, this verification -named in this case Binary-Attestation- is made through the measurement and collection of cryptographic hash values of several binary pieces of software, which rightness will afterwards be checked. The procedure of collecting these hash values is performed following a "chain of trust" so that every piece of software is evaluated (measured) before the control is being passed to it. These fingerprints are stored in a secure manner within tamper-resistant secure locations with the help of the built-in Trusted Platform Module so that they couldn't be modified in any way. Finally, once the hash values of every piece of software that should have been measured are collected, they can be used -whenever it is required- to perform the remote attestation of the platform.

There are some shortcomings of the *Binary Attestation* that motivate the introduction of a new approach. The most relevant are the following:

- Since the configurations are disclosed, some privacy concerns are emerge, making possible the discrimination of certain platforms and facilitating the attacks.
- The configurations are not really expressive to describe the desired platforms' charasteristics.
- The number of possible combinations of configurations is cumbersome, making its management very difficult.

**Property-Based Attestation (PBA)**

In order to justify the PBA's approach application in which the thesis is focused on, the working manner of the previously presented classical approach as well as its disadvantages will be shown and explained in detail in the following chapters. After its introducion, the property-based approach itself, as a theoretical solution to the lacks of the classical attestation, will be also described. Within the description of this promising approach will be included the fulfilled requirements that an attestation mechanism scheme should include as well as the evaluation of the feasibility of its real implementation in terms of computational and time capacity and the real fulfillment of the desired properties.

Even though there are several theoretical advantages that the PBA approach offers, the main encouraging reason that justifies its use is that eventhough if "different platforms with different components may have different configurations while they may all offer the same properties and consequently fulfill the same requirements" [21]. Therefore, instead of focusing in the specific configuration of a platform, a challenger[1] should focus in the

[1]Party that is responsible for attesting the platform of the client

properties[2] that the platform provides. It is worth noting that the knowledgement of a platform's configuration, in contrast to the properties, does not provide any other useful information to check the level of security that the platform provides.

The PBA approach entails new challenges which solution is also partially the objective of this work. Some of them are the decision of **which** kind of properties should be obtained to grant the security requirements of the corporation, **the manner** that these properties will be **acquired and measured** and its **attestation** to a third party among others.

**Corolary - (Summary conclusion)**

The objective of this thesis consists in carrying out a study of the possibility of controlling the access to a network by using a property-based policy enviroment, taking advantage of its theoretical benefits. To justify the development of this idea, it is necessary to:

- Examine the lacks that exist in the current approach (TCG Binary Attestation) that motivate the search of a new paradigm.
- Inspect the actual advantages of the PBA.
- Analyze the feasibility of the development of the paradigm.

And finally depict and develop the concept and prototype in which the PBA attestation approach is integrated within the TNC architecture, ensuring that the evaluated properties that are of interest to the challenger are actually in possession of the attested platform.

[2]Understanding a property [21] as a quantity that describes an aspect of the behaviour of a platform with respect to certain requirements

# Chapter 2

# Scope and Objectives

## 2.1 Scope

In order to give a solid background knowledge about the enviroment in which this thesis is developed beyond the general concepts that in the summary have been introduced just with the purpose of placing the reader in the context and to attemp to explain which are the objectives of this work, some essential concepts that are considered as fundamental elements of the topic, either as a consequence that they are a base upon the thesis is built on or because they themselves are theoretical concepts that should be developed along with this work, will be introduced and explained in the following pages.

Following this idea, the explanation will go through the elements that are part of the enviroment, from the general concepts to the specific details, aiming thereby to make the reader understand the upcoming reasonings and solutions that will be developed later on. Therefore, this chapter starts explaining, with an abstract high-level look, what **Network Access Control (NAC)** is as a general framework where every concept settles with a client-server architecture in which the client wants to accede to the network and the server has to make a decision about, going subsequently into further details concentrating in the server-side explaining what are the **NAC policies**, the parameters over which they are defined as well as the requirements of a good NAC policy as a base of the enforcement of the access control to the network.

Later on, the **Trusted Network Connect (TNC)** architecture will be introduced as an open specification of a specific NAC solution that is of utmost importance since it is the actual context within the solution to the problem of this work has to be developed. Afterwards, an implementation of the TNC open specification developed in the Fachhochschule Hannover named **TNC@FHH** will be explained as the -still under development- specific NAC solution implementation with which the developed product must be integrated. Finally, the promising **Property-Based Attestation (PBA)** approach will be detailed, starting with the explanation of the concept and finishing with the advantages that it presents and the lacks of the TCG Binary Attestation it is supposed to supply.

### 2.1.1 Network Access Control (NAC) and NAC policies

#### Need of control in the access to the network justification

Nowadays, the computation and specially the networks and the internet have become crucially important in our society as well as in the business. First, the corporations started

to rely more and more on the technology, placing sensitive data and resources under its control depending thus on electronic devices and networks, which the employees of the company would make use of afterwards. Since this traffic of information and access to resources contains sensitive data for the corporations, the companies want to take control of the circumstances under which the users of the network can access to certain -in general, previously specified- resources and also control in first place which users are authorized to access the network. This situation is exacerbated when other potential users that are not a part of the company gain access to the network (e.g. a guest or a partner that has temporal access privileges to the network to make use of his laptop or mobile device) or even considering some other situations such as an employee's personal computer has been infected by a virus without him to know.

This all means, that the classical vision of inside versus outside trust model is not valid anymore, and the experts in information technology had to think of more complex alternatives that fulfill the stricter secutiry requirements that the corporations demand and get through the lacks that the contemporary solutions offered.

### NAC as a technical solution to the network access control necessity

These kind of situations that have been described above which are a combination of privacy, security and information access limitation (or granting) concerns, suppose a motivating suggestion of that some sort of technology that allows the company in some way to take control of the access to the network should be developed and implanted inside the company's network boundary.

As a solution to this problem, a technological approach to the problem's solution named Network Access Control (NAC) has been proposed. NAC suppose the combination of the built-in security in the user endpoint, an authorization process to grant access across the network boundary whether it depends on the user or on the machine and network security enforcement which could be defined [10] as the performance - carried out by a policy enforcement point - of an action established in a previously defined policy in response to the state of a host. To summarize, we could say that the general basic NAC functions are:

- Endpoint management and compliance (endpoint-security assessment)
- Identity management (authentication and authorization)
- Usage policy enforcement (NAC policy enforcement)

**Shared concepts and architectures in NAC solutions**

Once the general motivating reasons that justify the use of NAC as well as the goals that the development of this technology pretends to fulfill have been already introduced, a more detailed description of a NAC architecture [8] and the basic operation of the system can be explained:

The basic framework of a NAC architecture consists in essence of three entities placed in a row so that the two entities in both ends communicate with each other through the one in the middle. Become lost in thought, from an abstract point of view, these entities are from end to end:

- In the client-side: the device that attempt to accede to the network, presenting the information that it is asked for in order to gain the access that aims.
- In the middle: the network-side components that act as an intermediary between the other entities.
- In the server-side: the server components in which the NAC policies are established and makes the decision, either granting the client the access to the network or not.

According to this description, the figure 2.1 shows by means of graphics the schematic framework of a NAC enviroment as well as the pahts of the messages that are sent by each participating entity carrying out a protocol that tries to perform the assessment of the client.

Already introduced the participating entities[1] and the basic functions and objectives that each of them have, the essential working manner that every NAC system share[2] by means of the basic succesion of delivered messages and activities performed by the entities can be introduced:

1. Firstly, the NAC system perform the assessment of the host (in some cases even as a part of the process of the user-authentication)
2. The client sends the assessment data to the server through the network components

[1] The name of the entities could vary between different NAC solutions

[2] this does not mean that every NAC solution perform exactly this activities and in the same order, but gives an idea of the way how they work [8] (e.g. depending on the system and policies defined, the client could have to pass through several reassessments or several numer of rounds could have to be performed)

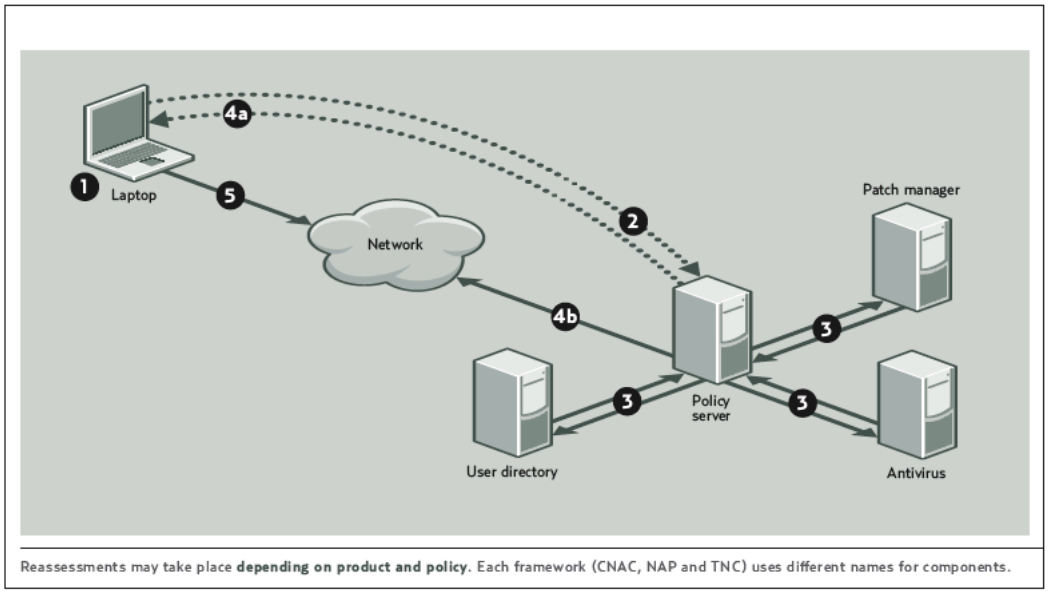



**Figure 2.1:** General NAC framework

3. The server-side[3] validates the host assessment and afterwards makes a decision about the access using, whenever it could be necessary, the help of another entities which strongly depend on the specific NAC solution.

4. The server-side, informs the host of the assessment result and sets enforcement on the host or in the network

5. Finally, the host access to the network (only if permitted) with the allowed privileges that the policy server could have decide. Between these posibilities can be found the denial of access, the grant of full access, limited access or access to the network only to be able to perform the remediation instructions, after its achievement and depending of course on the specific NAC solution, another assessment will be performed in order to check whether the client finally complies with the policy.

This protocol can also be seen as a four phases process [8]:

- Assessment
- Validation
- Decision

[3]This step may vary considerably from one solution to another because of the differences between the server architectures. The reader should pay attention in that this point besides the first one are the core of the process since they decide how and what information is collected from the client and afterwards how is it evaluated

- Enforcement

### Commercial NAC solutions and other available specifications

As a consequence of the importance that NAC has become in the field of the corporate networks security as an access control solution, several alternatives have appeared in the market offered by different software companies aiming to achieve the majority of the market quota. In spite of the fact of the existence of all these different frameworks, deserve remarkable attention three of these solutions, that could be from far considered the most important NAC frameworks that can be found in the market:

- Cisco Network Admission Control (CNAC) - Cisco Systems
- Network Access Protection (NAP) - Microsoft
- Trusted Network Connect (TNC) - Trusted Computing Group (TCG)

The two first alternatives have been introduced respectively by the well-known software companies Cisco Systems and Microsoft. **The first one**, named CNAC, is mostly a Cisco's NAC integrated solution[4] that can be [8] easlily integrated within a Cisco enviroment, but, although it can be also used along with different product of multiple vendors, its integration could become complicated since the equipment of the other vendors like routers and switches may not be able to enforce the policies sent by the Cisco's policy server. **The second** outstanding solution mentioned above is NAP from Microsoft, which is, in contrast to CNAC, just a software solution that is included in the *Active Directory*[5] and works only with the latest versions of the operative system Windows.

It is worth pointing out the existence of the Cisco/Microsoft NAC/NAP Interoperability Architecture, which is an alliance between the two companies with which they attemp to complement each other so that both solutions could coexist in the market operating with the other, so that Microsoft will give Windows and Active Directory support and Cisco will perform the support for non-Windows operative systems (and older, no compatible versions of windows) as well as the hardware enforcement through the use of Cisco

[4]Integrated in the sense of a complete software/hardware solution that can be integrated with other Cisco products in order to perform the NAC's objectives

[5]Active Directory [17] directory service is the distributed directory service that is included with Microsoft Windows Server 2003 and Microsoft Windows 2000 Server operating systems and enables centralized, secure management of an entire network

Security Agents.

Apart from the two leading commercial NAC solutions, there are in the market many others. To mention [1] the NAC frameworks developed by the companies Bradford Networks, Check Point Software, Cisco, ConSentry Networks, ForeScout Technologies, InfoExpress, Juniper Networks, Lockdown Networks, McAfee, StillSecure, Symantec, Trend Micro and Vernier Networks. Despite most of them have several similarities in their working manner and follow similar trends in order to perform the access control to the network, all of them are in any case supported and have been developed by different companies, consequently they can not interoperate with the rest as well as they are sometimes limitated so that they work only with certain platforms.

In an attemp to solve this problematic situation in which the NAC solutions do not count with mechanisms to detect lying endpoints [14] and several vendors offer their own NAC solution, that in general does not interoperate with the others, emerge the Trusted Network Connect (TNC) (metioned as the **third** leading solution in the list above) promoted and developed by the Trusted Computing Group (TCG). Trusted Network Connect is, as the TCG's TNC working group itself describes [33], an open, non-proprietary architecture specification as well as a growing set of standards for endpoint integrity that enables the application and enforcement of security requirements for endpoints connecting to the corporate network at or after the moment of the connection. The TNC architecture helps IT organizations to enforce corporate configuration requirements and to prevent and detect malware outbreaks, as well as the resulting security breaches and downtime in multi-vendor networks.

Solving therefore the mentioned existing problems, all these defined standards attempt on one hand to ensure multi-vendor interoperability across a wide variety of endpoints to attest, existing network technologies as well as policies[6] and on the other hand, [21] by means of the specification of a fully trusted hardware component that provide security functions required by operating systems[7] for trustworthy operations, in such way that -as an embedded trusted third party platform into the underlying hardware-, it collaborates as the core of the process in a mechanism called "attestation"[8] that attests the configuration of the platform determined by a set of hashed values that describes the status of the

[6] In general, there will be several different defined policies to apply depending on the specific platform, identity of the user in the client-side, etc

[7] this tamper-resistant hardware component that will be in detail explained afterwards is called Trusted Platform Module (TPM) and includes functionalities such as cryptographic functions, secure random number generator or secure storage amongst others

[8] attestation is defined [21] as the process of authenticating the configuration of a platform

endpoint, ruling out the possibity that the endpoint could lie about it.

It is also remarkable that TCG, which is an international industry standards group, consists of several members between promoters, contributors and adopters amongst which Microsoft itself can be also found.

### NAC policies

As it has been said before, the main objective of a NAC-infrastructure's development is -apart from the control of the access itself through authentication and authorizacion processes- the verification that every endpoint that attempts to go into the network fulfills certain concerns that are important to the corporation. Although in a general NAC framework these concerns can be or not related to security (as an example of a not-security related directrix, the company could take the decision that, in order to accede to the corporate network, the laptop of every employee must have as desktop background image the company's logo), the focus of this work are the security-related topics. Having said that, a crucial matter in this context is to solve the question which motivates this topic: *what is exactly considered as secure for the corporate network?*

The definition of the adequate conditions that every device that attempts to connect to the network should be in compliance with, is made by means of the establishment of a NAC policy, since the working manner of a NAC framework -and particularly, of a TNC-based NAC solution- is, firstly measuring some elements in the endpoints, sending afterwards the results of this measurement to the server, who will decide -through a comparison with the defined NAC policy- the suitability of the endpoint to access to the network, deriving eventually an access decision.

In spite of the impossibility of defining a general NAC that fits every network security requirement of every corporation -as a consequence of the different interests[9] that each of them have- there are some general shared concepts that every good NAC policy should pay attention to [14] that could be also considered as a guideline for the development of a NAC policy:

[9]It is clear that every corporation is interested in the security of the network, specially in the case they are implementing a NAC solution, but the point is that each of them could be interested in different specific concepts (e.g. while a corporation may be interested in the fact that the employees's laptops do not contain any P2P program other may not

1. Scope: defines which entities the policy applies to, since normally there will be several different policies defined.

2. Authentication: whether there will be any authentication mechanism, and, in that case, the allowed mechanisms that can be used.

3. Assessment: specifies what is the relevant information from the state or configuration of the endpoint that must be measured in order to perform an assessment whether to allow the access or not.

4. Evaluation: after receiving the selected information to measure in the endpoint, specifies how should this information be processed in order to take the decision. It is also important to mention that, the derived decision may not be a simple allow/block decision, but can be an intermediate decision as in a white-black scale there are also different grays, so that amongst these different decisions could be found the denial of access, the grant of full access, limited access or even remediation access[10] to the network.

5. Enforcement: defines how is the derived decision actually performed. The way of performing the decision's enforcement depends on the architecture of the NAC solution since it can be made by specific hardware[11], software[12] or by means of a combination of hardware and software.

### 2.1.2 Trusted Network Computing (TNC)

#### Motivation and general comments

When developing a NAC system, there are some important issues, apart from all the general functional and technical aspects that have been explain in the previous sections of this work, that must be considered. Between those issues it is worth highlighting two [14] of them:

- **Interoperability**: there should exist support for a multi-vendor infrastructure. This support must be understood in the sense of allowing the client to use the secutiry solution that he wants to as well as allowing the corporations to be independent

[10] to be able to perform the remediation instructions, after its achievement and depending on the specific NAC solution, another assessment could be performed in order to check whether the client finally complies with the policy

[11] e.g. if within the NAC solution there is an enforcement-capable switch, it will be in charge of enforcing the decision

[12] e.g. not allowing the endpoint to access to a Virtual Private Network (VPN)

of any integrated NAC solution, being therefore able to use the network hardware they want to.

- **Unforgeability**: in the sense that the entity that is in charge of checking the integrity of an endpoint by means of the measured data that it -in general, in response to a request- sends, can actually *trust* on this data.

At the moment, there is [14] no NAC solution that fulfills these two issues, since every commercial NAC framework looks down on the products of the rest of vendors, failing to carry out the first of the mentioned requirements. The second highlighted requirement is also hard to achieve since there must exist some piece of hardware within the client platform to ensure the trust of the integrity data sent by the client. This piece of hardware should also be standardized in order to fulfill the first requirement.

As a consequence of the existence of this issues that have been introduced, specially the existence of several non-standardized, non-interoperable NAC solutions in the market, the Trusted Computing Group (TCG), in an attemp of solve this situation and offer a standardized and common frame which every vendor's secutiry products as well as different infrastructures could afterwards fit with, has promote the Trusted Network Connect (TNC) architecture. The TNC could be considered as an open, non-proprietary architecture specification composed of a set of communication protocols and data formats that together set up a complete, secure and trustworthy NAC system.

The TNC architecture has been developed by the TNC working group that belongs to the Trusted Computing Group (TCG), which is an international not-for-profit industry standards group focused on developing, defining, and promoting open standards for trusted computing, that after developing these specifications amongst its members, publishes them for the industry's implementation and usage. The TCG counts with more that 170 members among which there are several important companies that are competitors in the market, granting with this to develop [32] the industry best capabilities that are vendor neutral and interoperable between them.

At this time, most of the specifications and standards that set up the TNC specification have been published, others are still under development and others have been published for public review. Many vendors [30] have already implemented the existing TNC specifications, just to mention some of them: Consentry Networks, Extreme Networks, Fujitsu, IBM, Juniper Networks, Microsoft, Q1 Labs, PatchLink, ProCurve Networking by HP, StillSecure, Symantec, Trapeze, Vernier Networks, and Wave Systems, as well as there

are several open-source implementations of the specifications amongst it is worth noting, for the importance that it has for this work, that the Fachhochschule Hannover itself has developed its own open-source implementation of the TNC specification.

Therefore, the TNC specifications pretend to define and standardize new exchange attributes[13] in the context of the network access solutions. These atributes will carry information related to [25] endpoint compliance information, software state attestation and also information pertaining to the Platform-Authentication exchange. It is worth highlighting that the term *Platform-Authentication* is usually understood in a more general sense of authentication/authorization, but, in the context of the TCG it has a different meaning and is related to two different [25] aspects of authentication:

1. **Proof of identity**: which consists in performing the platform credential authentication by means of a non-migratable within the platform key called Attestation Identity Key **(AIK)** and has no relation with the identity or actions of the user that uses the platform.
2. **Integrity verification**: which consists in performing the verification of the integrity status of a platform using the features of the **Trusted Platforms**[14].

**TNC Architectural concepts**

In this section, the TNC architecture as a whole and the existing elements within, including the **hardware basis**, its **components** established in **entities** and **layers** as well as the **interfaces** defined for the communication between those components, will be introduced and explained. Once that all these architectural concepts will have been introduced, the basic working manner of the TNC will be explain in the next section.

As it is graphically expressed in figure 2.2, in the TNC architecture there are mainly three **entities**[15] :

[13]the TNC architecture seeks to provide a richer set of secutiry atributes for its use in authorization policies by adding the ability to the NAC solution to measure and report on the security state of the endpoint platform as a part of the autentication and authorization process

[14]A Trusted Platform is such [20] that includes a TPM-Security Module which is a shielded encapsulated chip with a controlled interface to external trusted software (in a protected hardware) obtaining therefore security functions -protected against manipulations- within the platform. The detailed features of the *Trusted Platforms* will be explain later on, along with the explanation of the architectural concepts of TNC since they depend on the existence of the Trusted Platform Module, which is a component of the TNC architecture

[15]In the figure also appear, appart from these three entities, the Metadata Access Point (MAP) and some flow controllers and sensors, but they will not be explained since have no practical relation to the scope of this work

- Access Requestor (AR)
- Policy Decision Point (PDP)
- Policy Enforcement Point (PEP)

that are respectively the specific instances of the entities that appear in the abstract idea of a NAC framework, which are the client attemping to acceede to the network (in the TNC architecture known as AR), the server that establishes the access policies which the client has to be in compliance with (named PDP) and some network component that performs the actual control access (named PEP).

Defined over these three entities there are -setting up the TNC architecture together- three layers [25] gathering components that possess similar roles, so that for every entity in each layer there should be a component that is in charge of some operations that correspond to that entity in the level of abstraction of the metioned layer. These are the different **layers** indicated from the lowest to the highest level of abstraction:

- **Network Access Layer (NAL)**: in this layer that is placed in the lowest level of abstraction there are components that carry out the functions that pertatin to the network connectivity and security, which could as well, be a part of several networking technologies[16]
- **Integrity Evaluation Layer (IEL)**: in this middle-level layer are placed the components that are in charge of performing actions that involve the whole platform in the sense of overall integrity evaluation by means of using the data provided by the Integrity Measurement Layer (IML), which is in the highest level of the architecture and is more specialized. This means that the components in this layer ,on the client-side, will collect the integrity measurement information and will act as well as a mediator with the server-side. Afterwards, the components corresponding to the server-side in the same level will receive this collected information and will check it against the corresponding policy in order to take later on decisions regarding the access.
- **Integrity Measurement Layer (IML)**: this layer, that is placed in the highest level of the architecture, contains some specific plug-in components that are specialized in collecting (in the client-side) and in verifying (in the server-side) the

[16] one might recall at this point that one of the main objectives of the TNC architecture is the multi-vendor support and the interoperability. So several networking technologies should be supported by the NAC framework

information which is related to the integrity of some specific security applications or security parameters of the client's devices in which a corporation -that is who establishes the access policies- could be interested in.

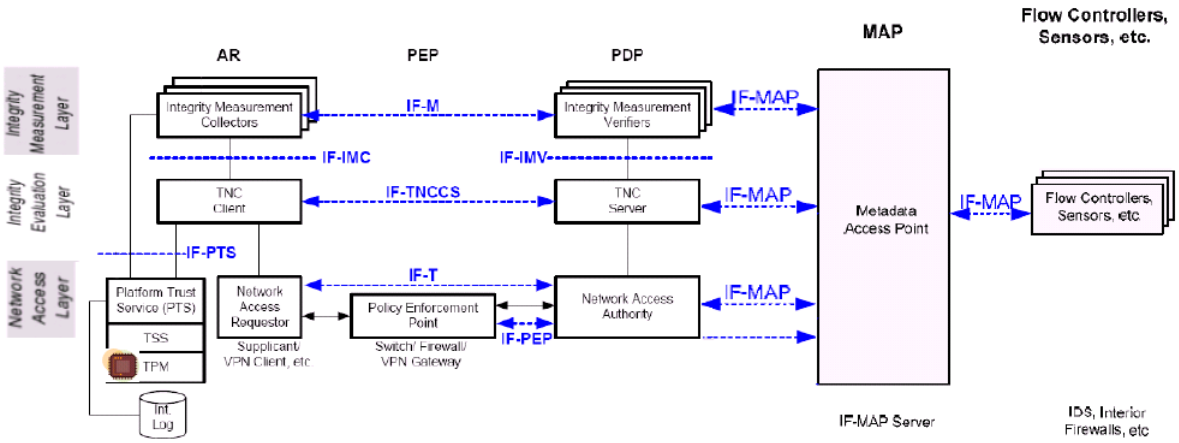



**Figure 2.2:** TNC Architecture

Once the participating entities in the TNC framework and the abstract layers that are defined over them have been introduced, getting therefore the reader into scene by providing the necessary background, every component can be now explained.

**Trusted Platform Module (TPM)** Starting with the *basements* of the architecture, one of the concepts that makes the difference between TNC and other NAC solutions is the existence of a standardized, passive[17], built-in piece of hardware within the client platform that provides some security and cryptographic capabilities named Trusted Platform Module (TPM).According to the TNC Architecture specification itself [25], the TPM provides the following capabilities:

- **Protected capabilities and shielded locations**: a protected capability is such [19] whose correct operation is necessary in order to trust the operation of the whole platform. These capabilities have exclusive access to the shielded locations[18].

[17] in the sense that the TPM does not act by its own initiative as an activ component, but as a passive one [20], working in response to requests, initiating never an interruption or other such operation and can not alter execution flow of system

[18] A shielded location is an area in which the data contained is protected

- **Integrity measurement and storage**: the process of obtaining metrics of the platforms characteristics' integrity and afterwards, storing those metrics in the Platform Configuration Registers (PCRs)[19].
- **Integrity reporting**: after having collected and stored the measurements of the platform characteristics in the PCRs, the integrity reporting is performed. This process consists in sending integrity measurement log[20] portions of the to other parties along with a set of PCRs which the these parties can use to validate the contents of the integrity measurement log.
- **Attestations**: which consits on vouching the accuracy of the information in order to decide if the third party trusts the endpoint or not.

In figure 2.4 the logical hardware components and contents of a Trusted Platform Module (TPM) are shown. Classified by functionality, these components are the following:

- **Secured input-output**: as it was explained before, the capabilities of the TPM are protected and exclusive access permissions are required in order to use them. The secured input-output manages [20] information flow over the communications bus.
- **Cryptographic processor**:
  - Random number generator
  - RSA key generator
  - SHA-1 hash generator
  - Encryption-decryption signature engine
- **Persistent memory**:
  - Endorsement key (EK): unique key that uniquely identify each TPM of which private part -only existing in a shielded location- never leaves the TPM. This key is normally generated by the manufacturer and is usually backed by an EK certificate issued by the TPM manufacturer
  - Storage Root Key (SRK): top level element of TPM key hierarchy and it is created during the ownership take
- **Versatile memory**:

[19] which are shielded locations placed in the TPM

[20] the integrity measurement log includes a list of components loaded on the platform and measured by the TPM as well as how these components have been composed

– Platform Configuration Registers (PCR): 160 bit storage location for integrity measurements that is stored into shielded locations inside TPM. Although each PCR can store the fingerprints of multiple components using a *hash-chain mechanism*[21], the specification of the TPM requires [18] a minimun of 24 PCRs.

– Attestation Identity Keys (AIK): RSA key that is used only for the attestation. Each TPM may create and own an arbitrary number of AIKs.

– Storage keys: RSA keys created during user initialization that are used to encrypt other elements in the TPM key hierarchy.

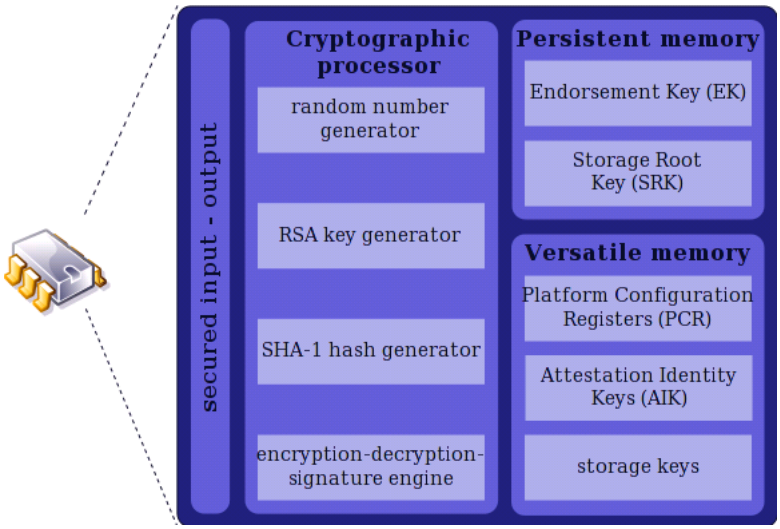



**Figure 2.3:** TPM logical components

Concentrating on the hardware components of the TPM, from an architectural point of view [20], the structure of a TPM is shown in figure 2.4

[21] The hash-chain mechanism within a PCR is the following: when a measurement value $m$ is being recorded into a PCR, the value $m$ is extended into the PCR, which results in a SHA-1 hash over the concatenation of the current PCR value and this new value ($m$):

$$PCR_i^n = SHA1(PCR_{i-1}^n || m)$$

where,
the initial value of a PCR is $PCR_0^n = 0$
$n$ denotes the index of the PCR register
|| denotes a concatenation.

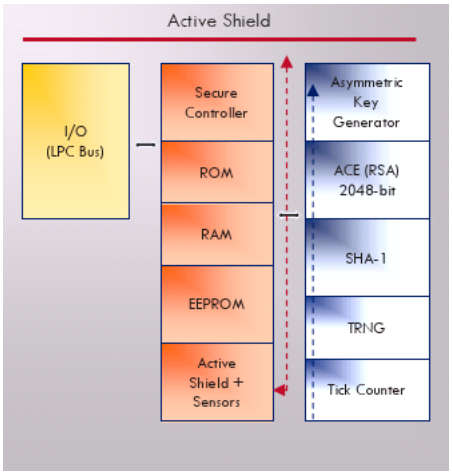



**Figure 2.4:** TPM Architecture

To take advantadge of this piece of hardware and provide anti-tamper resistance, the so-called integrity reports[22] are sent to the PDP along with PCR values -that can not be modified by the user since they are kept inside shielded locations- cryptographically signed by the TPM, so that these integrity reports reflect the *actual state*[23] of the AR to the PDP.

In the following pages, the components within the entities of the architecture are going to be introduced and explained. It is worth noting that, in order not to get lost, the reader can at any moment during the reading of those descriptions have a look to the figure 2.2 to know where is exactly in the architecture each component.

### Components within the Access Requestor (AR)

- **[Optional] TPM-related components**: when a TCG trusted platform makes up the host environment[24], there are some additional components [25] that lay on the Access Requestor:
  - Platform Trust Service (PTS): service of the system that exposes the available trusted platform capabilities that were explained above.
  - The TCG Software Stack (TSS): middleware stack that enables applications to use higher level interfaces for communication with the TPM support func-

[22] an integrity report is a set of collected integrity information that attempts to offer the verifier an overview of the access requestor's device state

[23] since neither the AR, nor any other entity or attacker could modify the contents of an integrity report without making clear that it has be done

[24] the existence of a TPM is not mandatory -but highly advisable- within the TNC architecture

tions. Amongst unlimited key storage, key caching and higher-level interface abstraction are found.

- The Trusted Platform Module (TPM): the actual piece of hardware soldered to the platform that was explain in the previous section.

- **Network Access Requestor (NAR)**: component that is in charge of the negotiation -implemented as a software component (e.g. Supplicant in 802.1X)- and establishment of the network access with a given network. The NAR is must implement the network layer protocols as well as the message transport paying attention to the security and some others related concerns. In the case that there were more than one connection to different networks, these connections could be handled by several NARs on a single AR.

- **TNC Client (TNCC)**: software component that acts as an intermediary between the PDP and the client's components that are responsible for performing the integrity measurements (IMCs) ans assists them. So, the TNCC, by means of the aggregation of the integrity measurements that receives from the IMCs, produce the report of the local platform and performs the Integrity Check Handshake (TCG attestation protocol).

- **Integrity Measurement Collector (IMC)**: software component that measures some AR's integrity security aspects (e.g. software programs installed, status and versions of these programs, patches, antivirus parameters, firewall state, etc). The design of the TNC architecture has been developed in such a manner that within the AR there will be multiple IMCs[25] that will interact with a TNC-Client/Server (or even with more than one), allowing the customers to implement integrity policies such that they could involve a wide range of products from different vendors.

**Components within the Policy Enforcement Point (PEP)**

- **Policy Enforcement Point (PEP)**: is the component that actually controls the access to the protected network where the TNC framework has been deployed and therefore the access policies are implemented by means of a Policy Decision Point (PDP). In order to perform that control, the PEP consults the PDP to determine whether the access should be granted (e.g. the Authenticator in 802.1X, often implemented within the 802.11 Access Point).

[25] e.g. an IMC for each secutiry component

**Components within the Policy Decision Point (PDP)**

- **Network Access Authority (NAA)**: this component that is in the lowest layer of the architecture is the one that decides whether the access to the network of an Access Requestor (AR) is granted. In order to take that decision, the NAA consults a TNC Server to check the AR's integrity measurements compliance to the PDP's security policy.

- **TNC Server (TNCS)**: component that **manages** the flow of messages between the Integrity Measurement Collectors (IMC) and Integrity Measurement Verifiers (IMV) that gathers the *Action Recommendations* that the IMVs provide to eventually combine, based on a policy, all those recommendations into an overall *Action Recommendation* to the NAA.

- **Integrity Measurement Verifier (IMV)**: this component that is in the higher layer of the architecture is in the charge of verifying a **particular aspect** of the AR's integrity by means of the integrity measurements received from the IMCs in the Access Requestor. The TNC architecture has been developed so that several IMVs may reside on a PDP.

**Interfaces to provide communication between the components**

- **IF-PTS**: interface within the Access Requestor regarding the TPM and the Platform Trusted Service (PTS) that can be used [23] by the Network Access Requester (NAR), TNC Client (TNCC) and the Integrity Measurement Collectors (IMCs) in order to report on endpoint integrity state. This interface can be used [23] to improve some *trusted computing* objectives such as:

  - Enabling the platform's components to participate in Transitive Trust chains.
  - Computing and collecting the integrity measurements over TNC and other application components.
  - Formatting the integrity measurements collected by TNC and other applications to favor the interoperability.
  - Local verification of the measurements (in the client-side).

- **IF-PEP**: standard interface between the Policy Decision Point (PDP) and the Policy Enforcement Point (PEP). Although it has not been yet finalized for the moment, the TCG plans to develope specifications that describe different protocol bindings

with several protocols such as RADIUS, Diameter or SNMP. IF-PEP provides [26] the following features:

  - Endpoint Isolation[26]
  - Network Access Decision Transport
  - Support of Remediation and Handshake Retry[27]

- **IF-T**: interface [34] in the lowest layer of the architecture. The IF-T protocol provides a transport service to carry the TNCCS protocol messages over the network. Recalling to the interoperability goal of the TNC, this interface should be able to operate with different network technologies (in an ideal scenario, it should be able to operate with all the existing networking technologies). The usage of the IF-T protocol permits the existence of endpoint assessments as they are joining the network as well as after the endpoints are already inside the network, either by its own will or by the verifier's desire.

- **IF-IMC**: standardized interface [27] between Integrity Measurement Collectors and the TNC Client. It works actively in cooperation with the IF-IMV inteface, which connect the IMVs to TNC Server in the Policy Decision Point (Verifier). This interface is used by the TNC Client to gather the integrity measurements (that express the endpoint's state) performed in the AR by the IMCs, so that these integrity measurements can be communicated -within Integrity Check Handshakes- to the IMVs in the server-side to be later on verified. The following features are provided by IF-IMC:

  - Integrity Check Handshake: as it was said before, the IMCs and the IMVs send messages to each other always during a Integrity Check Handshake in which the IMCs send integrity measurements to the IMVs who, optionally, can answer sending some remediation instructions or requests for more information, mantaining possibly this dialog during a while until the IMVs decide on their IMV Action Recommendations.

[26] Isolation is a kind of network access that provide the posibility of acceding just those network resources that are necessary for remediation or basic network access

[27] Remediation is a solution that the network can provide the Access Requestors to the problem that the AR were not in compliance with the access policy, so that it is advised to follow certain instructions that could mend that situation, providing afterwards the posibility of performing another Handshake protocol in order to check whether the AR is in compliance to the access policy

  - Connection Management: once a TNCC-TNCS connection has been established, the `ID` of that connection should be conserved across many Integrity Check Handshakes, so that the IMCs and IMVs could keep state information associated with an earlier handshake as well as allowing an IMC to request a handshake retry for a particular connection (e.g. once it has completed the remediation instructions sent by an IMV)
  - Remediation and Handshake Retry: there are many situations in which retrying an Integrity Check Handshake is necessary.
  - Message Delivery: the protocol provides support to the exchange of messages between IMCs and IMVs. These messages have a standardized structure consisting of a message body, a message type, and a recipient type.
  - Batches: in order to simplify the development of IMCs and IMVs and because of all the possible underlying protocols (amongst some of them may require the participants to send the messages in turns as in a half-duplex communication), the IF-IMC groups the messages into batches.

- **IF-IMV**: interface between IMVs and a TNC Server that has as main functionalities [24] receipting the integrity measurements sent from the IMCs within the Access Requestor, enabling message exchanges -exclusively within Integrity Check Handshakes- between the IMCs and the IMVs and allowing IMVs to supply their recommendations to the TNCS.

- **IF-TNCCS**: interface in the middle level of the architecture that [31] defines a protocol and the necessary data formats for the exchange of messages between the TNC Client and the TNC Server. It is worth highlighting that, since the TNC Client and TNC Server act respectively as the basis for the IMCs and IMVs in the protocol stack, the following kind of information [31] could be sent amongst the sent messages:
  - IMCs $\longrightarrow$ IMVs messages (e.g. integrity measurements)
  - IMVs $\longrightarrow$ IMCs messages (e.g. requests for integrity measurements or remediation instructions delivery)
  - TNC Clients $\longrightarrow$ TNC Servers messages (e.g. control messages)
  - TNC Servers $\longrightarrow$ TNC Clients messages e.g. IF-TNCCS Access Recommendation message)

Note that as in every protocol, the messages encapsulated in a level $n$ that come from the levels above $n+1, n+2$...(in this case the two first mentioned messages that intercommunicate the IMCs and the IMVs) are transparent to the components in that level. So that, the components in this layer don't go beyond the delivery of these messages, trusting the underlying layers to provide an appropriate secure authenticated channel and make themselves sure that the messages are delivered to the correct TNCC or TNCS according to the corresponding case.

- **IF-M**: is a -still under development- attempt to deploy an application level protocol [28] able to carry standardized *Integrity Check Handshake* messages between a variety of IMCs and IMVs from different vendors, providing thus interoperability.

Prior to the existence of this specification, the communication between the IMCs and IMVs was ment to be made by means of vendor-specific messages, so that the interoperability between products, which is a goal of the TNC architecture, was not possible. The IF-M defines thus the standard message formats to communicate IMCs and IMVs as well as a set of a TNC standard attibutes[28] to define the state of a software product[29] of a certain component type[30] that can be used. Since -even as the IF-M specification delivered by the TCG itself mention- it is expected that in the future both models will be used along with each other, taking therefore the advantadges that each of them provide[31]. As a consequence of a possible mixed paradigm, besides the definition of standard attributes for the exchange within the also standard messages, the attibutes namespace can be extended with vendor-specific attributes, allowing therefore the existence of a mixed model in which both approaches can be combine within the same framework.

Going into further technical details, the IF-M is a multiple roundtrip messaging protocol which main goal is providing the IMCs the capability of sending integrity measurement information to the IMVs in the PDP for its evaluation against a network security policy. This information is sent in form of standardized messages carried by the IF-TNCCS protocol to whom the IMVs can reply as well through IF-M messages in which they may request additional measurement data or send its

[28] **Attributes**: e.g. "`Product Information`", "`Numeric Version`", "`Operational Status`"
[29] **Software product**: e.g. antivirus Norton, operating system Windows XP, etc
[30] **Component type** within an endpoint (e.g. firewall, operating system, anti-virus, VPN, etc)
[31] on one hand [28] interoperability between IMCs and IMVs from different vendors, on the other the tight integration that the vendor-specific IF-M messages provide

IMV Action Recommendation[32].

As it has been said before, the IF-M messages are carried within IF-TNCCS messages as a consequence of the fact that this protocol is located in the underlying layer of the architecture providing -independently of the underlying protocol- multi-round trip reliable transport as well as end-to-end message delivery to suscribed parties.

In the following tables will be exposed and explained the standardized format of the IF-M protocol messages. Starting with an overview of how the IF-M message is encapsulated inside the IF-TNCCS messages, following with the IF-M message format itself, and finishing with the format of the IF-M message's payload, that is actually a set of attributes.

IF-M Message within an IF-TNCCS Message:

| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |
|---|
| IF-TNCCS Header<br>Includes Overall Length |
| IF-TNCCS Message of type TNCCS-IF-M-Message<br>Includes IF-M Message Type (for TNCC/TNCS Routing)<br>Containing IF-M Component Type (e.g. firewall) |
| IF-M Message Header (Includes Version & Message ID) |
| IF-M Attribute (e.g. Product Information) |
| (...) |
| IF-M Attribute (e.g. Operational Status) |

As it can be seen in the table above, the IF-M is encapsulated in the payload of the IF-TNCCS message. In addition to that, in the header of the IF-TNCCS message itself, should be remarked for TNCC/TNCS routing purposes that it contains an IF-M message within[33]. It is also necessary pointing out in the IF-TNCCS header the endpoint's component whom the message is related to, so that the TNCC/TNCS will be able to route the message respectively to the corresponding IMCs/IMVs, that would have registered themselves for the messages related to a specific component

[32]The answer could include as well a set of remediation instructions for the IMC to perform them in order to bring its associated component within the endpoint into compliance with the corresponding policy

[33]As in the TCP/IP architecture, the IP level must specify the transport level protocol, which is located in the immediately subsequent layer

type (e.g. firewall, operating system, anti-virus, VPN, etc). So, instead of existing an IMC/IMV pair in which each one communicate with the other through a vendor-specific IF-M, the IMCs and IMVs are communicated with each other by means of an standardized protocol by registering themselves for a specific component type they are insterested in.

IF-M Message Format (Header and attributes):

| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |
|---|
| Version \| Reserved |
| Message Identifier |
| Attribute 1 |
| (...) |
| Attribute n |

The table above define the IF-M messages format, just to mention that in the header it includes the **version** of the protocol, a **reserved** space for future use and an **message identifier** which defines uniquely a message from a particular IF-M sender, that can be used in return to refer to it. In the body of the protocol there is a **set of attributes** concerning to the component specified in the IF-TNCCS header. The format each of these fields that define an attribute is shown in the next table:

IF-M Attribute Format (payload of the IF-M message):

| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |
|---|
| Flags \| Vendor ID (Attribute Type Space) |
| Attribute Type |
| Attribute Length |
| Correlation ID |
| Attribute Value (variable length) |

It is worth highlighting that an IF-M message must mandatory contain exactly one IF-M Header but 0 or more attributes, each of them must contain the IF-M Attribute Header defined above. The elements in every IF-M Attribute Header are the following:

– `Flags`: just the two first bits are at the moment meaningful. The bit 0 indicates whether the recipients should skip any unsupported attribute or not[34]. The

[34]In the case that the recipients must not skip an unsupported attribute, they should send an error message

bit 1 indicates the use of the field `Correlation ID` that will be explained afterwards.

- `Vendor ID`: indicates the name of the attribute type namespace owner. As a standard protocol, IF-M allows the existence of attributes (e.g. "`Product Information`", "`Numeric Version`", "`Operational Status`", and so on) from different vendors for the same component type (e.g. firewall, anti-virus, etc) within the endpoint. To achieve this, the owner of the attribute types that are being used within the message has to be defined. For example, all the standard attribute types must use the TCG SMI Private Enterprise Number Value, which means that `Vendor ID = 0x005597)`
- `Attribute Length`: indicates the length in octets of the attribute including the TLV header. If the attribute is not supported by the IMC, this lenght can be used to skip it and jump to the next one.
- `Attribute Value`: contains information that varies depending on each attribute.

### How does TNC Works & Binary Attestation

At this point, the whole elements of the TNC architecture as well as a basic idea of the overall functionality have been already explained. Nevertheless, this section attempts to clarify how does TNC work in the context of the Trusted Computing Group model.

Every tool used in a platform aiming to provide security will work as it is expected only in the case that the underlying software and hardware were secure, in particular the underlying operative system and the PTS[35]. Thus, on platform startup a *Trusted Boot* is necessary. A Trusted Boot consists basically in performing [21] a *Chain of Trust*, in which all the participating components in the boot process, starting from BIOS[36] as the first element and ending with the operative system as the last one, are measured[37]. This measurement consists in a set of cryptographic hash values calculated by the previous element in the chain, that afterwards are stored into a PCR within the TPM. Using the TPM's tamper-resistant registers to store this information granst that it could not be modified. The first change to the normal boot process that must be established in order to make

[35]Platform Trust Service (PTS): system service that exposes the available trusted platform capabilities

[36]Basic Input Output System

[37]The objective of this measurements is checking whether the components have been modified in the meanwhile the last boot was made, the versions of these components or the even the components themselves are suitable for the endpoint's verifier. Although this delivery of information will be performed only once the platform attempts to access a TNC-compliant network, the boot is the only moment in which these measurements can be made

this chain is, as the reader can easily notice, blocking the BIOS boot by intercepting the boot process to execute in first order a set of instructions that will measure the BIOS itself as well as modifying the rest of elements to give them capabilities to measure the next element in the chain. The combination of blocking the BIOS boot and the hardware component TPM is known as the **Core Root of Trust for Measurement** (CRTM) [18], and it takes that name because they both need to be trusted to be able to trust afterwards the measurements collected during the trusted bootstrap.

Once this measurements have been collected and upon the connection to a TNC-compliant network, a potential verifier within the network could compare [36] the platform's state at the immediately subsequent moment after the (trusted) boot process -since it was securely stored within a PCR register (or several) inside the TPM- and compare it against some accepted value. In the case the platform would have presented a set suitable post-boot configuration values, the operative system would have been proven to operate correctly and therefore, to provide a stable basis over the future execution of programs can be (trustwortihy) measured and verified through the **remote attestation** process.

The **TNC Handshake** is performed by an approach with three phases [36, 12, 3]: assessment, isolation and remediation as shown in figure 2.5. As it has been already explained, in the **assessment phase**, the metrics related to security concerns collected by the IMCs in an Access Requestor (AR) trying to gain access to the network are examined by the corresponding IMVs in the Policy Decision Point (PDP) comparing them to its own network access policies. From all the verifications made by the IMVs, the PDP makes an access decision that will afterwards communicate to the Policy Enforcement Point, who will perform the actual enforcement of the decision. This decision could imply that the access to the network can be granted or rejected to the AR or, even in the case the AR would have been authenticated but some of the IMV's integrity-verification procedures would have failed, an **isolation** process may be realized. This isolation process consists of quarantining [12] the *non-healthy* endpoints in another network, where the PEP will forward the instructions sent originally by the PDP of which objective is **remediate** the state of the endpoint, so that if it executes them correctly, could, after a new assessment phase, grant access to the network (or repeat the whole process depending on the decision maked by the PDP).

These assessments are not realized just in the moment of the connection, but several times during the whole connection. Both the TNC Client and the TNC Server can initiate the assessments [27] in the following cases and under several situations described in the

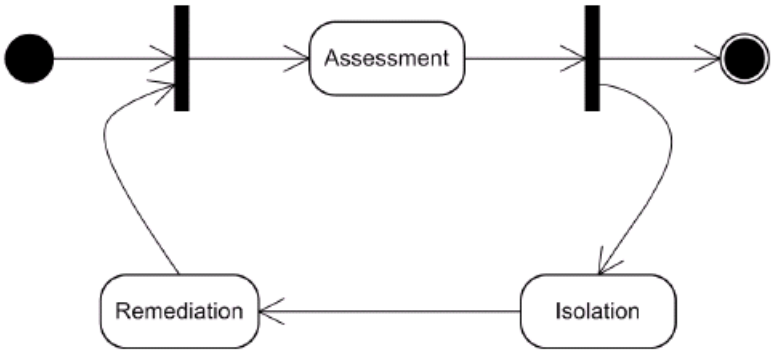



**Figure 2.5:** TNC Handshake

IF-T specification. Since there is a cumbersome number of different situations that can favor the performance of an assessment or a re-assessment, in this section all of them will be merely summarized [34]:

1. TNC Client initiated assessment or reassessment.
2. TNC Server initiated assessment or reassessment.
3. TNC Client establishes open connection for subsequent (TNC Client or TNC Server initiated) assessments.
4. TNC Client and TNC Server send IF-TNCCS messages outside of an assessment. This use case may not impact IF-T unless IF-T is aware of IF-TNCCS state (start/end of an assessment).
5. TNC Client and TNC Server use L3 and TLS IF-T connection to exchange non-TNC messages.
6. Session reuse for reassessment.
7. TNC Client dynamic discovery of TNC Server address prior to joining network
8. Security protected assessment.

The assessment is performed in the context of TNC by an [6] offered functionality called **binary attestation**. This funcionality allows a remote party (a PDP within TNC) to get a report with the actual **binary configuration**[38] of the endpoint's platform (i.e. an AR in the TNC context) by the given signature provided by the AR's TPM on the reported configuration.

[38] understood as the binary files (executables or libraries) in use inside the endpoint's platform

### 2.1.3 TNC@FHH

TNC@FHH is [14, 3, 12, 13] an open source based implementation of the TNC achitecture, developed at the moment at the University of Applied Sciences and Arts of Hanover (Fachhochschule Hannover) and that was started in order to gain experience with the functionality, interoperability and feasibility of the TNC approach. TNC@FHH implements all the core TNC architecture's AR and PDP entities components as well as most of the interfaces between them.

Figure 2.6 represents the actual state of the TNC@FHH's architecture. Within the graphics, the green boxes highlight [14] the components that have been developed within the TNC@FHH project, the green ovals indicate the interfaces and the orange boxes represent the open-source software that has been integrated within the architecture.

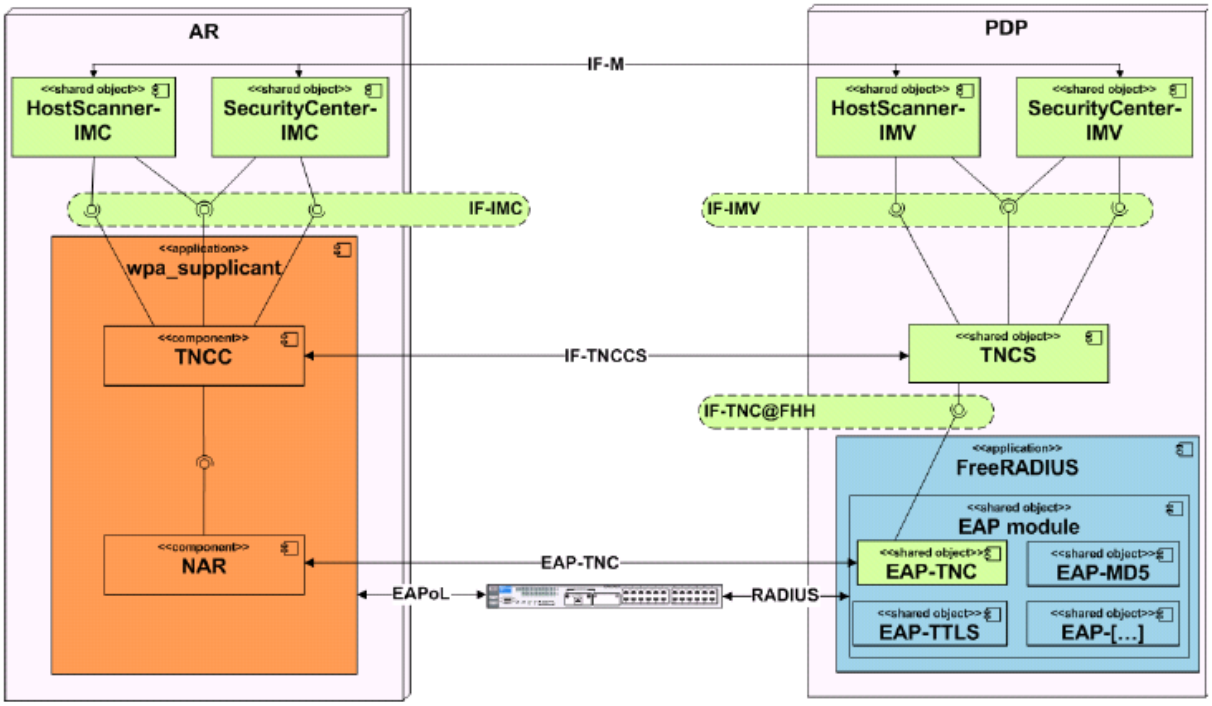



**Figure 2.6:** TNC@FHH Architecture

Although the interoperability level potential is very high as a consequence of the openness of the specifications[39]In order to provide a higher level of interoperability [3], there was a change to the original plan of developing a entire open source based TNC im-

[39]In particular [3] the following TNC components have a high degree of interoperability between each other because of the to the high quality of the specifications mentioned beside:

plementation. Thus, the development of a TNC Client (TNCC) and a Network Access Requestor (NAR) within the Access Requestor (AR) from the scratch was stopped since this implementation was -as an access requestor to a network- giving support just to TNC, lacking therefore other EAP methods[40], proposing as reasonable alternative contributing with the further TNC developments that could be made to already existing supplicants (e.g. XSupplicant and wpa_supplicant).

Going into further technical details of the implementation:

Within the **Access Requestor** (AR), as it was already said before, the **TNC Client** (TNCC) and the **Network Access Requestor** (NAR), that originally had been implemented from scratch [14] by assembling EAP-TNC packets and communicating them via 802.1X and EAPoL to and from the switch, have been replaced by open source supplicants that -appart from TNC- also support other authentication methods[41]. This situation has been represented by highlighting as orange boxes both components in figure 2.6.

The communication at the **Network Access Layer** (NAL) realizing the actual transport of the packets over the network is made through the protocols EAPoL and RADIUS, encapsulating the sent messages by using the EAP-TNC method.

Within the **Policy Decision Point** (PDP), which despite of running under Linux supports both Windows and Linux on the AR side, the **Network Access Authority** (NAA) has been implemented by extending the FreeRADIUS server running EAP-TNC (with a module) inside of an EAP-TTLS tunnel.

The rest of elements have been implemented from the scratch:

- **TNC Server** (TNCS), which has been attached to the FreeRADIUS server by the EAP-TNC module.
- Exemplarily **Integrity Measurement Collectors** (IMCs) and **Integrity Mea-**

- IMCs and TNC Client, due to IF-IMC.
- IMVs and TNC Server, due to IF-IMV.
- TNC Client and TNC Server, due to IF-TNCCS.
- NAR and NAA, due to IF-T.
- NAA and PEP, due to IF-PEP.

[40] Extensible Authentication Protocol
[41] Specifically `wpa_supplicant` and the `Xsupplicant`

**surement Verifiers** (IMVs).

- **Horizontal interfaces** between the components.
- **Vertical interfaces** between the components.

The TNC@FHH does not have any kind of unforgeability in order to detect lying endpoints since at the moment the TPM support has not yet completely implemented.

### 2.1.4 Property Based Attestation (PBA)

In this section, the PBA conceptual approach is explained, deferring the motivating reasons of its use including the deficiencies of the TCG attestation approach as well as some proposed solutions for its implementations to following sections.

When a device that attempts to gain access to a network is forced to fulfill certain security requirements, the network's administrator will generally think, at the moment of defining the access policies, of abstract security properties which such device must be in compliance with. The problem is that a property, interpreted as [21] *a quantity that describes an aspect of the behavior of the platform (or application [5]) with respect to certain requirements*, is an abstract concept which fulfillment is not easy to extract from the platform (or the program) and that is one of the reasons because different approaches have been used for the meantime.

Moreover, as it was already said in the corresponding point in the summary, although "different platforms with different components may have different configurations, they may all offer the same properties and consequently fulfill the same requirements" [21]. Therefore, if a verifier is only interested in the properties that a platform has within and these properties do not have a linear relation to the configuration that provides them[42] , the mechanism to attest an endpoint platform should depend only on the properties that the platform presents since:

1. the verifier is actually interested only in the properties.
2. the set of possible properties is smaller that the cumbersome number of different configurations, so the computational effort is lower.

[42]There is of course a relationship between a platform's configuration and the properties that it provides, but this relationship is not linear in the sense that not only one configuration (but several) can provide a property

3. the actual configuration of the platform is not disclosed because the attestation is performed through the evaluation of the properties. Thus the privacy is keeped and no underlying HW/SW could be discriminated.

The desirable requirements [21, 16] that an infrastructure capable of performing a property-based attestation mechanism should have are the following:

1. **Availability**: even if the underlying configuration of the platform changes, but the properties are mantained, any sealed data[43] should be still accesible.

2. **Security**: the security of the attestation (and sealing) functionality must not be reduced.

3. **Scalability**: the infrastructure should operate in an typical corporative environment with a large number of machines.

4. **Reduced Complexity**: the enhancements to the actual infrastructure should not be -as far as possible- complex.

5. **Privacy**: the users should be able to control which property attest their platforms and there should not exist the possibility of obtaining the actual configuration of the platform in any way, neither from the properties nor from any other method.

6. **Non-Discrimination**: verifiers should not have the possibility of favoring selected configurations (principally because the configurations are not a part of the game), so that no platform could be discriminated.

7. **Compatibility**: existing operating systems and the current TCG-compliant hardware should be compatible to the solution.

[43] sealing is a mechanism such that certain data is linked to specific properties (configurations for the *Binaty Attestation* mechanisms) so that it is disclosed only if the corresponding platform fulfills them

## 2.2 Objectives

As it was already depicted in the summary section, the objective of this work is to carry out a study of the viability of combining -as an innovative mechanism- a property-based policy enviroment with the TNC architecture to perform network access control. This approach is in contrast with the TCG approach, that bases these policies on configuration elements of the platforms (such as binary hash values and version, patches or operational status of certain programs among others) and tries to overcome its lacks.

### 2.2.1 Motivation for the use of PBA

There are some encouraging reasons [21, 5, 6] that motivate the rejection of the TCG attestation approach exposing its deficiencies. The following table describes the most important and confronts them with the advantage that would be taken in the case that the PBA approach would be used:

| Binary attestation shortcoming | PBA's point of view |
|---|---|
| **1. Privacy concerns**: since the configuration information is revealed to the third party that requests the platform's state, it makes easier the achievement of an attack. | *Focusing on propeties, the configuration is not disclosed, and therefore the potential attacks cannot take any advantage of it.* |
| **2. Configurations discrimination**: as a consequence of the configuration's disclosure, a potential challenger might be able to discriminate (or favor) certain platforms. | *With the property-based approach, the challenger wouldn't be able to discriminate certain platforms (e.g. specific operating systems) since it does not even know them.* |
| **3. Feasibility**: the number of different configurations (regarding different programs, versions, applied patches or compiler options for example) is cumbersome, making very difficult -if not almost impossible- the adequeate assurance of the security parameters that an endpoint fulfills. | *On the contrary, the number of different properties is much lower, making therefore possible its management.* |
| **4. Configurations uniqueness**: although different platforms could have different configurations, the all may fulfill the same properties, favoring so focusing on the properties. | *If the Verifier focuses on properties, it should be much more easy the definition of the policies, since it has to consider fewer possibilities.* |

| | |
|---|---|
| **5. Inconsistent configurations**: any change performed to the platform's configuration (either related to security aspects or not) will change the binary state of the machine, even if the properties have not been changed. Therefore the process of attesting a platform should be repeated constantly although the important aspects to the challenger (which are the security properties) wouldn't have been changed at all. Another shortcome of this is that any **sealed data**, binded to a configuration could not be accessed after a change to the configuration, having therefore to reseal such data to the new configuration. | *If there is a configuration change with the PBA, the entity within the client which is resposible for the attestation of certain property, will check if such change affects the property it is in charge of. In that case (and only in that case) the assessment should be repeated.* |
| **6. (alternative) Applications discrimination**: since an application vendor could bind data to their own application, it could be impossible for alternative software to access that data (e.g. if a text editor binds its files to be accessed only by itself, any other text editor or reader could not be able to use such documents). For this reason, the sealing of data should be performed only by means of the properties that the machine fulfills, avoiding so the discrimination (or favoring) of certain applications. | *Since the configuration is not disclosed with the property-based approach, the challenger wouldn't be able to discriminate any kind of application, existing therefore only the possibility that the plaforms that don't fulfill certain properties could be discriminated (which seems much more reasonable).* |
| **7.** The knowledge of a platform's configuration does not necessarily imply that the platform complies with the desired security properties. | *On the contrary, the PBA approach is much more expressive, and allows a potential verifier to request more complex sets of information describing the platform than only plain configuration data.* |

### 2.2.2 Proposed solutions for PBA

At the moment, several proposals [21, 9, 5, 16, 6] that suggest how a property-based attestation approach can be realized have been presented. Although these suggestions use elements inherent in the trusted platforms (e.g. the Trusted Platform Module), all of them are, of course, out of the scope of the TNC architecture, otherwise this work wouldn't make sense.

There are mainly two problems to which the proposals offer different points of view when it comes to facing their solution:

- How the properties of a platform are determined. The fact of determining a property within a platform is problematic as a consequence of its own nature: a property is an abstract concept, that, in contrast to a configuration (which is a specific concept), cannot easily be obtained from the platform. This is a motivating challenge of the work: the achievement of the worthwhile advantages that the property-based approach brings within is attached to the difficulty of obtaining such properties.
- Once the properties of the platform have been determined, how to attest them to a third party.

#### Overview

With respect to how the first difficulty regarding the obtaining the properties is overcome, three different approaches can be distinguished [5]:

1. **Code Control**: consists of a (trusted) piece of software within the attestor that enforces a machine to behave as expected. Thus, if there is a property defined that requires a specific behaviour of a platform[44], this *code controller* will be in charge of ensuring that the platform (or its applications) behave as expected.
2. **Code Analysis**: which lies on the endpoint's code analysis by the property attestor itself, extracting from it the properties that the code (and therefore the platform) has within.
3. **Delegation**: there are some factors that have motivated this approach:
   (a) the extraction of the properties by the platform is difficult since the SW/HW performing this action should be very specialized, being able to know which

[44]Note that a property can be seen as a certain behavior of the applications within a platform or the platform itself

properties has to examine, from where and how should it take this information, etc.

(b) the trustworthiness of these results is not easy to achieve since they don't come directly from the platform (in contrast to a configuration for example), fact that would make the process much simplier.

(c) the becoming idea of taking advantage of the current TCG infrastructure based on binary measurements and configurations.

Therefore, most of the proposals suggest a solution based on *delegating* the obtaining of the properties to a **(trusted) third party**, which, by means of a platform's configuration, will derive (or certify) the fulfilled properties of the platform.

### Delegation-based Solutions (Trusted Third Party)

In the following, some interesting solutions based on the *delegation approach* and related to the TCG functionality model will be presented, since they could be helpful in the creation of a solution in the context of the TNC.

In all these solutions, the third party is an entity able to know which properties $(P_i)$[45] are fulfilled by a given configuration $(S_0)$[46] (or check whether a given configuration fulfills certain property or not), and provide afterwards a **digital certificate** $cert(sk_{TTP}; P', S')$ ensuring that the configuration $S'$ fulfills the property $P'$. Having said that, the following suggestions are different implementations to the use of that third party and its issued digital certificates, so that they could prove that they have certain properties because they are provided by their own configurations (and the link between them[47] is certified by a trusted third party) **without disclosing their configuration to the Verifier** at any moment.

### Solutions Extending the TCG Hardware

1. **Extended TPM** [21]: consists in using a TPM with an extended functionality so that its able to perform the following protocol:

   - After having received the necessary data:
     - a demanded property $P_i$ by the Verifier

[45] Where $P$ refers to a property and the subscript $i$ indicates that it can be any of the possible property in which the verifier could be interested.

[46] The existing configuration within the TPM's PCR registers

[47] The link between a configuration and a property expressed within the digital certificate $cert(sk_{TTP}; P', S')$.

  - the public key of a trusted third party (TTP) $pk_{TTP}$
  - a property certificate signed with the TTP's secret key $cert(sk_{TTP}; P', S')$ which links a configuration and a property
  - a *nonce* $r$ provided by the Verifier for freshness[48]

- The extended TPM can prove whether $P_i = P'$ (the demanded property and the property within the certificate are the same) and $S_0 = S'$ (the existing configuration within the TPM's PCR registers and the configuration within the certificate are the same)
- In the case that the previous verifications were positive, the extended TPM would generate and return a PBA certificate signed with its secret key over the property and the nonce $cert_{TPM} := cert(sk_{TPM}; P_i, r)$, proving that the platform fulfills at that moment the desired property.

2. **Group signatures** [21]: a group signatures mechanism allows a member of a group to sign anonymously on behalf of the whole group, so that the TTP generates a unique public key associated to a property and the appropiate group of secret keys binded to the set of configurations that provides the property (one key for each configuration). If the platform has one of the configurations that provide the property, it could be able to sign with one of the secret keys, proving therefore that it fulfills the demanded property. In order to avoid the misuse of one of the keys, attesting for example a wrong configuration, the TTP binds the group signature to the certified configuration.

**Solutions Extending the TCG Software**

1. **Use of a Trusted Attestation Service (TAS)** [21]: in order not to modify the TCG hardware (which could be difficult not only for the cost, but also because it has been already included as a built-in component in several platforms), there are some proposals that suggest modifying the TCG software, so that it may be able to perform the same actions that the extended TPM discussed in the section above. Since a software implementation cannot be easily trusted (in contrast to a hardware implementation, which is more expensive, but trustworthy because it cannot be modified), the software should work over a trustworthy basis.

   Ensuring this requirement in any common operating system is very difficult for several reasons (starting with its complexity and the permissions of the administra-

[48] The Verifier generates a random number ($r$) and sends it to the Attestor, so that it is forced to use it in order to prove that the results are computed in that moment

tor, which allows him to perform almost any change within the system), therefore, there are some difficulties that come along with the software solution.

The only manner to develop a trustworthy software in the platform is building it over a trustworthy basis. Since the operating systems nowadays are too complex, one of the ideas [21] that have been proposed to solve this situation consist in using the PERSEUS architecture, that uses a L4 family microkernel as a basis, as it is exposed in the figure 2.7. Using this architecture, the operating system and the TAS service are used in parallel directly over the microkernel, so that the TAS implementation can be fully trusted.

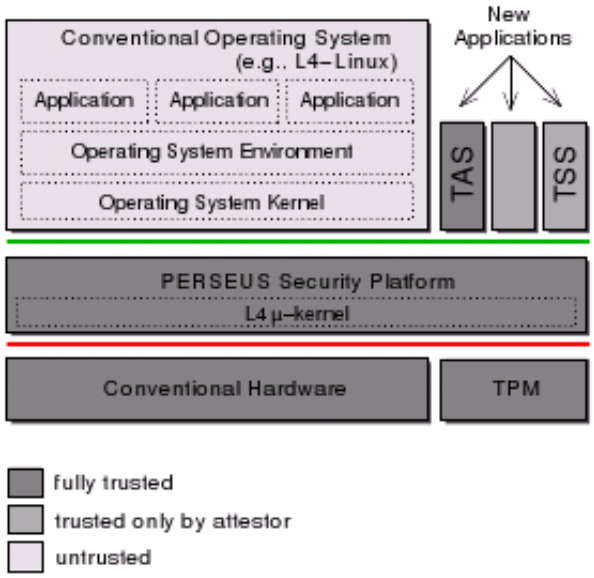



**Figure 2.7:** TAS developed over a trustworthy basis [21]

Once the TAS uses a trustworthy basis, its purpose is performing the tasks that the extended TPM was intended to, but in this case, instead of being the TPM responsible for the communication with the TTP and the verifier, the TAS manages the connection with both and makes use of the TPM for obtaining the securely stored configurations as well as for the use of cryptographic funtionalities in order to gain its attesting and sealing objectives.

2. **Without a TAS** [21]: this solution, that is also based on the use of property certificates, tries to prove by means of *zero-knowledge proofs*[49] that there is a valid

[49] A zero-knowledge proof is such that allows an entity to prove that it possess a digital signature or certificate without disclosing how it looks like

link between the two following signatures (which are secret inputs to the protocol since the Verifier should not learn the platform's configuration):

- $sig_{TPM} := Sign(sk_{TPM}; S_0, r')$, which is the **conventional attestation** signature.
- $sig_{TTP} := Sign(sk_{TTP}; S'_0, P')$, the signature that proves the link between a configuration and a property provided by it.

given $pk_{TPM}$ ,$pk_{TTP}$, the desired property $P$ and a nonce $r$ as an input to the protocol, so that this implementation, since it has be trusted only by the attestor, could be seen as an extension of the TSS, that will help in the verifications below:

- $true \longleftarrow Verify(pk_{TPM}; sig_{TPM})$.
- $true \longleftarrow Verify(pk_{TTP}; sig_{TTP})$.
- $P = P'$; $S_0 = S'_0$; $r = r'$.

To avoid cheating and not to disclose the configurations, some modifications to the protocol are performed: the TPM will not only sign only its configuration $S_0$, but also it will sign its configuration $S_0$ encrypted ($cipher_{pk_{TTP}}(S_0)$) with the public key of a trusted third party ($pk_{TTP}$) (it is $sig_{TPM} := Sign(sk_{TPM}; cipher_{pk_{TTP}}(S_0), r')$, so that the user cannot lie about the underlying configuration. The problem of this solution is that the Verifier cannot verify the enforced property by the user platform online (only the TPM's manufacturer vendor or the court could).

To solve this problem and allow online verification of the enforced property, the use of a *proof of membership* protocol, that, by means of lists of configurations $\{\{S_{1,1}, S_{1,2}, ..., S_{1,n}\}, \{S_{2,1}, S_{2,2}, ..., S_{2,n}\}, ..., \{S_{m,1}, S_{m,2}, ..., S_{m,n}\}\}$ and its corresponding fulfilled properties $\{P_1, P_2, ..., P_m\}$, can prove after performing the conventional attestation protocol in which the signed configuration is hidden, that:

(a) the encrypted cofiguration ($cipher_{pk_{TTP}}(S_0)$) is contained in the list of configurations published by the TTP that provide the property.

(b) the TPM attestation signature is correct.

**Solutions without modifying the TCG HW/SW**

1. **Translation by the bootloader with TrustedGRUB** [16]: this solution proposes a slight modification of the well-known bootloader GRUB so that through the enhancement of the bootstrap process and with the help of an improve of the digital

certificates proposed in the sections above, the boot process could be able to converse the binary measurements that set up the configuration into properties. But the process of extracting the properties is rather simple since it supose to find property certificates directly from the hash value of a file and doesn't consider more complex alternatives when it comes to derive security properties from a configuration.

#### Solutions without a Trusted Third Party

A more recent proposal, out of the classification made above, suggest a new approach with which a Trusted Third Party would not be necessary in order to attest the properties of a platform.

1. **PBA without a TTP** [6]: a new proposal suggest that the use of a trusted third party is not necessary. For doing that, the solution proposes that the Attestor and Verifier agree on a set of configurations that provide certain properties. Therefore the protocol consists in proving that the platform's configuration is one of the set without disclosing it by means of using a ring signatures mechanism, so that the fulfillment of the property can be demonstrated. Although theoretically valid, this solution doesn't fulfill some of the motivating reasons for the use of a property-based approach. Among others, the *Configuration discrimination* requirement should be remarked, since verifier and attestor must agree on a set of configurations, allowing therefore the verifier to demand only certain configurations.

### 2.2.3 Goals of the work

As a result, this work aims to obtain, as far as possible, a slight modification[50] to the *Trusted Network Connect* architecture (whether it be in some of its components, interfaces or protocols) in order to perform a trustworthy property-based attestation without changing the security paradigm. The obtained solution should fulfill, as far as feasible, the requirements that in the sections above were established. Some of these requirements correspond to lacks of the conventional binary attestation (privacy and discrimination concerns for example) and others are characteristic of an attestation mechanism (including efficiency, security or trustworthiness concerns amongst others).

The attestation mechanism proposed should be independent of the attested properties semantics. Which means that the proof that a platform possesses certain properties should be independent of the specific properties being attested, in contrast to the

[50] Whenever it could be possible, it is intended to perform a slight modification concerning only software changes

achievement of the property, which depends strongly on the semantics of the property. For example, obtaining the property *Virus free* should be very different to obtaining the property *No unnecesary open ports* since different files or internal services should be read or called, but the attestation mechanism, once the properties have been obtained, should be the same.

It is out of the scope of these work which specific properties should be obtained and how. Even though, some exemplary properties will be used as a basis to show how the proposed solution works.

# Chapter 3

# Development (Concept, Implementation, Experiments)

# 3.1 Concept

## 3.1.1 Solutions Suitability and Feasibility

**Prior to the development** of a conceptual design and once the PBA idea has been discussed, two questions should be considered:

- Is it advisable to develop a property-based attestation mechanism?
- Is its realization feasible? (in the case that it were advisable)

By having a look to the already commented theoretical advantages that the PBA present in contrast to the conventional binary attestation, an affirmative answer to the **first question** can be easily given: a property-based attestation mechanism is completely advisable. However, the answer to the **second question**, that responds to the actual realization of such mechanism, is much more difficult since it raises certain issues that should be solved. It is worth noting that although it has been demonstrated that a property-based attestation mechanism can be performed [21,9,5,16,6], all the approaches are out of the TNC scope and some of then, even if theoretically adequate, have several shortcomes.

Therefore at this point, we can ensure that the solution to the problem is possible, but each solution, depending on its conceptual basis and implementation, will:

1. Fulfill different desirable requirements.
2. Be more (or less) adequate on its integration with the TNC architecture[1].

## 3.1.2 Questions to be Considered

### General Questions

**When it comes to develop** an attestation mechanism, there are some questions that must be considered:

1. Which configuration elements should be measured?
2. How, by whom and when should be those configuration elements evaluated?
3. How can the trustworthiness of the collected data ensured?

[1] This level of adequateness refers to the possibility that the solution would fit in the context of TNC by means of its potential use of the vast majority of its elements and its working manner or not (for example, the use of a Trusted Third Party to obtain links between configurations and properties is, although valid, is not typical for the TNC)

4. Once received the data and checked its trustworthiness, how can the client configurations be evaluated? (proof that the platform fulfill the requirements)

5. How can be ensured that the configuration does not change?

This set of questions to consider is slightly modified if the attestation mechanism to develop is based on properties, since not only the obtention of a property from a platform, as it was already commented in previous chapters, is much more difficult than the obtention of a configuration, but also the ensurement of its trustworthiness[2] is as well more difficult to obtain. The new set of questions is the following:

1. Which properties should be considered?

2. How, by whom and when should be those properties evaluated?

3. How can the trustworthiness of the collected data ensured?

4. How can the verifier evaluate the platform's properties? (proof that the platform fulfill the requirements)

5. How can be ensured that the properties do not change?

#### Combination of PBA and the conventional TCG Binary Attestation

Besides these questions and prior to its resolution, another matter regarding the combination of both attestation approaches should be discussed. Some proposals suggest that both, the property-based and the binary approaches, can be combined in order to obtain the best of both worlds.

For example the proposal on [9] makes use of them. Thereby, the process starts with a secure boot and attests by binary attestation a Java Virtual Machine which is used as a basis [7] to analyze the code and the flow execution (by means of capturing and monitoring calls and traffic like the use of sockets for example) of certain programs. Finally, this JVM will check if the behavior of the programs corresponds to the stablished policies based on properties. In other words, the use of the conventional binary attestation could be used to provide an adequate (trusted) basis to attest certain applications that could perform the property-based attestation itself. This idea of combining both suggestions as in a **two level chain of attestations**, although not satisfying every requirement since it demands the obtaining of binary measurements and therefore the possession of specific

[2]Understood as a proof that the extracted property is actually fulfilled by the platform

configurations, could be of help in the development of a solution.

**Another approach** could be the actual combination at the same time (in contrast to the two level chain) of both attestation paradigms. In this way, there will be some entities in charge of attesting the worthwhile properties whereas other entities would attest the configuration concerns (refering to the `Attributes` of the `Software Products` that are of certain `Component Types`). This model has the advantage that the verifier can combine the expressiveness that the properties confers and the verfication of certain configuration attributes in order to obtain useful information about the platform. As a disadvantage, the configuration of the platform is exposed, thus the privacy and the non-discrimination requirements would be not fulfilled.

In short, although the combination of both approaches is perfectly valid, there are some requirements regarding privacy and non-discrimination concerns which this solution will not comply with. Consequently, dependind on the reasons that motivate the development and use of a property-based attestation mechanism, the combination of both should be advisable or not.

### 3.1.3 Different Alternatives to the Solution

Being aware of the following points:

- Feasibility and suitability of the PBA approach,
- the requirements that an infrastructure capable of performing PBA should possess commented in the section 2.1.4 and the overcome lacks of the TCG conventional binary attestation that a PBA approach offers detailed in the section 2.2.1 that can be also seen as requirements of the solution,
- the issues expressed in the section above 3.1.2 that every solution should solve,
- the possible combination of the PBA with the conventional TCG binary attestation,
- and last but not least, the different solutions that have already been proposed to implement PBA,

a classification of different alternatives to the solution of the problem can be in the following presented. Every solution entails advantages and disadvantages regarding the efficiency, complexity of its practical realization, integration within the TNC architecture and fulfillment of the requirements among others aspects.

The main element that has influence in the classification of the solutions is the way the properties are obtained. Thus, there are basically two manners of obtaining the properties within a platform. The first one consists in using a third party to perform the *translation* between configurations and properties. The second one consists in obtaining the properties inside of the platform itself, without the help of any other party.

#### Solutions with a Trusted Third Party (TTP)

The main feature of this idea is the use of a Trusted Third Party that is in charge of **obtaining properties by using platform configurations** as a base. Afterwards, a mechanism should be used in order to prove the relation between the platform and the properties i.e. proving that the platform has certain configuration and the valid link between the configuration and the property[3]. Besides the proof of the valid link between the platform and the property, the mechanism should be **designed not to disclose the configuration** of the platform to the verifier at any moment.

In this way, the following relation gives an indication of the idea explained above:

$$Platform \overset{[TPM]}{\longleftrightarrow} Configuration(C_i) \overset{TTP}{\longleftrightarrow} Property(P_i)$$

The platform has a configuration, which can optionally be ensured with the help of a Trusted Platform Module, and the TTP links configurations and properties.

This approach has as an **advantage** that demands performing few changes in the model but requires, as a **shortcome**, the management of digital certificates. The use of the digital certificates to link configurations and properties provided by them entails two problems:

1. **Management of digital certificates**: depending on the implementation:
   - The management of the certificates could imply connections to a third party at the moment of the connection to a network (as a consequence of the requirement of certain property to which the platform does not have an appropiate certificate) which could be terribly inefficient.
   - The management of the certificates could also imply the previous obtention of digital certificates to prove the link between different properties and the current platform's configuration, that should be obtained again if the configuration of the machine changes.

[3]Generally obtained through a digital certificate issued by the TTP

2. **Dependence on Configurations**: although the verification of the attestor is eventually performed by the verifier and based only on properties, the whole process still depends on configurations. This fact involves that the points 2,4,5 and 7 of the section 2.2.1 wouldn't be totally fulfilled by this approach. Some examples of this are that:

   - a third party in charge of linking configurations and properties should be aware of every possible configuration, which seems impossible as a consequence of the cumbersome number of possible combinations and consequently some configurations would be discriminated.
   - Any change to a configuration element, even if it is not related to any interesting property, would force the new obtaining of the digital cerfiticates.
   - The number of possible configurations is still cumbersome and, although the problem of extracting properties from configuration has been moved to a third party specialized in such task, it is still very difficult to obtain real properties only by means of configurations. This difficulty responds to the huge number of possible configurations combination and to the absence expressiveness of the configurations (as it was said before, the knowledge [5] of a platform's configuration does not necessarily implies that the platform complies with the desired security properties). Therefore, focusing on configurations maybe is not the best advisable approach.

Being the configuration securely (and trusty) stored within the TPM or not, it is needed to have in any case an intermediary[4] between the Attestor platform, the Trusted Third Party and the Verifier. This mediator could be carried out by one of the entities proposed in the next points.

**Hardware Modification** There are already some proposals [21] that suggest an extension to the *fully trusted*[5]

[4] An intermediary understood as an entitiy in charge of performing all the tasks formulated above including the communication with the TTP and the management of the certificates among others

[5] It is worth remining that [21]:

- **Fully trusted**: can violate the security requirements of both attestor and verifier.
- **Only trusted by the attestor/verifier**: can only violate the security requirements of one party.
- **Untrusted**: All other components or entities.

Trusted Platform Module so that it could be able to manage the certificates, the configurations and the attestation of the properties. Although this solution is theoretically valid, it doesn't not comply with the requirement 4 of the section 2.1.4 regarding to the complexity of the solution. Moreover, it doesn't worth to invest efforts in developing a complex solution that could not easily be integrated in the context of the TNC, since it uses the hardware to perform the main tasks instead of using the components within the TNC architecture (that are situated in higher software layers). Thus, can be established, that a hardware major based solution to the problem of performing a property-based attestation in the context of the TNC is not advisable.

**Software Modification** On broad lines, with the TTP approach, the task of obtaining the properties within a platform is distanced from the platform itself. Thus, the basis of this model is the use of an extern entity, which is in charge of evaluating the properties that the platform has, to finally prove them with its help to a potential challenger. The only thing that the system can provide this Trusted Third Party to evaluate its properties, is the configuration data, so that the TTP will act as a *mapper* of configurations into properties.

As a consequence of the use of a TTP to perform this conversion, most of the TNC technology (in the form of several IMC/IMV pairs) that could be associated to the extraction of properties inside the platform would be deprecated. This fact would provoke a much more simplier set of *Collectors* and *Evaluators* in the top layer of the TNC architecture, since they will have just to obtain the configuration within the machine and transfer it to a third party and afterwards evaluate it is results.

Moreover, as a result of the implantation of this proposal, the consequent simplicity of the *Integrity Measurement Layer* suggests that this approach could be combined with the TNC attestation model, obtaining therefore the best of both worlds, taking therefore advantage of the power and expressiveness of the properties when possible and desired. This model would have consiquently a set of IMC/IMV pairs attesting configuration elements and other managing the connection to the TTP and attesting the properties.

Despite the possible combination of approaches, the solution regarding the properties will guarantee the preservation of the whole configuration, not disclosing it to the challenger at any moment. With respect to the possible IMC/IMV pairs performing the conventional TNC attestation, the privacy concerns will depend on local policies, so that the user could establish the configuration elements he wants to disclose.

In the following, different alternatives to perform a property-based attestation integrated within the TNC architecture by means of using a Trusted Third Party are considered:

1. **Version 1: SimpleAttestor**

   The background idea of this version is supported by the use of a simple IMC that will act as an intermediary between the IMV and the TPM, making some processing and forwarding the TPM some data provided by the IMV, so that the Access Requestor (AR) could provide its configuration signed and encrypted (making use of a small cryptographic trick that will be explained in the following) allowing the Policy Decision Point (PDP) to check the reliability of the data, but without being able to inspect its contents. Finally the IMV will be in charge of the communication with the TTP by making use of the data provided by the Access Requestor, so that it could obtain the properties fulfilled by its configuration.

   To clarify the already presented general idea of the solution, the figure 3.1 schematize graphically the participating entities and the flow of messages. Finally, the protocol will chronologically explained, including the message flows and the data processing.

   An important aspect of the solution is that the communication between the $IMC_p$ and the $IMV_p$ is performed through the standardized interface IF-M. This interface, as it was already introduced in the section 2.1.2, is an attempt to deploy an application level protocol [31] able to carry standardized Integrity Check Handshake messages between a variety of IMCs and IMVs from different vendors. This interface provides therefore a framework for the communication between IMC and IMV pairs allowing them to request measurement of *Component Attributes* as well as returning the value of these measurements. As it was also mentioned, this protocol in the *Integrity Measurement Layer (IML)* has a set of standardized attibutes. Since it has been design with a focus on the existing attestation paradigm, most of those standardized attibutes [31] are product-oriented (e.g. `Product Information`, `Operational Status`, `Numeric Version`, `String Version` and so on), therefore vast majority of them won't be used in these solutions. Moreover, the protocol allows the definition of own attributes by using a different `Vendor ID` (Attribute Type

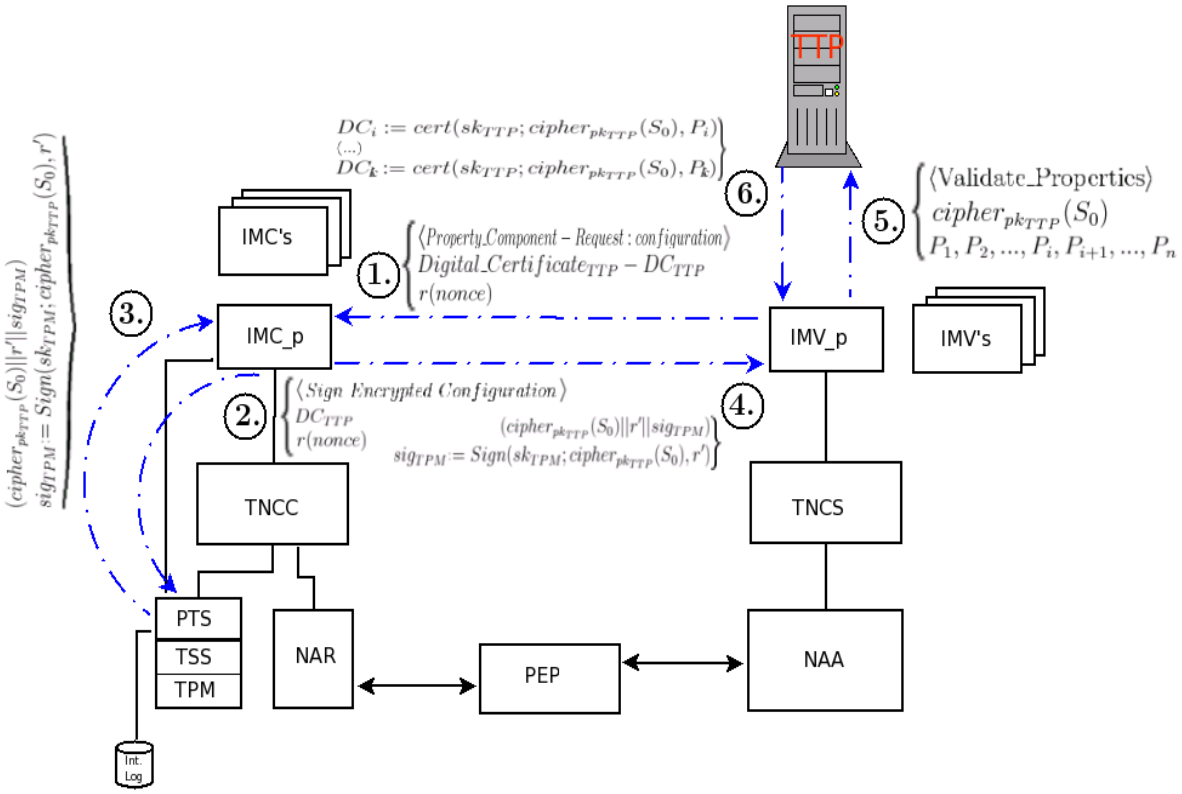



**Figure 3.1:** PBA within the TNC by means of using a TTP - Version 1

Space) of the TCG SMI (0x005597). In this way, the already registered FHH's SMI Private Enterprise Number (PEN) 0x0080ab could be used to define new property-related attributes and combined with the TCG SMI PEN to perform the necessary actions within the solution.

The chronological steps and procedures of the solution regarding the figure 3.1 are explained in the following:

1 The process starts with the $IMV_p$ requesting for the *Access Requestor*'s configuration that it needs in order to obtain the properties with the help of a TTP. Adding to this request, the $IMV_p$ attachs a set of admitted TTPs by the PDP by means of a *digital certificate*[6] for each admitted TTP as well as a nonce $r$ for

[6]A digital certificate(DC) [35] is an electronic document that binds an identity to a pair of electronic keys that can be used to encrypt and sign digital information. A DC is issued by a Certification Authority (CA) and its signed with the CA's private key (so that the rightness of the DC's contents can be proven. In this case, every TTP certificate will (at least) contain:

- TTP's public key $pk_{TTP}$.

freshness.

The objetive of sending this TTP's digital certificates to the AR is that the platform's configuration that will afterwards be sign by the TPM and sent to the PDP should not be readable by the verifier in order to agree with the privacy and non-discrimination concerns established in the requirements of the sections 2.1.4 and 2.2.1. In this way, after having proved the integrity of the selected TTP digital certificate and therefore its actual identity, the TPM will be able to sign, instead of the configuration of the platform itself, the configuration encrypted with the public key of a TTP. Therefore, even having access to the encrypted configuration, the PDP will not be able to read its actual contents, being only the TTP able to do so. It is worth noting that the TTP will issue certificates linking Properties and Encrypted Configurations, so that the $IMV_p$ will be able to prove if the certified encrypted configuration is the same that the one that was sent by the AR.

2 The $IMC_p$ demands the TTP the encryption of the configuration with the TTP public key $pk_{TTP}$ and its consequent signature with its secret key ($sk_{TPM}$).

3 The TPM provides the $IMC_p$ the block of information $(cipher_{pk_{TTP}}(S_0)||r'||sig_{TPM})$, which is composed by the following data:

- $cipher_{pk_{TTP}}(S_0)$: the platform's configuration encrypted with the public key of the TTP
- $r'$: the nonce (is marked as $r'$ instead of $r$ because it is something that the AR could lie about. The $IMV_p$ will have to check afterwards if the sent nonce $r$ and the received nonce $r'$ are equal.
- $sig_{TPM} := Sign(sk_{TPM}; cipher_{pk_{TTP}}(S_0), r')$: basically, the signature of both elements above.

4 The $IMC_p$ forwards the received information to the $IMV_p$.

- TTP Name.
- Expiration date of the public key.
- Name of the CA that issues the DC.
- Serial Number of the DC.
- Digital Signature of the issuer (to check the rightness of the contents).

5 After checking the received information, the $IMV_p$ the platform's encrypted configuration ($cipher_{pk_{TTP}}(S_0)$ as well as a set of demanded properties $P_1, P_2, ..., P_i, P_{i+1}, ..., P_n$ to the TTP, so that the TTP will check, once it has decrypted the encrypted configuration, which of the demanded properties are fulfilled.

6 The TTP sends a list of fulfilled properties by the encrypted configuration in the form of a set of issued digital certificates. Thus, the $IMV_p$ would be able to prove the links:

$Platform \overset{[TPM]}{\longleftrightarrow} Encrypted\_Configuration(cipher_{pk_{TTP}}(S_0)) \overset{DC_{TTP}}{\longleftrightarrow} Property(P_i)$

And therefore the link: $Platform \overset{fulfills}{\longleftrightarrow} Property(P_i)$

There are some important considerations regarding the use of the IF-M interface to communicate IMC/IMV pairs and the use of its `Component types` and `Attributes` combined with the idea of the properties.

Since the approach of the Property-Based Attestation has not yet been considered within the context of the TNC, the IF-M is, as it was already said before, is mainly focused on products. In this way, in order to define new attributes regarding properties, the FHH's SMI Private Enterprise Number (PEN) 0x0080ab will be used as `IF-M Vendor ID` and the equivalences in figure 3.2 will be made.

In figure 3.2 is shown how the conventional `Component types` like an Antivirus or a Firewall IF-M subtypes are compared to the Property Managers (at the moment only the Version 1 has been explained, but the graphic contains also some concepts regarding the PBA attestation model version 3, that will be explained later on). Similarly, the `Attributes` of the conventional attestation such as Product Information or String Version are replaced (or complimented) with Properties. More over, the IF-M is able to deal with the existence of different `Products` of the same `Component type`, so that an unique IMC in charge of managing certain type of product could be able to attest both of them by means of activating the optional `Correlation ID` flag (and therefore making use of the correspondent field). In this way, with the PBA approach it will be allowed (by using the correlation ID) to have different underlying ways to obtain the properties that could work in parallel and the IMC responsible for attesting those properties could be able to identify them.

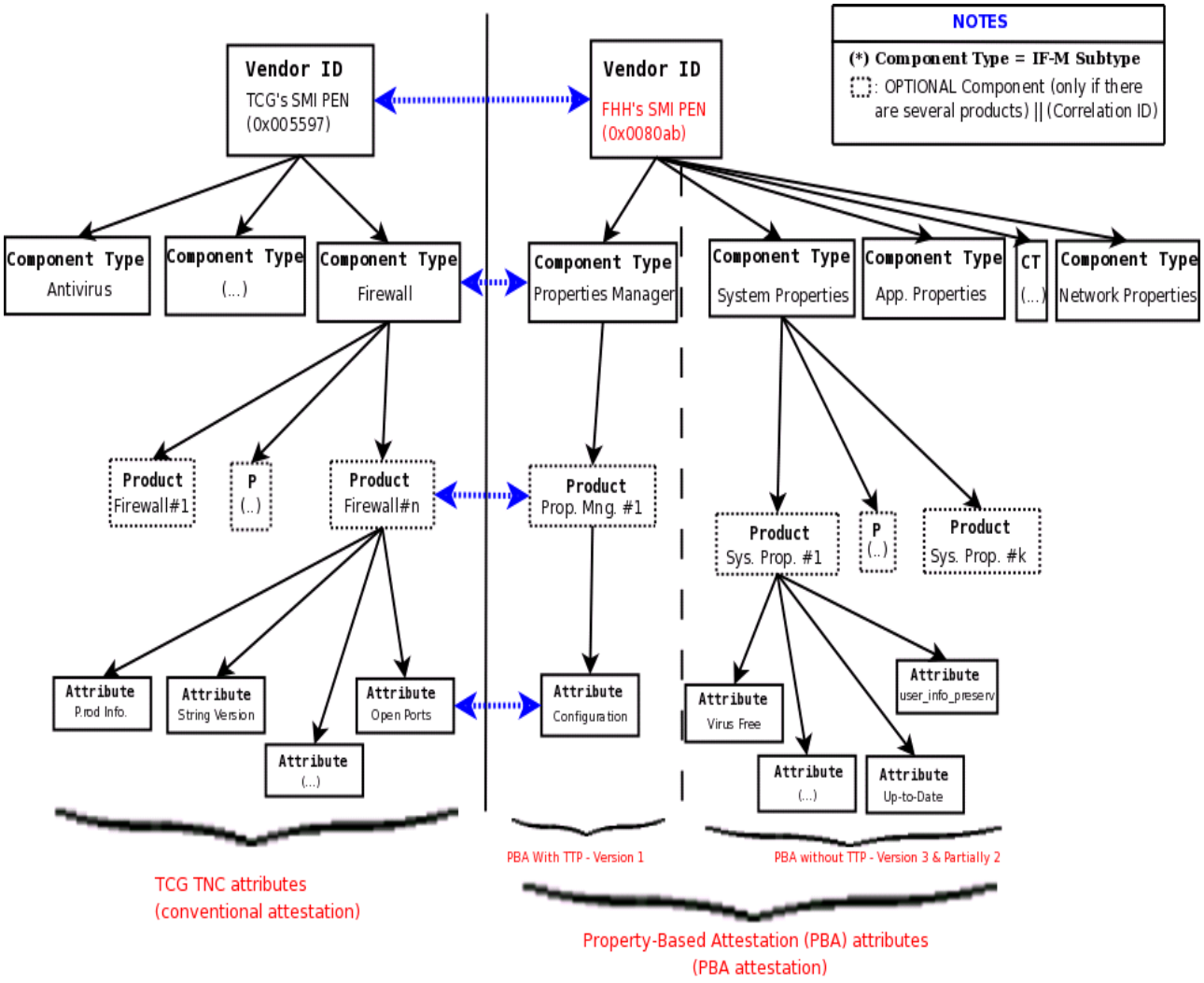



**Figure 3.2:** Use of IF-M attributes to define properties (equivalences)

To clarify how exactly the communication between the IMC/IMV pairs work by means of using IF-M Messages that are encapsulated within IF-TNCCS Messages, in figure 3.3 an exemplary message from a $IMV_p$ to an $IMC_p$ corresponding to the communication flow 1 described above, in which the "attribute" configuration of the machine is requested.

In figure 3.3 it is shown how the IF-M messages are sent from PDP's side by the $IMV_p$ in the top level of the TNC architecture to the $IMC_p$ in the AR's side and how this messages are encapsulated within the IF-TNCCS messages that the TNC Client and TNC Server send to each other. In this case, a batch of IF-TNCCS messages, which is only filled with one IF-TNCCS message of type IF-M, that in

turn, contains exactly one IF-M message, is used to perform the communication.

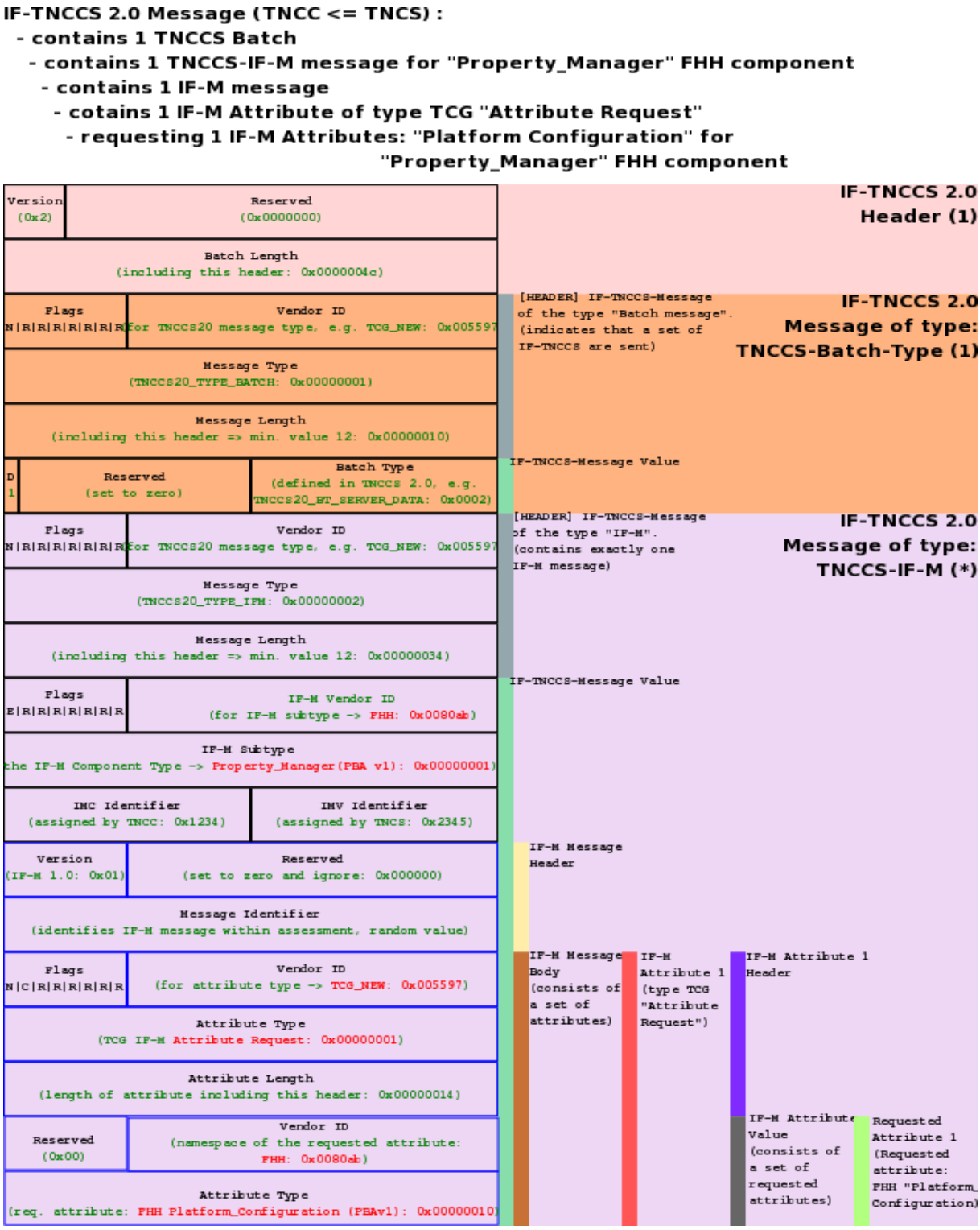



**Figure 3.3:** Message $IMV_p \longrightarrow IMC_p$ requesting the attribute `Platform_Configuration` for Property-Based Attestation (PBA) with a Trusted Third Party (TTP) - Version 1

It is worth noting that inside this IF-TNCCS IF-M message, the `IF-M Subtype` (also known as `Component Type`) must be indicated for routing purposes. This is, that the TNC Client must be aware of the Component Type which the messages are sent to in order to be able to identify the IMCs that should receive the IF-M message that it contains.

With reference to the IF-M message itself, it is important highlighting that it consists of a set of Attributes. In this case, only one attribute (of type TCG `Request Attribute`) is sent. This attribute consists of a set of requested attributes, but in this case, only one attribute of type FHH `Platform_Configuration` is requested.

Paying special attention to the privacy and non-discrimination concerns, one may consider that the PDP could be able to cheat the AR if it registers itself as a TTP and afterwards it requests the AR its configuration (as PDP) to perform PBA. Sending within the request its own identity as TTP, for the AR to encrypt the configuration with its public key, being able eventually to decrypt the data and therefore access the platforms configuration. The solution to this problem could consist in maintaining a list of accepted TTP's within the client, so that the PDP could (as much) be able to force the AR to use one of them. With this approach, the public keys of the admitted TTP would be stored in the AR, not being necessary for the PDP to transfer them.

### 2. Version 2: ComplexAttestor

This version consist in a slight modification to the idea explained in the version 1 consisting in a simple IMC property attestor within the AR. In the mentioned version, the only functionality of this attestor was collecting the platforms configuration and sending it to the PDP, which would be in charge of forwarding it to the TTP to obtain the properties of the configuration.

Although in this version most of the process also turns around an IMC/IMV pair which is responsible for the management of the properties, the so called $IMC_{p_2}$ is much more complex than its predecessor $IMC_p$, being in this case the entity that establishes the communication with the TTP. Said that, the reader can easily imagine the pertinent changes that should be made with respect to the Version 1. For example and amongst others, the AR must be conscious of the properties the

PDP is interested in.

The figure 3.4 represents, from a high level point of view, the interaction between the entities in order to perform a property-based attestation with a Trusted Third Party (TTP) and a more complex IMC than the one presented above.

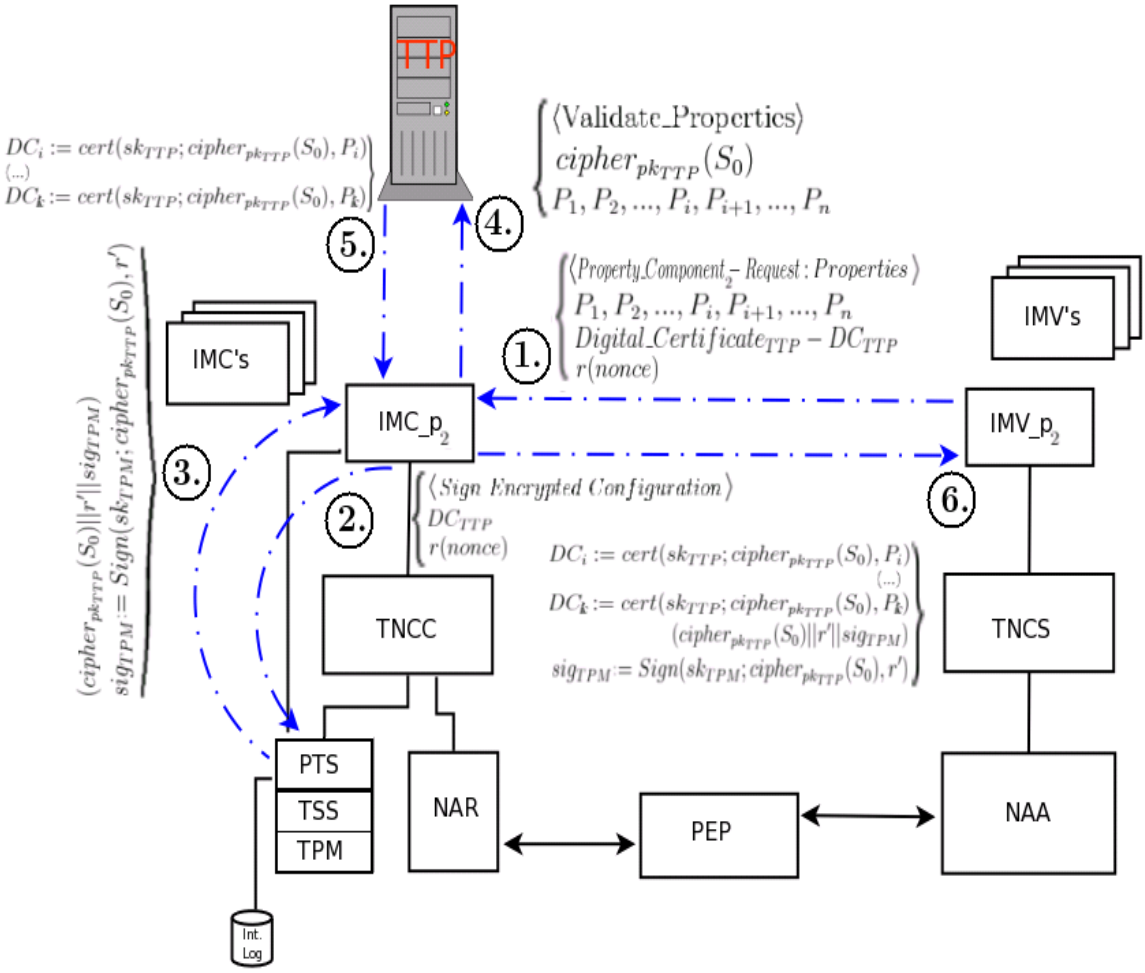



**Figure 3.4:** PBA within the TNC by means of using a TTP - Version 2

The chronological steps and procedures of the solution regarding to the figure 3.4 are explained in the following:

1 The process starts with the $IMV_{p_2}$ requesting by means of IF-M messages the $IMC_{p_2}$ within the *Access Requestor* to prove whether the platform fulfills a set of properties $P_1, P_2, ..., P_i, P_{i+1}, ..., P_n$ or not. Attached to this request, the $IMV_{p_2}$

sends also a set of admitted TTPs by the PDP by means of a *digital certificate* for each admitted TTP as well as a nonce $r$ for freshness.

2 The $IMC_{p_2}$ demands the TTP the encryption of the configuration with the TTP public key ($pk_{TTP}$) and its consequent signature with its secret key ($sk_{TPM}$), forwarding it the nonce $r$ provided by the $IMV_{p_2}$ that should also be signed.

3 After executing the requested operations, the TPM provides the $IMC_{p_2}$ the block of information ($cipher_{pk_{TTP}}(S_0)||r'||sig_{TPM}$), which is composed by the following data:

- $cipher_{pk_{TTP}}(S_0)$: the platform's configuration encrypted with the public key of the TTP
- $r'$: the nonce (is marked as $r'$ instead of $r$ because it is something that the AR could lie about. The $IMV_{p_2}$ will have to check afterwards if the sent nonce $r$ and the received nonce $r'$ are equal.
- $sig_{TPM} := Sign(sk_{TPM}; cipher_{pk_{TTP}}(S_0), r')$: basically, the signature of both elements above.

The $IMC_{p_2}$ could be at this moment able to prove the PDP that the platform has the configuration $cipher_{pk_{TTP}}(S_0)$ (which is encrypted with the public key of a Trusted Third Party), since its signed by the TPM.

4 After receiving the information, the $IMC_{p_2}$ checks if it is in possession of some (offline) Digital Certificates (obtained in previous handshakes) for any of the requested properties $P_1, ..., P_n$ corresponding to the ciphered configuration of the machine ($cipher_{pk_{TTP}}(S_0)$). For every property for which the AR is not in possesion of a digital certificate, the $IMC_{p_2}$ sends to the TTP the platform's encrypted configuration ($cipher_{pk_{TTP}}(S_0)$ along with this set of demanded properties $P_l, ..., P_m$, so that the TTP could check, once it has decrypted with its own private key the encrypted configuration, which of the demanded properties are fulfilled.

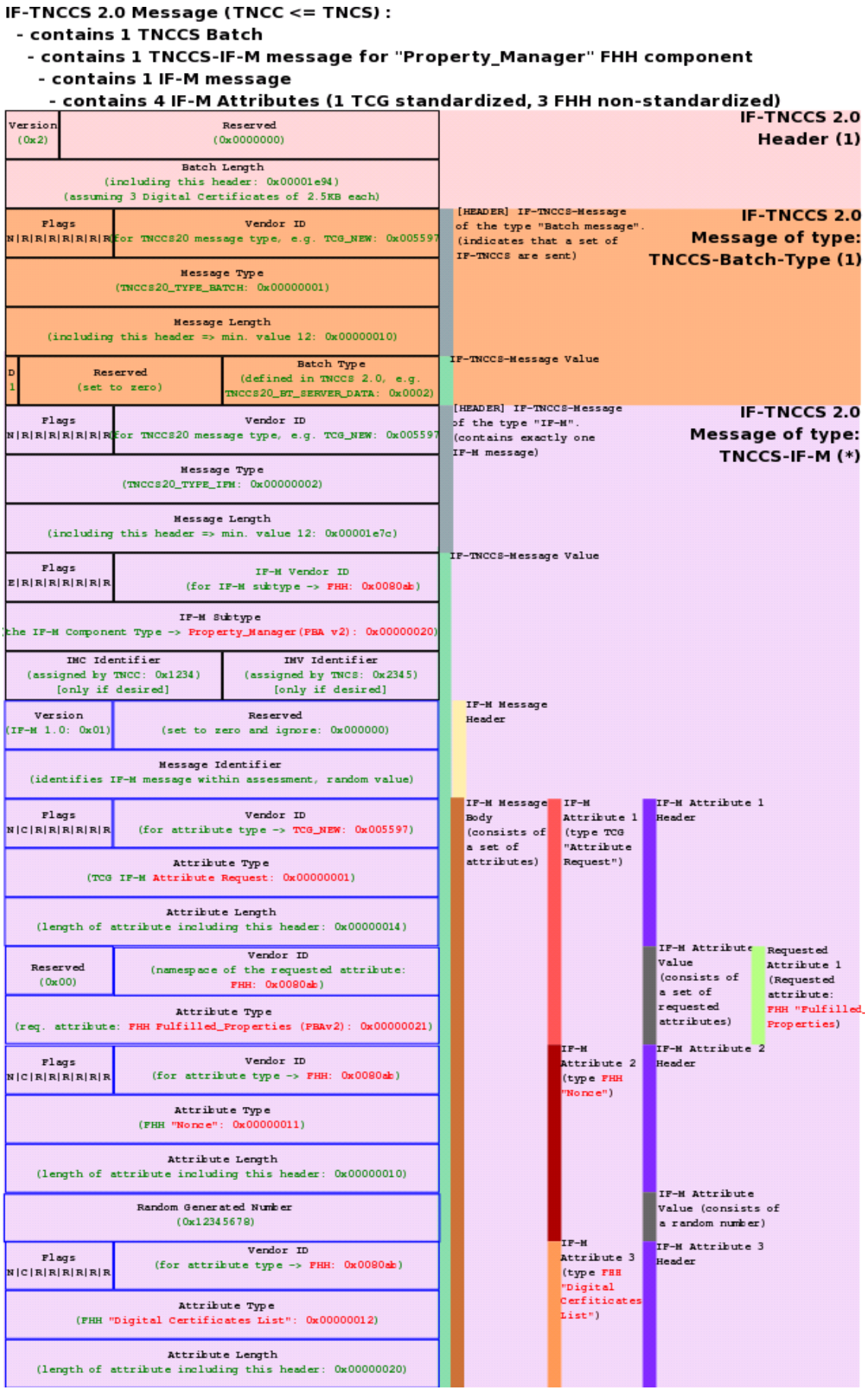
5 The TTP sends to the $IMC_{p_2}$ a list of fulfilled properties $P_j, ..., P_k \in P_l, ..., P_m$ by the encrypted configuration in the form of a set of issued digital certificates issues by itself, which afterwards, will be sent to the PDP.

6 At this point, the $IMC_{p_2}$ is in possession of the following elements:

- The proof that the platform possess certain configuration (encrypted with the public key of a TTP):
  $(cipher_{pk_{TTP}}(S_0)||r'||sig_{TPM})$, where $sig_{TPM} := Sign(sk_{TPM}; cipher_{pk_{TTP}}(S_0), r')$
- The proof that a configuration (encrypted with the public key of a TTP) fulfills certain properties:
  $DC_j := cert(sk_{TTP}; cipher_{pk_{TTP}}(S_0), P_j)$, ..., $DC_k := cert(sk_{TTP}; cipher_{pk_{TTP}}(S_0), P_k)$

This information will be forwarded by the $IMC_{p_2}$ to the $IMV_{p_2}$ through the IF-M interface by means of IF-M messages. Finally the PDP will check the received information and will decide, comparing the results with its local access policies, the approval, deny or remediation of the requesting plaform.

Obviating the concerns regarding the client-server protocol between the IMC and the TTP that should be implemented from scratch, in the following in figure 3.5 and as an example, the IF-M message transmitted from the $IMV_{p_2}$ to the $IMC_{p_2}$ is shown in detail. Within this message, the $IMV_{p_2}$ requests the AR to prove that its configuration fulfills a set of properties. To perform this request, the $IMV_{p_2}$ makes use of the standardized attribute `Attribute_Request` asking the $IMC_{p_2}$ for the FHH attribute `Fulfilled_Properties`. Once received this attribute, the $IMC_{p_2}$ will understand that the verification of a list of properties is needed, and it will be conscious of the existence of the rest of elements attached within the message in the form of attributes (i.e. the nonce, the set of digital certiticates corresponding to the admitted TTP's and the list of preperties to be attested).


IF-TNCCS 2.0 Message (TNCC <= TNCS) :
- contains 1 TNCCS Batch
- contains 1 TNCCS-IF-M message for "Property_Manager" FHH component
- contains 1 IF-M message
- contains 4 IF-M Attributes (1 TCG standardized, 3 FHH non-standardized)
IF-TNCCS 2.0 Header (1)
IF-TNCCS 2.0 Message of type: TNCCS-Batch-Type (1)
IF-TNCCS 2.0 Message of type: TNCCS-IF-M (*)
Version (0x2)
Reserved (0x0000000)
Batch Length (including this header: 0x00001e94) (assuming 3 Digital Certificates of 2.5KB each)
Flags N|R|R|R|R|R|R|R
Vendor ID for TNCCS20 message type, e.g. TCG_NEW: 0x005597
Message Type (TNCCS20_TYPE_BATCH: 0x00000001)
Message Length (including this header => min. value 12: 0x00000010)
D 1
Reserved (set to zero)
Batch Type (defined in TNCCS 2.0, e.g. TNCCS20_BT_SERVER_DATA: 0x0002)
[HEADER] IF-TNCCS-Message of the type "Batch message". (indicates that a set of IF-TNCCS are sent)
IF-TNCCS-Message Value
Flags N|R|R|R|R|R|R|R
Vendor ID for TNCCS20 message type, e.g. TCG_NEW: 0x005597
Message Type (TNCCS20_TYPE_IFM: 0x00000002)
Message Length (including this header => min. value 12: 0x00001e7c)
[HEADER] IF-TNCCS-Message of the type "IF-M". (contains exactly one IF-M message)
IF-TNCCS-Message Value
Flags E|R|R|R|R|R|R|R
IF-M Vendor ID (for IF-M subtype -> FHH: 0x0080ab)
IF-M Subtype (the IF-M Component Type -> Property_Manager (PBA v2): 0x00000020)
IMC Identifier (assigned by TNCC: 0x1234) [only if desired]
IMV Identifier (assigned by TNCS: 0x2345) [only if desired]
Version (IF-M 1.0: 0x01)
Reserved (set to zero and ignore: 0x000000)
Message Identifier (identifies IF-M message within assessment, random value)
IF-M Message Header
Flags N|C|R|R|R|R|R|R
Vendor ID (for attribute type -> TCG_NEW: 0x005597)
Attribute Type (TCG IF-M Attribute Request: 0x00000001)
Attribute Length (length of attribute including this header: 0x00000014)
IF-M Message Body (consists of a set of attributes)
IF-M Attribute 1 (type TCG "Attribute Request")
IF-M Attribute 1 Header
Reserved (0x00)
Vendor ID (namespace of the requested attribute: FHH: 0x0080ab)
Attribute Type (req. attribute: FHH Fulfilled_Properties (PBAv2): 0x00000021)
IF-M Attribute Value (consists of a set of requested attributes)
Requested Attribute 1 (Requested attribute: FHH "Fulfilled_Properties)
Flags N|C|R|R|R|R|R|R
Vendor ID (for attribute type -> FHH: 0x0080ab)
Attribute Type (FHH "Nonce": 0x00000011)
Attribute Length (length of attribute including this header: 0x00000010)
IF-M Attribute 2 (type FHH "Nonce")
IF-M Attribute 2 Header
Random Generated Number (0x12345678)
IF-M Attribute Value (consists of a random number)
Flags N|C|R|R|R|R|R|R
Vendor ID (for attribute type -> FHH: 0x0080ab)
Attribute Type (FHH "Digital Certificates List": 0x00000012)
Attribute Length (length of attribute including this header: 0x00000020)
IF-M Attribute 3 (type FHH "Digital Cerfiticates List")
IF-M Attribute 3 Header

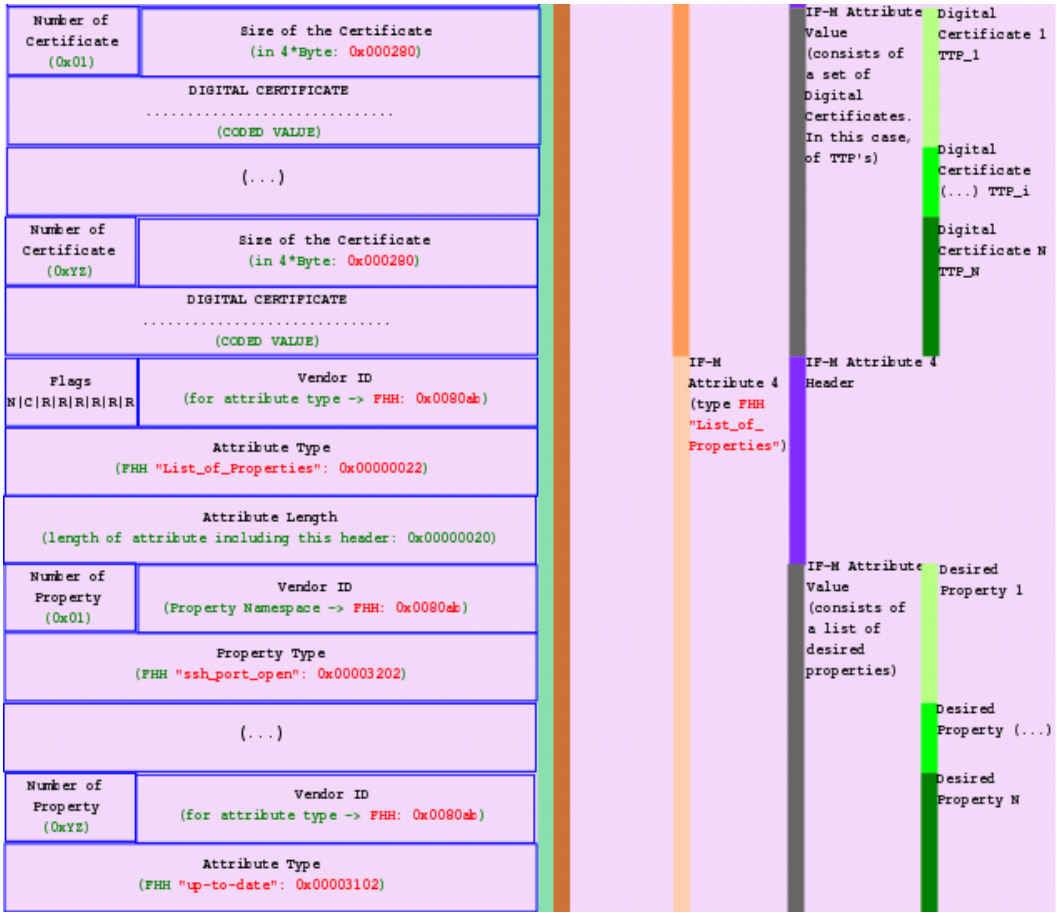



**Figure 3.5:** Message $IMV_{p_2} \longrightarrow IMC_{p_2}$ requesting the attribute `Fulfilled_Properties` and providing it the necessary information (a nonce, a set of TTP as well as the list of properties to be checked) for Property-Based Attestation (PBA) with a Trusted Third Party (TTP) - Version 2

#### Solutions without a Trusted Third Party (TTP)

Although valid, the suggestions proposed in the section above may not be considered as the best approaches to perform a property-based attestation whether within the contest of TNC or not. The reason that motivates this fact is that there are some intrinsic elements pertaining to these solutions that lack the potential of the approach amongst which can be highlighted:

- Firstly and as a **general issue** that affects all the solutions that use that model: the dependence on Trusted Third Parties in the obtention of the properties, which supposes the management of digital certificates and therefore the (regardless) dependence on configurations.
- Secondly and as a **paticular issue** regarding the integration of the Property-Based Attestation with the TNC architecture: since the presented PBA approaches are based on the use of a third party, the obtaining of the properties is moved further the platform itself. Consequently, the AR's sofware components in form of IMC/IMV pairs are practically nonexistent, making no use of the vast majority of the TNC elements and taking therefore no advantage of it. As already mentioned, this simplicity suggest that these solutions could be combined with the conventional TCG attestation or even with this version.

### Version 3: Multiple IMC/IMV pairs

In view of the foregoing, arises the need of developing an approach that would take advantage of the TNC architectural elements as well as overcome the lacks that are a consequence of the TTP's use.

The general idea of this proposal consists in the use of a set of IMC/IMV pairs, each of them specialized in certain type of properties (for example `System_Properties`, `Network_Properties` amongst others). Thus, using the NAC policy within the PDP, the IMV will ask the corresponding IMC for the properties the PDP is interested in. In order to attest a property, each IMC, as an expert in such kind of properties, will execute the necessary procedures to calculate them to finally send the results to its IMV pair. The interface IF-M, that provides an adequate framework to permit the communication between Integrity Measurement Collectors and Verifiers, will be used in the design of the

solution since it allows the existence of self-defined attributes.

Although this modelation of the solution to the problem seems much more promising than the offered before because it has several advantages regarding the paradigm of determining properties and it doesn't need the help (and the connection) of any other parties, it also has certain shortcomes. The following list of disadvantages indicates some of these shortcomes:

- The implementation requires a bigger effort since the process of obtaining the properties is performed within the platform.
- The properties are obtained by IMCs without any help of the TPM (for the moment), so that the trustworthiness could be jeopardized.
- The fact of implementing each IMC to be able to analyze properties is particularly expensive[7] because every procedure that obtains a property should be written as a consequence of the properties specificity. That means that every property is different, and therefore the way to obtain them, too.
- The difficulty of elaborating a complete properties portfolio along with the manner to obtain all of them.

Once the general concept of this version have been introduced in the paragraphs above, a deeper explanation can be given. The background idea is that there are some software entities that are specialized in the obtention of a set of field-related properties within a machine. With the conventional TCG attestation model, the IMCs are software entities, which know exactly how to obtain and attest several data related to a software component [28] (named as `Component Type`). For example an IMC/IMV is in charge of collecting and verifying all the data related to the AntiVirus software, so that the IMV could request the IMC the measurement of any kind of attribute regarding the configuration or state of one (or even several [28]) software products. In the same manner, if this idea is brought to a model which focuses on properties, the IMCs, instead of being responsible of certain `Component Types` as they were before, they would be in charge of attesting the properties.

Although there is no need in fact to group the properties in any manner (that means that every property could be attested by only one IMC or that all the properties could also be attested by one IMC), it has been decided to create a classification based on the security field they can be placed. There are some reasons that motivate and justify this classification:

[7] Expensive understood as costs of implemtation

- Granularity: in a scale of coarse-/fine-grained, the grouping of the properties in security field-related sets would be in the middle, offering therefore more versatility than a coarse-grained solution as well as easier to be manipulated than a fine-grained solution.

- Efficiency: a set of too many IMCs could be terribly inefficient and computationally more expensive.

- Compatibility and extensibility: the IMCs could be easily replaced or combined with other IMCs that manage the same field properties. By an approach with only one IMC, it should be fully replaced and with an approach of an IMC per property the scenario could be too difficult to manage.

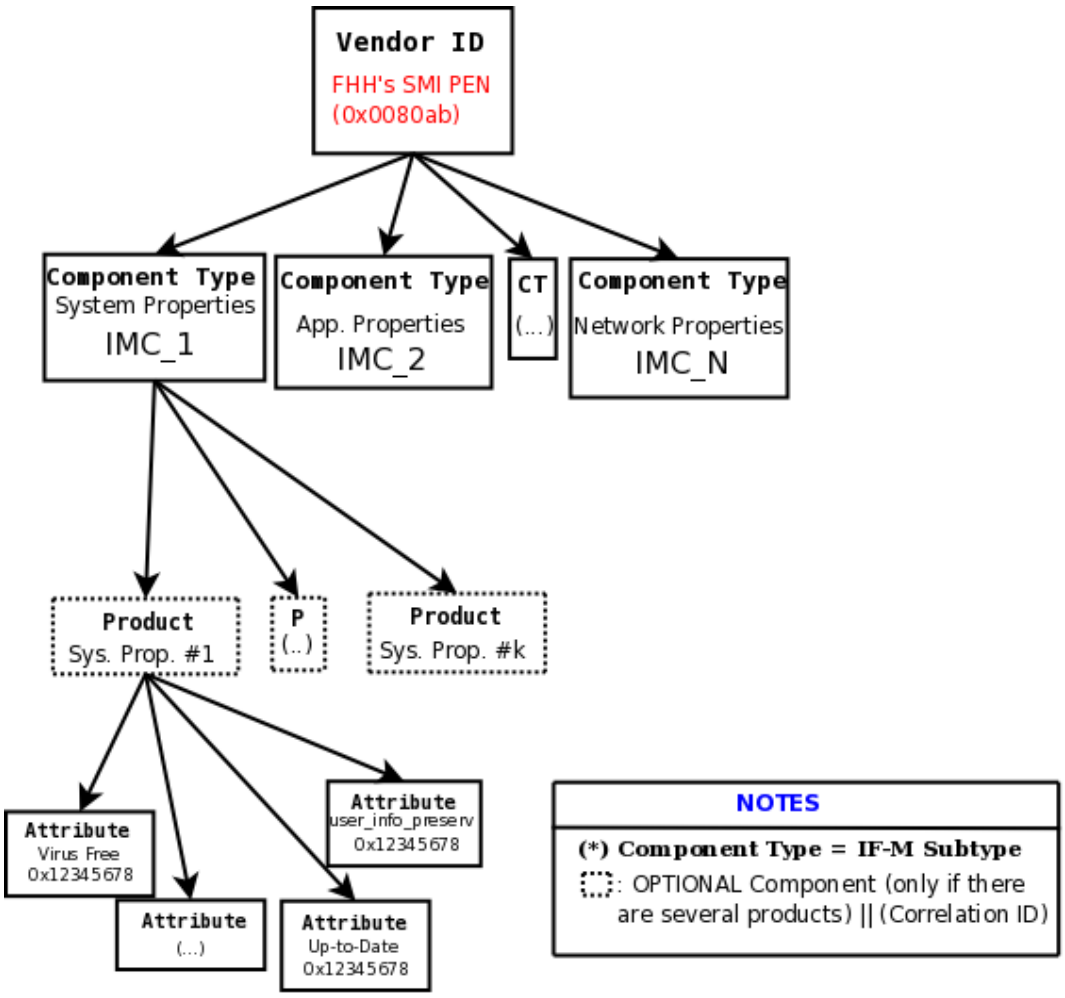



**Figure 3.6:** (partial) Classification of Properties within a hierarchy using the FHH SMI PEN as Name Space

Regarding this grouping of properties, this (uncomplete) classification as in a hierarchy is represented in figure 3.6. The FHH vendor ID (0x0080ab) has been used as a name space to define the properties. Therefore, when in an IF-M message a property must be somehow identified, the corresponding IMC or IMV will fill the `Vendor ID` field with the FHH SMI PEN, and the `Attribute Type` field with the code of the specific property.

At the moment and just with exemplary purposes, an uncomplete classification of the catalogue or properties is proposed:

- Network Properties
- System Properties
- Application Properties
- Capabilities properties
- Identity Properties
- Connectivity Properties
- (...)

In figure 3.7 it is represented the architecture of this PBA version without a TTP from a high-level point of view. In this figure, it is shown how every pair IMV/IMC uses the interface IF-M to communicate with each other in order to request the measurement of properties as well as to return the results of such measurements. Thse messages are collected by the TNCS (or the TNCS depending on the case) and filled within batch IF-TNCCS messages that will be delivered to the corresponding entity in the *Integrity Evaluation Layer*. These entities will eventually deliver each of the IF-M messages to the *Integrity Measurement Collector/Verifier* that would have expressed their interest for them.

In order to provide a more complete overall overview, in the following and making use of the IF-M Specification [28] as a reference, an entire IF-M case of use will be explained (adapting of course the concepts to the version that is being considered). Within this exemplary interaction between the entities, the calls and messages between them (including the initiator of the process as well as the sender and receiver of the messages) are indicated.

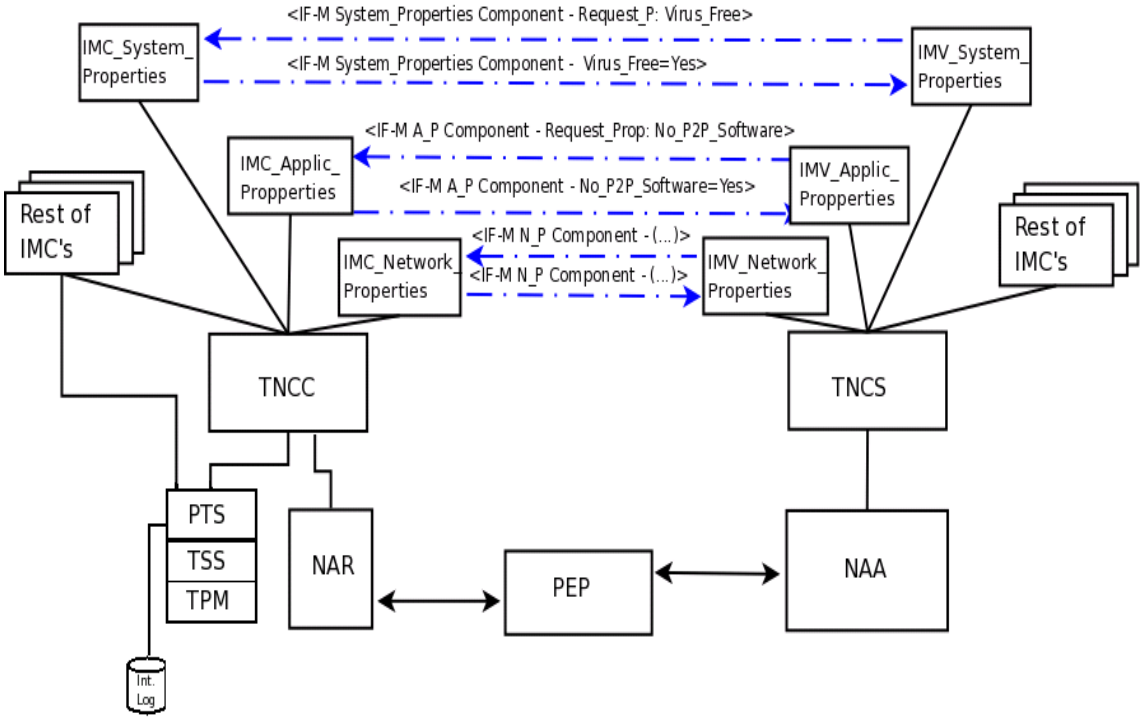



**Figure 3.7:** PBA within the TNC without using a TTP - Version 3

The example corresponds to a use case in which the TNCS decides to initiate the assessment of an endpoint because it is required. There are several reasons that could motivate this decision. Amongst others: that the TNCS becomes aware of the occurrence of certain event, as a consequence of a change in the TNCS's policy or even due to a policy that requires periodic reassessment.

Whatever may be the reason that motivates the assessment of the endpoint, once it has been decided the TNCS within the PDP invokes a set of IMVs depending on the policy to initiate the assessment. Each one of this set of IMVs consults its policy and decides whether to:

(a) Send any kind of IF-M messages in the case that it is unwilling or unable to participate in the assessment

(b) Send a request for measurement of properties to its corresponding IMC. In addition

to this request, the IMV could also attach certain information that the IMC may need on its properties evaluation. This fact can be easily achieved by using the IF-M, since it allows the existence of self-defined attributes by means of using an alternative *Attribute Name Space*. So that, once the Name Space (`Vendor ID`) has been indicated, the so called field `Attribute Value` can be filled with any data depending on the needs, allowing therefore to include any kind of data, that the IMC should be able to interpret.

As a reminder, the table below represents the **structure of an IF-M Message**. At this point it is worth noting that an IF-M message is the whole set of information that a IMC receives. It is actually within the IF-TNCCS message of type IF-M in which the IF-M is encapsulated where it is indicated the `Component Type` to which the message is related (and therefore, the IMC component that should receive it).

| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |
|---|
| Version \| Reserved |
| Message Identifier |
| Flags \| Vendor ID (Attribute Type Space) |
| Attribute (1) Type |
| Attribute (1) Length |
| Correlation ID |
| Attribute (1) Value (variable length) |
| (...) |
| <Rest of Attributes> |
| (...) |
| Flags \| Vendor ID (Attribute Type Space) |
| Attribute (N) Type |
| Attribute (N) Length |
| Correlation ID |
| Attribute (N) Value (variable length) |

Each IMV that is interested in the participation in the assessment will prepare an IF-M Message composed of a set of attributes. Finally, all these IF-M messages will be packed within a IF-TNCCS Batch message that will be forwarded to the TNC Client

within the Access Requestor.

After receiving this set of IF-M messages, the TNCC passes them on to the corresponding IMCs which are interested in certain type of `Component Type`. These messages would therefore contain the properties request (in form of self-defined attributes) as well as the optional information (also in form of self-defined attributes) necessary to provide the requested properties.

Afterwards, each IMC consults its attribute policy and could choose to:

(a) Send any kind of IF-M messages in the case that it is unwilling or unable to participate in the assessment

(b) Send a subset of the requested properties that it's able to collect (possibly factoring in privacy policy)

(c) Send every property that has been requested

(d) Send previously received and cached assertion properties (possibly in addition to requested properties)

As well as it was done in the sent of the request, the IF-M messages will be packed within a IF-TNCCS Batch message, that eventually will be sent.

After receiving this set of IF-M response messages with the attested properties, the TNCS passes them to the corresponding IMVs which have expressed interest in the measurement information received from the TNCC.

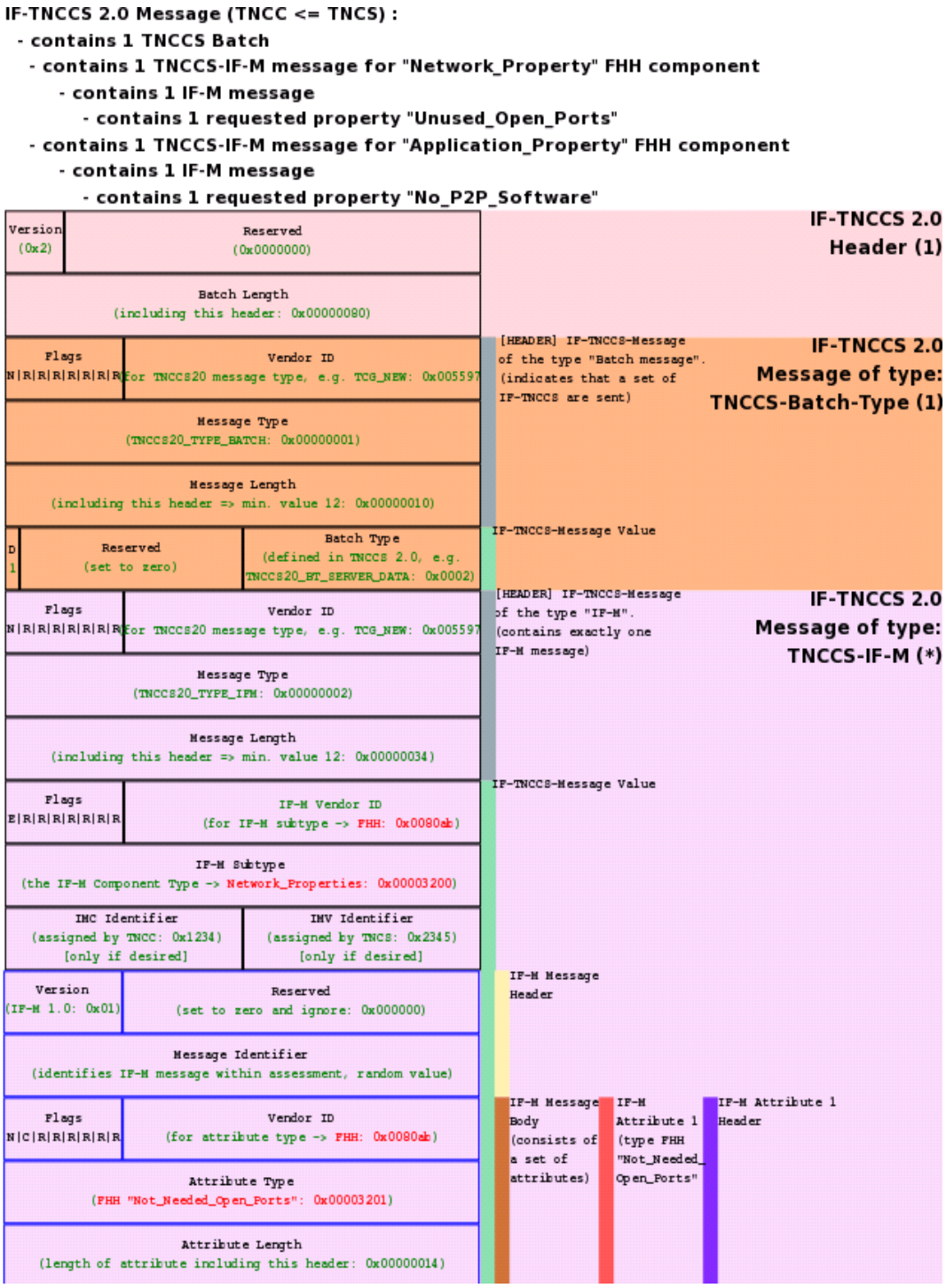
Eventually, each IMV consults its assessment policy and chooses whether to:

(a) Send any kind of IF-M messages, so that the IMCs would't be notified.

(b) Request additional measurement information (in the form of properties) from the IMCs by sending IF-M messages. In response to such IMV requests for measurement

information, the IMCs will repeat step 3 above, which could lead to additional IMV requests in step 4, according with that to the multi-roundtrip design of the protocol.

In figure 3.8 it is represented an exemplary flow of messages transmitted by the IMCs as a response to the requests of the IMVs.



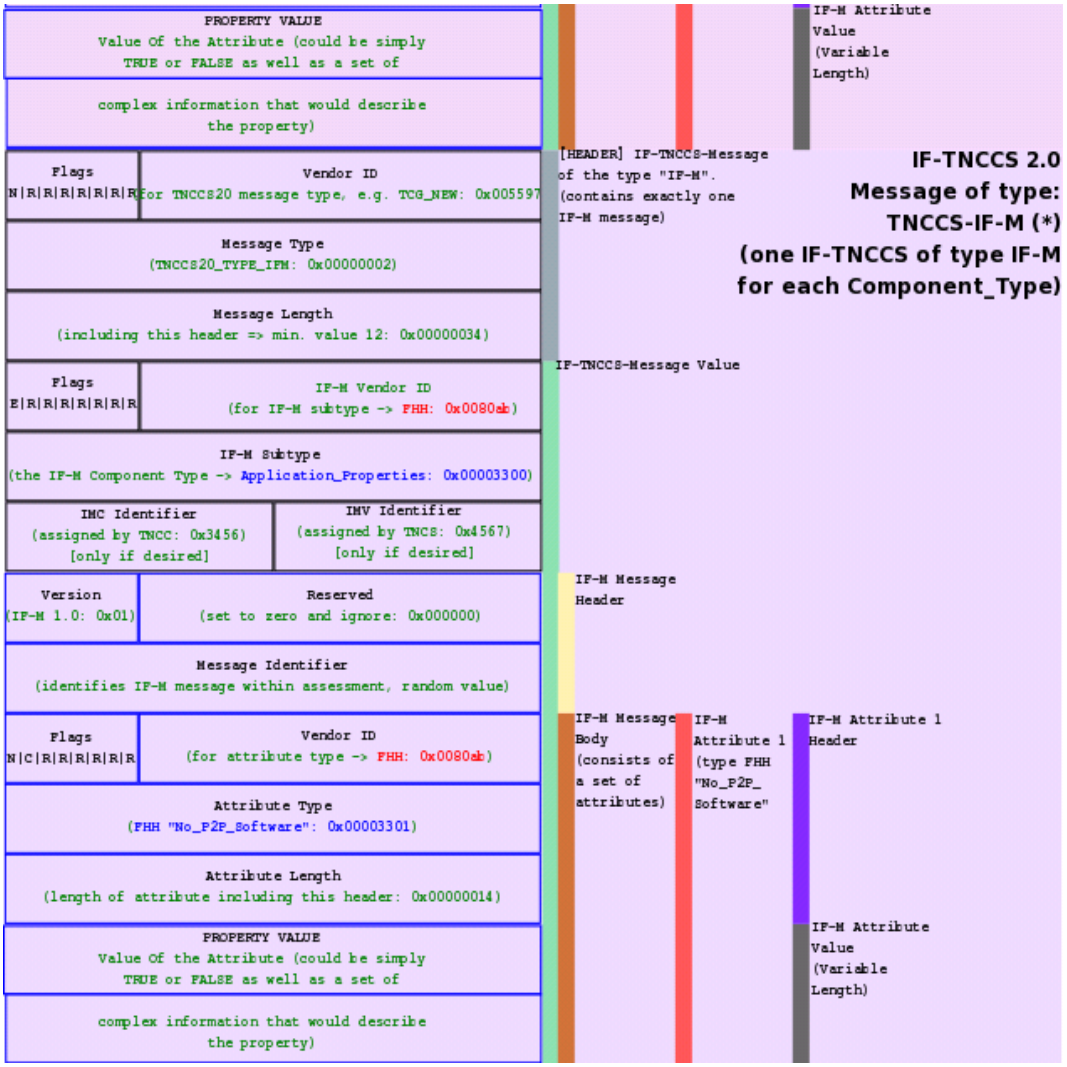



**Figure 3.8:** Flow of two IF-TNCCS Messages of type IF-M encapsulated within a IF-TNCCS Batch message from the TNCC to the TNCS. With the contained IF-M messages, the coresponding IMCs respond the IMVs with the requested properties.

## 3.2 Implementation

Making an analysis of the proposed solutions, the approach introduced with the version 3 in section 3.1.3 seems the most promising as a result of its flexibility and smooth integration with the TNC architecture. For this reason and in order to provide an example of the integration of the property-based paradigm within the *Trusted Network Connect* Architecture, it has been decided to focus on it.

Specifically, as it was mentioned in section 2.1.3, any proposed solution should be eventually integrated within the tnc@fhh. **tnc@fhh** [4, 14, 3, 12, 13] is an open source implementation of the TNC achitecture specified by the Trusted Computing Group and developed at the Fachhochschule Hannover, which implements all the core TNC architecture's Access Requestor (AR) and Policy Decision Point (PDP) entities components as well as most of the interfaces between them.

The vast majority of elements of this version are closely related to the TNC's layer named *Integrity Measurement Layer* (IML), which, as described in section 2.1.2, is placed in the highest level of the architecture. Within the IML are situated some specific plug-in components that are specialized in collecting (IMCs) and in verifying (IMVs) the information which is related to the integrity of some specific security applications or security parameters of the clients' devices. Since the development of the components that set up the third suggested solution are strongly dependent on this layer, a deeper overview of its implementation within the tnc@fhh as well as the way to develop the IMC/IMV pairs making use of it will be made in the following.

On the 17th of September, a brand new version (0.6.0)[8] of the tnc@fhh has been published. This new tnc@fhh realease includes precisely ***imunit-dev** 0.6.0*[9], which is a new project targeted for IMC/IMV pair developers that contains the **imunit framework** that describes the development of IMC/IMV pairs based upon imunit along with the additional documentation [4] that it is needed to understand how it works and how the framework is used. An important aspect to highlight of this version, because of its relevance in this work, is that the IF-M interface has not been yet implemented. Therefore, the communication between IMC/IMV pairs, instead of being made through this standardized interface, must be made by defining the messages' types that correspond to

[8] tnc@fhh version 0.6.0 released: `http://trust.inform.fh-hannover.de/joomla/index.php/component/content/article/1-latest-news/85-tncfhh-version-060-released`

[9] tnc@fhh imunit-dev 0.6.0: `http://trust.inform.fh-hannover.de/joomla/index.php/downloads/doc_download/28-tncfhh-imunit-dev-060`

each pair upon its development (as it will be explained later, this has to be defined in the library extended classes).

As mentioned above, the imunit provides a framework to ease the development of the Integrity Measurement Collectors and Verifiers. Not paying attention to other issues (for example amongst others the definition and management of policies) the implementation of the solution consists, roughly speaking, in developing a set of IMC/IMV pairs attesting properties. Each IMV responsible for a type of properties would, upon local defined policies, request its attestation when it comes to assess a platform (independently of the reason). On the client's side, the IMCs would register themselves in the TNC Client as entities in charge of managing the messages related to certain type of `Component Type`[10], so that when an IMV would send a message requesting a property (or a set of properties) that is considered to be related to such `Component Type`, the TNCC will deliver the message to the corresponding IMC.

It is important to note the this `Component Types` are , within the PBA approach, different from the conventional (Firewall Component, Antivirus Component...) and in this case regard to security fields concerns (`System Properties`, `Network Properties`, `Application Properties`...).

Finally, after receiving the messages delivered by the TNCC which originally were sent by the IMVs, each IMC will evaluate the requested properties of which it is responsible for and will eventually send the results to the IMV (acting exactly as described at the end of section 3.1.3).

[10]Since the IF-M version has not been yet implemented, the reality is slightly different: the IMC/IMV pairs are responsible for managing certain type of messages that have to be defined within their classes

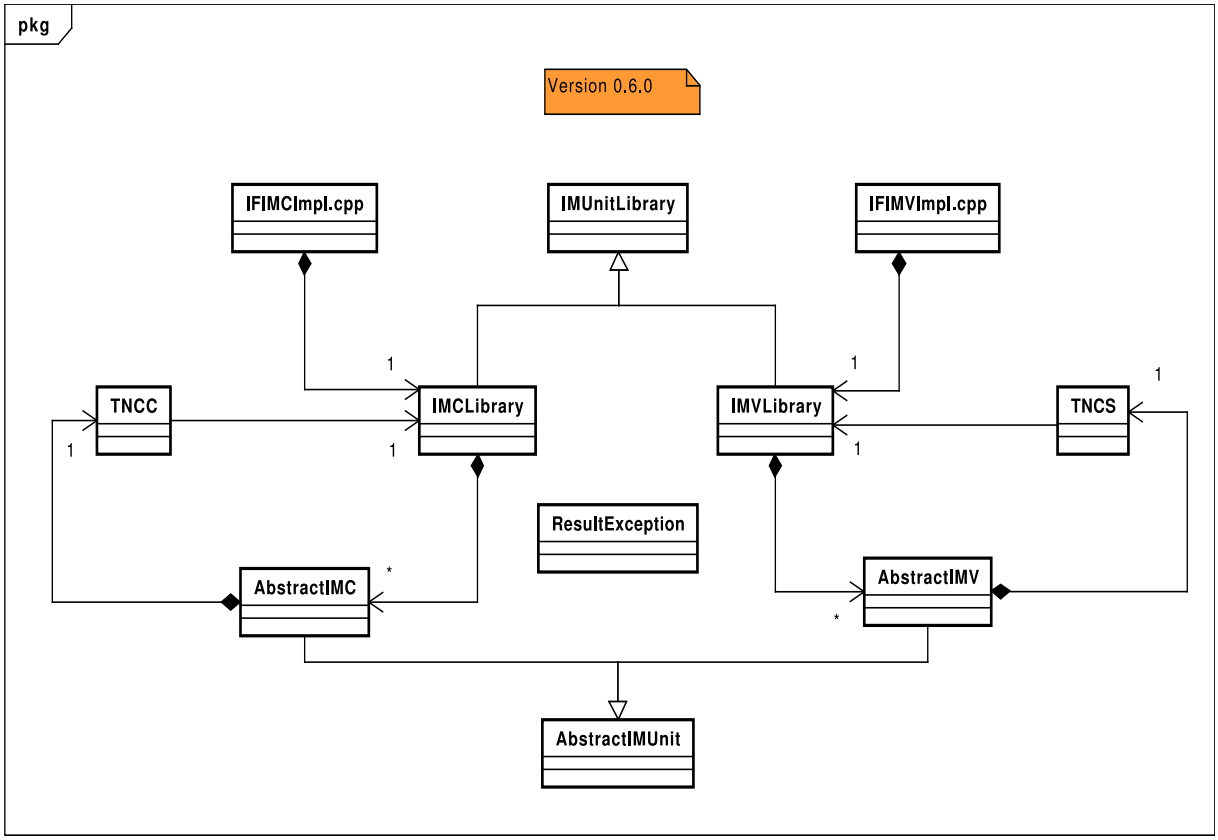



**Figure 3.9:** Architecture overview for imunit [4]

Going into further details, figure 3.9 gives an overview of the imunit architecture, indicating the classes that set it up as well as the relations that are established between them. Roughly speaking, the components of the imunit that are especially relevant for the development of the solution are the following:

- `IMUnitLibrary`: general class that encapsulates similarities of an IMC and an IMV **library**. Provides general information about the specific library (name, message types used) and handles the de-/initialization process. There will be one instance of this class for each TNCC or TNCS that uses the IMC/IMVLibrary that extends this class. Since it is supposed that there will be several libraries (at least one for each `Component Type`), there will be as well several instances.
  - `IMCLibrary`: class that inherits from `IMUnitLibrary` and encapsulates the IMC specific functionalities of an IMC library. Regarding the connections, it multiplexes the incoming calls from a TNC Client to the specific AbstractIMC's instance that makes use of that Library.

  - `IMVLibrary`: class that inherits from `IMUnitLibrary` and encapsulates the IMV specific functionalities of an IMV library. Regarding the connections, it multiplexes the incoming calls from a TNC Server to the specific AbstractIMV's instance that makes use of that Library.

- `AbstractIMUnit`: general class that encapsulates similarities of an IMC and an IMV **instance** that is bound to a specific connection. The connection is handled via the TNCC (or the TNCS). This class implements methods that are available for the IMCs/IMVs (`notifyConnectionChange()`, `batchEnding()` and `receiveMessage()`).

  - `AbstractIMC`: class that inherits from `AbstractIMUnit` and represents instances of an IMC that are bound to a certain connection. It manages the state of a specific IMC related to a given connection ID.
  - `AbstractIMV`: class that inherits from `AbstractIMUnit` and represents instances of an IMV that are bound to a certain connection. It manages the state of a specific IMV related to a given connection ID.

When it comes to develop an own IMC/IMV pair, the developer must write (for each pair) the code of four classes that extend the classes below:

- `IMCLibrary`: i.e. IMCLibrary_Network_Prop.{cpp,h}, IMCLibrary_System_Prop.{cpp,h}, IMCLibrary_Application_Prop.{cpp,h}...
- `IMVLibrary`: i.e. IMVLibrary_Network_Prop.{cpp,h}, IMVLibrary_System_Prop.{cpp,h}, IMVLibrary_Application_Prop.{cpp,h}...
- `AbstractIMC`: i.e. IMC_Network_Properties.{cpp,h}, IMC_System_Properties.{cpp,h}, IMC_Application_Properties.{cpp,h}...
- `AbstractIMV`: i.e. IMV_Network_Properties.{cpp,h}, IMV_System_Properties.{cpp,h}, IMV_Application_Properties.{cpp,h}...

This situation has been explained by figure 3.10 where the orange boxes **represent the classes** that are already developed as a part of the tnc@fhh (some of them set up the imunit framework) and the blue boxes **represent instances of the classes** that should be implemented. This way, it is shown how there is one instance of each library type for each TNCC/TNCS using it and there is one instance of each IMC/IMV type for each connection that uses it. Just as an exemplary sample of how it works, the class

`IMCLibrary_Network_Prop.cpp` has been instanced twice (one for each of the two *existing* connections), but that is not the general situation, in which there will be one instance of each library type as well as one instance of each IMC/IMV type (i.e. of each IMC/IMV that is responsible for a `Component Type`).

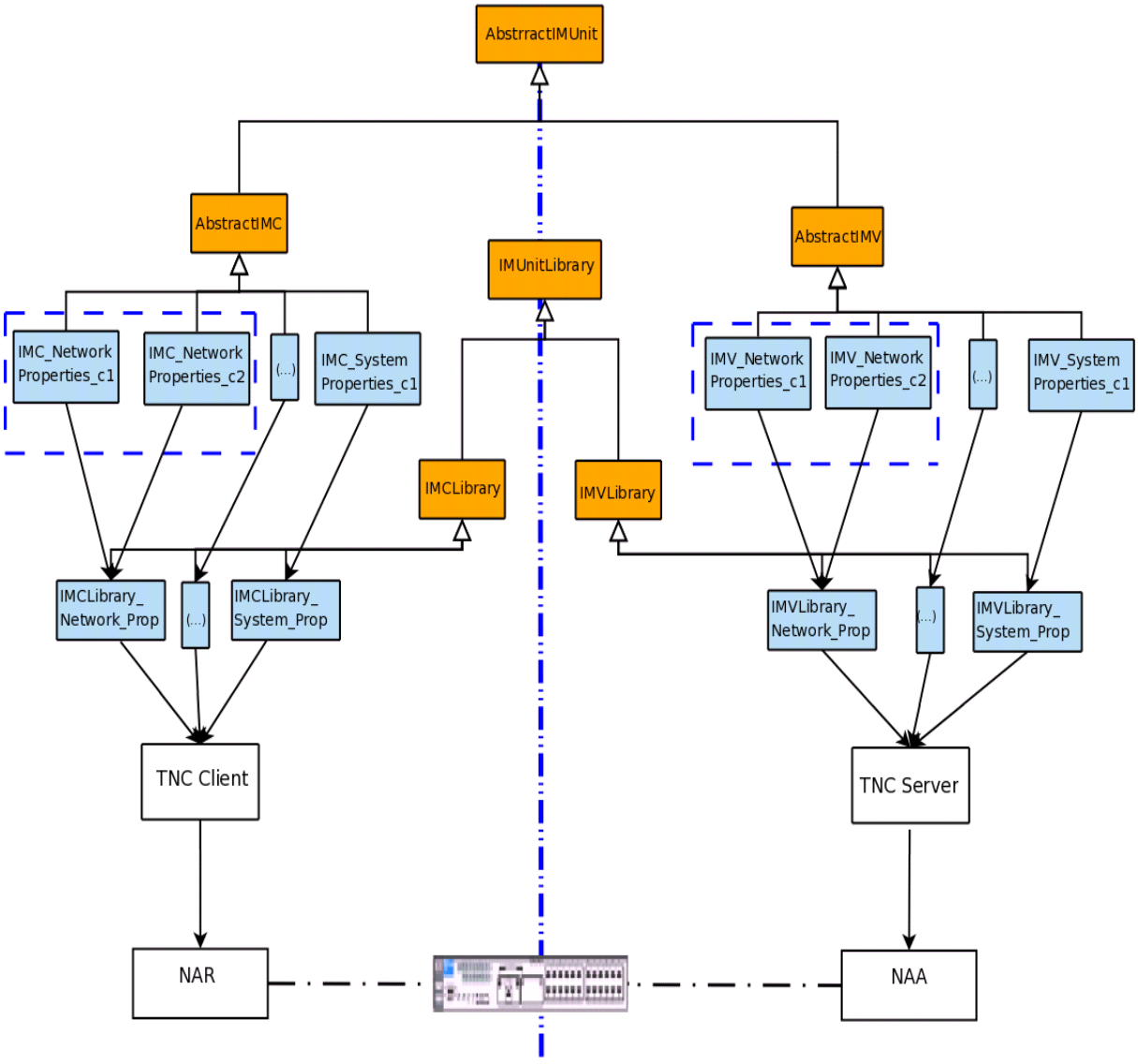



**Figure 3.10:** Overview of the implementation of IMC/IMV pairs with the use of the imunit-dev framework

An important comment about the graphics in figure 3.10 is that the represented TNCC and TNCS are the *real* entities and not the TNCC and TNCCS *interface classes* represented in the diagram above, which are used internally to allow the implemented IMC/IMV pairs to call the TNC Client and TNC Server through themselves.

It is worth remarking that, since the TNCC/TNCS can handle different connections, there is the necessity of having a connection-based representation of each IMC/IMV. This means that there will be **one instance of the IMC/IMV for each ongoing connection** independently of the number of TNCC/TNCS. Nevertheless, there will be only one **instance of the corresponding IMCLibrary/IMVLibrary for each existing TNCC/TNCS**. This situation (not considering the existence of more than TNCC/TNCS) is also represented in Figure 3.10.

The specific steps [4] that should be followed in order to implement each one of the classes mentioned above (that is the ones that are needed to develop a Collector/Verifier pair) are indicated below. As a clarifier example, the steps will be explained in the following along with the development of the IMC/IMV pair that is responsible for the *properties* related to the applications (`Application_Properties` (as for example the properti `No_P2P_Software`.

1. **Development of the IMC**

   (a) **Creation of an extending class of** `IMCLibrary`: i.e. `IMCLibraryApp_Prop`

   - Definition of the IMC's message types:

```
#define VENDOR_ID 0x0080ab
#define MESSAGE_SUBTYPE 0xfe //actually, the one from the message
```

   - Implementation of a de-/constructor. Addition of the message type defined above to the list of message types the IMC wants to receive:

```
IMCLibraryApp_Prop::IMCLibraryApp_Prop(){
   (...)
   this->addMessageType(VENDOR_ID, MESSAGE_SUBTYPE);
}
```

   - Initialization of the imunit framework (mapping of IF-IMC C functions to C++ functions):

```
TNCFHH_IMCLIBRARY_INITIALIZE(IMCLibraryApp_Prop) ;
```

   - Implementation of the pure virtual factory method, which is called when a **new connection** is created. By calling this method, a new instance of the IMC class is created (i.e. the Class that will be developed below). Since there is only one (for each type) Library instantiation for each TNCC/TNCS, but one IMC/IMV instance for each ongoing connection, the Library

will manage the existences of the IMCs/IMVs. Therefore, this method, which is called upon a new connection is in charge of creating the new IMC's instances.

```
tncfhh::iml::AbstractIMC *IMCLibraryApp_Prop::createNewImcInstance(
    TNC_ConnectionID conID){
    (...)
    return new IMCApp_Properties(conID, this);
}
```

(b) **Creation of an extending class of `AbstractIMC`**: i.e. `IMCApp_Properties`

- Definition of the de-/constructor. This instantiates a TNCC object which can be used to talk to the *real* TNCC by calling methods of the TNCC instantiation (via the pointer to the `IMCLibraryApp_Prop`). This is the method which is called upon a new connection. It requieres

```
IMCApp_Properties::IMCApp_Properties(TNC_ConnectionID conID,
    IMCLibraryApp_Prop *pIMCLibraryApp_Prop) IMCApp_Properties(conID,
    pIMCLibraryApp_Prop){
    // initialize
}
```

- Implementation of the mandatory `beginHandshake()` method.

```
TNC_Result IMCApp_Properties::beginHandshake()
{
   (...)
   std::string sendMessage("Example message from IMCApp_Properties"); //Ex
   this->tncc.sendMessage((unsigned char*)sendMessage.c_str(), sendMessage.
       size()+1/*for'\0'*/, VENDOR_ID, MESSAGE_SUBTYPE);
   return TNC_RESULT_SUCCESS;
}
```

- Implementation of optional methods, which although already implemented by the imunit framework, will be on a regular basis overridden.
  - `receiveMessage()`

```
TNC_Result IMCApp_Properties::receiveMessage(TNC_BufferReference
    message, TNC_UInt32 messageLength, TNC_MessageType messageType)
{
   (...)
   std::string sendMessage("Another example message from
       IMCApp_Properties.");
   this->tncc.sendMessage((unsigned char*)sendMessage.c_str(),
       sendMessage.size()+1/*for'\0'*/, VENDOR_ID, MESSAGE_SUBTYPE);
   return TNC_RESULT_SUCCESS;
}
```

  - `batchEnding()`

```
TNC_Result IMCApp_Properties::batchEnding()
{
  (...)
```

```
  return TNC_RESULT_SUCCESS;
}
```

– `notifyConnectionChange()`

```
TNC_Result IMCApp_Properties::notifyConnectionChange()
{
  if(this->getConnectionState() == TNC_CONNECTION_STATE_HANDSHAKE)
  return TNC_RESULT_SUCCESS;
}
```

2. **Development of the IMV**

(a) **Creation of an extending class of** `IMVLibrary`: i.e. `IMVLibraryApp_Prop`

```
/*The implementation of the IMVLibraryApp_Prop is completely symmetrical with
    the one explained in the section above (IMCLibraryApp_Prop). Therefore, the
     steps will just be introduced, but no explained.*/
```

- Definition of the IMV's message types
- Implementation of a de-/constructor. Addition of the message type defined above to the list of message types the IMV wants to receive.
- Initialization of the imunit framework (mapping of IF-IMV C functions to C++ functions)
- Implementation of the pure virtual factory method, which is called when a new connection is created (to create a new instance of the IMV class)

(b) **Creation of an extending class of** `AbstractIMV`: i.e. `IMVApp_Properties`

- Definition of the de-/constructor. This instantiates a TNCS object which can be used to talk to the *real* TNCS by calling methods of the TNCS instantiation (via the pointer to the `IMVLibraryApp_Prop`).

```
/*The implementation of the IMVApp_Properties is very similar to above (
    IMCApp_Properties). Therefore, only the (different) relevant steps will
     be explained in the following.*/
```

- Implementation of optional methods, which although already implemented by the imunit framework, will be overridden.
  – `receiveMessage()`: this method is called for the IMV to receive a message sent from the IMC (which was received by the TNCS). There are several possibilities, amongst othes it can be checked whether the first round is having place or not and act in a different manner depending on that. For example, requesting the IMC the property `No_P2P_SW` in the first round and (depending also on the results) asking for more information or providing a recommendation in the following. The management of the round counter that allows that is indicatd below:

* Set to 0 at the end of `IMC/IMVLibrary::notifyConnectionChange()` when it is called with `newState == TNC_CONNECTION_STATE_HANDSHAKE`.
* IMCs/IMVs: increased before IMC/VLibrary::batchEnding returns.
* IMCs: increased before IMCLibrary::beginHandshake returns.

```
TNC_Result IMVApp_Properties::receiveMessage(TNC_BufferReference
    message, TNC_UInt32 messageLength, TNC_MessageType messageType)

{
  if (this->getRound() < 1) {
   (...)
   std::string sendMessage("Example message from IMVApp_Properties");
   this->tncs.sendMessage((unsigned char*)sendMessage.c_str(),
       sendMessage.size()+1/*for'\0'*/, VENDOR_ID, MESSAGE_SUBTYPE);
  }else {
   (...)
   validationFinished = true;
   actionRecommendation = TNC_IMV_ACTION_RECOMMENDATION_ALLOW;
   evaluationResult = TNC_IMV_EVALUATION_RESULT_DONT_KNOW;
  }
  return TNC_RESULT_SUCCESS;
}
```

– `batchEnding()`.

– `notifyConnectionChange()`.

In order to implement the `Application_Properties` IMC/IMV pair proposed as an example, which is responsible for attesting properties related to the applications within the platform (for example `No_P2P_SW`), the methods above should me modified. Especially, the methods refering to:

- The initialization of the IMC/IMV upon a new connection (`IMCLibraryApp_Prop::createNewImcInstance()`, `IMVLibraryApp_Prop::createNewImcInstance()`).
- The begin of the Handshake (`IMCApp_Properties::beginHandshake ()`).
- The receivement and sending of messages (`IMCApp_Properties::receiveMessage()`, `IMVApp_Properties::receiveMessage()`).

In conclusion, in this section, the imunit as a framework to facilitate the implementation of IMC/IMV pairs has been introduced. To deploy a complete implementation of the solution proposed in section 3.1.3, a whole set of collector and verifier pairs should

be implemented as explained above. Each of this pairs would be responsible for a set of security-field related properties. Finally, and to complete the implementation, the local policies considering the endpoint's fulfillment of properties should be also integrated.

# Chapter 4

# Results and Conclusions

# 4.1 Results

## 4.1.1 General Overview and Evalution of the Solutions

In this work it has been intended to **merge** somehow a **property-based NAC policies** approach for attestation with the **Trust Network Connect Architecture**. The idea of including the PBA idea within the TNC was motivated **on one hand** by the lacks of the TNC conventional attestation (amongst others, its incapacity to express the desired platforms' characteristics as well as the privacy concerns that its use entails), and **on the other hand** by the expressiveness[1] of the properties to describe the features of the platforms. Therefore, theoretically, the property-based attestation provides a good approach to evaluate the platforms that attempt to connect to the network, but when it comes to facing the development of its solution, there are some difficulties that should be solved.

Nonetheless, in spite of the existing difficulties, three versions that try to solve the problem attending to certain requirements defined in the sections 2.1.4 and 2.2.1 have been proposed. Adding to the already mentioned requirements, the concept of the solutions flexibility, understood as the capacity of the solution to be extended with all sort of properties, has been also considered. Regarding this concerns, the table 4.2 indicates the fulfillment of those properties.

Moreover and depending on several factors[2], it is suggested that the conventional TNC attestation (based on configurations, product state or information among others) could be used along with any of the three proposed versions. This fact is perfectly feasible since the base of all the approaches consists of a set of IMC/IMV pairs that could be easily combined with the others. The only pertinent change that should be made in this case settles in the treatment of the policies, which should consider on one hand the privacy concerns in the client side and on the other hand the combination of the results that come from different approaches.

With reference to this fact, in the table 4.2 have been included three more versions that represent this combination (marked in the table as V1*, V2*, V3* for the versions

[1]However, this expressiveness may also lead to the difficulty of actually obtaining such properties, becoming therefore a problem. This fact could result in an implementation that would differ from the theoretical concept since it is not able to completely describe the platforms' characteristics.

[2]Some of these factors could be the possible difficulty of determining certain properties or the importance given to the privacy concerns (that in any case could be limitated by local privacy policies) amongst others.

| Version / Requirement | V1 | V1* | V2 | V2* | V3 | V3* |
|---|---|---|---|---|---|---|
| **Availability** | × | × | × | × | ✓ | ≃ / × |
| **Security** | ✓ | ✓ | ✓ | ✓ | ≃ | ≃ |
| **(no) Configuration Discrimination** | ✓ | ≃ / × | ✓ | ≃ / × | ✓ | ≃ / × |
| **Privacy** | ✓ | ≃ | ✓ | ≃ | ✓ | ≃ |
| **Scalability** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **Flexibility** | ≃ | ≃ | ≃ | ≃ | ✓ | ✓ |
| **Reduced Complexity** | ×[3] | ×[3] | ×[3] | ×[3] | ×[4] | ×[4] |
| **Compatibility** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |
| **(no) Inconsistent configurations** | × | × | × | × | ✓ | ≃ / × |
| **(no) Alternative Applications Discrimination** | ✓ | ≃ / × | ✓ | ≃ / × | ✓ | ≃ / × |

**Table 4.2:** Requirements' fulfillment of the solutions

V1, V2 and V3 combined with the conventional TNC attestation model).

The contents of each field within the table should be interpreted as indicated in the following:

- ✓: requirement completely fulfilled.
- ≃ : requirement partially fulfilled.
- ×: requirement not fulfilled.
- ≃ / ×: requirement partially/not fulfilled depending on the implementation.

[3]The complexity of the implementation is a consequence of the necessity of a TTP's use and therefore the need of implementing an interface between it and the system and as well as implementing the software in the TTP to perform the mapping of configurations into properties.

[4]Since the procedures to obtain each of them can strongly vary from one to another, and can be very complex, the complexity of this version consists in implementig the IMCs to be able to collect the properties.

### 4.1.2 Definition and Organization of the Properties

As it was already mentioned in the sections above, the IF-M interface has been used to communicate the IMC/IMV pairs in all the proposed solutions. This interface allows, by means of the use of the fields:

- `IF-M VENDOR ID` and `IF-M SUBTYPE (IF-M COMPONENT TYPE)` within the IF-TNCCS Message of type IF-M.
- `VENDOR ID` and `ATTRIBUTE TYPE` within the IF-M message.

the selection of the IMC Component[5] that will receive the IF-M messages by using the first two fields as well as the use of self-defined properties (by using the second two fields that actually refer to attributes within the IF-M Specification [28]).

The `IF-M VENDOR ID` field actually indicates in both cases a *Name Space* to define the `Component Types` and the `Attributes`. There is a standardized set of `Component Types` and `Attributes` defined in the specification. For all these standardized elements, the value of the field will be the corresponding to the TCG SMI (0x005597), but in the case of the solutions proposed, will be the referent to the FHH (0x0080ab).

Once it is clear that the used name space will be the one from the FHH, some notes about its use to define the Component Types and Properties (as attributes) can be introduced.

The table below represents a classification of some exemplary Component Types and Properties defined by making use of the FHH Name Space. Some of these elements have been used in the examples of the chapter 3.1.3 and the others only try to give an idea of the properties that can be defined as well as the reason of this classification. It's important highlighting that this catalogue of properties doesn't intend neither to be complete nor a strict reference, since a complete catalogue of properties requires a deeper study of the needs of an organization and this work is focused on the mechanisms.

[5]There are actually two manners of communicating with a IMC: the first of them consists in indicating the Component Type which the IF-M message is intended to, so that a set of IMCs are registered within the TNC Client to receive all the IF-M messages sent to such type of component, while the second manner consists of indicating directly the identifier of a specific IMC

The organization of the elements within the FHH's name space has been made following a classifications based on the Component Types that have been used. The Component Types (of which the IMC are responsible for) and the attributes must be numbered. Although these assignments could coincide, the identifiers of the properties have been distributed in relation to the IMC that should be responposible for managing the information related to a component type. However this relationship is not so strict, being therefore any IMC able to reuse other properties.

Classification of the (exemplary) self-defined attributes:

| PBA Version | IMC responsible | Properties |
|---|---|---|
| **PBA_v1** *(with TTP)* | *IMC_p* (0x00000001) | •Platform Configuration (0x00000010) •Nonce (0x00000011) •Digital Certificates List (0x00000012) |
| **PBA_v2** *(with TTP)* | *IMC_p*$_2$ (0x00000020) | •Fulfilled Properties (0x00000021) •List of properties (0x00000022) •Nonce (0x00000011) •Digital Certificate List (0x00000012) |
| **PBA_v3** *(without TTP)* | **System Properties** (0x00003100) | •Virus_free (0x00003101) •Up-to-date (0x00003102) •Compliant_With_The_Law (0x00003103) •Configured_TPM (0x00003104) •MultiUser_System (0x00003105) |
| | **Network Properties** (0x00003200) | •Not_Needed_Open_Ports (0x00003201) |

| | | |
|---|---|---|
| | | • ssh/ftp/telnet_port_open (0x00003202)<br>• Firewall_installed/in_use (0x00003203)<br>• Incoming_Email_Checked (0x00003204)<br>• No_eXecute_bit (NX) (0x00003205) |
| | **Application Properties** (0x00003300) | •No_P2P_Software (0x00003301)<br>•Browser_HTML_Capable (0x00003302)<br>•Browser_Flash_Capable (0x00003303) |
| | **Capabilities properties** (0x00003400) | •Multimedia (0x00003401)<br>•CRM_capable (0x00003402)<br>•Read_pdf/jpg/.../odt (0x00003403) |
| | **Identity Properties** (0x00003500) | •Valid identity (0x00003501)<br>•Computer's_Name (0x00003502)<br>•Number_of_Users (0x00003503) |
| | **Connectivity Properties** (0x00003600) | •Proxy_in_use (0x00003601)<br>•IPv6_Stack_Activated (0x00003602) |
| | **(...) Properties** (0x00003700) | •P1 (0x00003701)<br>•P(...) (0x00003702) |

It's important highlighting at this point that the table above just indicates the the organization of the elements as well as its identifiers, but doesn't include any information regarding the data contained when the properties are attested. There are different models that could be used to transfer the information that describes a property when it is being attested.

Regarding the granularity of the properties' description, there are different approaches that can be followed. For example, with a **fine-grained** aproach, there would be a big number of different properties that will be fulfilled with a True/False value or with a level of adequateness (whether it be quantitative or qualitative). Whereas, with a **coarse-grained** approach, there would be fewer properties but each property would be more complex, having more fields describing its contents. developer should take the decision of which approach will be used with each property when it comes to implement the functions within the IMCs that manage the requests as well as the receipt and treatment of themselves within the IMVs.

## 4.2 Conclusions and Future Work

A set of **property-based** network access control policies seems to be a **good approach** to evaluate the suitability of the platforms that attempt to connect to a corporate network by means of considering its relevant characteristics. The PBA allows a potential challenger to obtain abstract and complex properties from the platforms that pretend to come into its boundary network . Moreover, it is been proved that the **PBA can be combined** with the **TNC Architecture** (whether it be easily integrated and according to its model of functioning or not). Nevertheless, the PBA approach entails, in spite of the former, some **difficulties** at the actual moment of it is development.

The problems mentioned above consist mainly of the following issues:

- The complexity of defining a complete catalogue of properties, useful enough to sufficiently describe the features of the platforms which the verifier could be interested in.
- As a consequence of the first point, the difficulty of the procedures' development, since each one of those procedures that suppose the obtention of a property are different from the others and could be complex.
- Develop a mechanism to ensure (or at least to limite the ease of modify) the trustworhtiness of this results.

Therefore and depending on the needs of each corporation, the group responsible for the administration of the network and the establishment of the policies should study the specific requirements and determine the first point (and consequently the following too). Or in the case that this approach would be standardized, use the potential properties that in such situation would be defined.

Regarding the proposed solutions, there are some points that may be changed, so that some improvements could be achieved. In the following, some of these changes are suggested.

#### Modification to the Versions with a TTP (Versions 1 & 2)

A modification to the versions that make use of a Trusted Third Party (i.e. Versions 1 & 2 in section 3.1.3) can be made, so that instead of using a only IMC/IMV pair to attest

all the properties, a set of more complex pairs would do so. The IMCs would be more complex in the sense that they will be conscious of the platform's configuration elements that should be collected in order to obtain certain properties (still obtained through the TTP) of which it would be responsible for.

### Modification to the Versions without a TTP (Versions 3)

Althought the version 3 described in section 3.1.3 seems the most promising as a consequence of its flexibility and its smooth integration with the TNC Architecture, it suffers from a trustworthiness shortcoming. This problem is related to the reliability of obtained data, because even if trusted the whole components of the TNC architecture (including the *Integrity Measurement Collectors*), the property data is (for the moment) taken directly from the platform, existing therefore the possibility that they could have been maliciously modified.

Therefore, it is part of the future work performing a modification to this version, so that the trustworthinness of the properties could be achieved, for example through the use of the Trusted Platform Module.

# Bibliography


[1] Andress, Mandy: NAC alternatives hit the mark, Network World, 2007. http://www.networkworld.com/reviews/2007/073007-test-nac-main.html

[2] Bente, Ingo: Trusted Computing and Trusted Network Connect in a Nutshell, 2008. http://trust.inform.fh-hannover.de/joomla/index.php/projects/67

[3] Bente, Ingo and Helden, Josef von: Towards Trusted Network Access Control, Future of Trust in Computing, 2009.

[4] Bente, Ingo: imunit: A framework for the development of IMC/IMV components according to the TNC architecture. For imunit version 0.6.0., tnc@fhh developer documentation, September 2009.

[5] Chen, Liqun and Landfermann, Rainer and Löhr, Hans and Rohe, Markus and Sadeghi, Ahmad-Reza and Stüble, Christian: A protocol for Property-Based Attestation, STC '06: Proceedings of the first ACM workshop on Scalable trusted computing, 2006.

[6] Chen, Liqun and Löhr, Hans and Manulis, Mark and Sadeghi, Ahmad-Reza: Property-Based Attestation without a Trusted Third Party, ISC '08: Proceedings of the 11th international conference on Information Security, 2008.

[7] Dietrich, Kurt and Pirker, Martin and Vejda, Tobias and Toegl,Ronald and Winkler, Thomas and Lipp, Peter: A Practical Approach for Establishing Trust Relationships between Remote Platforms Using Trusted Computing, 2008.

[8] Fratto, Mike: Analysis: Network Access Control, Network Computing, 2006. http://www.networkcomputing.com/gswelcome/showArticle.jhtml?articleID=193101592

[9] Haldar, Vivek and Chandra, Deepak and Franz, Michael: Semantic remote attestation: a virtual machine directed approach to trusted computing, VM'04:

Proceedings of the 3rd conference on Virtual Machine Research And Technology Symposium, 2004.

[10] Fratto, Mike: Tutorial: Network Access Control Mike Fratto, Network Computing, 2007. http://www.networkcomputing.com/channels/security/showArticle.jhtml?articleID=201001835

[11] Dana Hendrickson: A Closer Look At NAC Policy-Enforcement, Secure Access Central,2008. http://www.secureaccesscentral.com/wordpress/?p=71.php

[12] Helden, Josef von and Bente, Ingo and Vieweg, Jörg and Hellmann, Bastian: Trusted Network Connect (TNC), 3rd European Trusted Infrastructure Summer School (Slides), September 2008.

[13] Helden, Josef von and Bente, Ingo: Towards real interoperable, real trusted network access control: experiences from implementation and application of Trusted Network Connect, Information Security Solutions Europe (ISSE) (Slides), October 2008.

[14] Helden, Josef von and Bente, Ingo: Towards real interoperable, real trusted network access control: experiences from implementation and application of Trusted Network Connect, ISSE 2008 Securing Electronic Business Processes, 2009.

[15] Hendrickson, Dana: Measuring Network Access Control (NAC) is Harder Than Counting Rabbits, Secure Access Central, 2007. http://www.secureaccesscentral.com/wordpress/?p=65.php

[16] Kühn, Ulrich and Selhorst, Marcel and Stüble, Christian: Realizing Property-Based Attestation and Sealing with Commonly Available Hard- and Software, STC '07: Proceedings of the 2007 ACM workshop on Scalable trusted computing, 2007.

[17] Microsoft: Microsoft Server TechCenter: Active Directory, 2003. http://technet.microsoft.com/en-us/library/cc780036(WS.10).aspx

[18] Munetoh, Seiji and Nakamura, Megumi and Yoshihama, Sachiko and Kudo, Michiharu: Integrity Management Infrastructure for Trusted Computing, IEICE - Trans. Inf. Syst. Vol.E91-D, No.5, 2008.

[19] Pearson, Siani: Trusted computing platforms : TCPA technology in context, Prentice Hall PTR, 2002.

[20] Raymond Ng: Trusted Platform Module: TPM Fundamental, Asia Pacific Trusted Infrastructure Summer School (APTISS), 2008.

[21] Sadeghi, Ahmad-Reza and Stüble, Christian: Property-based attestation for computing platforms: caring about properties, not mechanisms, NSPW '04: Proceedings of the 2004 workshop on New security paradigms, 2004.

[22] Snyder, Joel: Network Access Control : How does it work?, Network World Test Alliance, Network World, 2006. http://www.networkworld.com/buyersguides/guide.php?cat=866251&mt=how&PRWORK=Network%20Access%20Control

[23] TCG Infrastructure Working Group: Platform Trust Services Interface Specification (IF-PTS), November 2006.

[24] TCG Trusted Network Connect: TNC IF-IMV, May 2006. Specification Version 1.1

[25] TCG Trusted Network Connect: TNC Architecture for Interoperability, May 2007. Specification Version 1.2

[26] TCG Trusted Network Connect: TNC IF-PEP: Protocol Bindings for RADIUS, February 2007. Specification Version 1.1

[27] TCG Trusted Network Connect: TNC IF-IMC, February 2007. Specification Version 1.2

[28] TCG Trusted Network Connect: TNC IF-M: TLV Binding, February 2008. Specification Version 1.0

[29] TCG Trusted Network Connect: TNC IF-M Security: Bindings to CMS, February 2008. Specification Version 1.0

[30] TCG Trusted Network Connect: Specifications FAQ, February 2008. http://www.trustedcomputinggroup.org/files/static_page_files/0A663A1A-1D09-3519-AD4C59422A864A71/Feb_2008_TNC_Spec_Release_FAQ_final_feb_12.pdf

[31] TCG Trusted Network Connect: TNC IF-TNCCS: TLV Binding, January 2008. Specification Version 2.0

[32] Trusted Computing Group: About TCG, 2009. http://www.trustedcomputinggroup.org/about_tcg

[33] TCG Trusted Network Connect: Developers resources. `http://www.trustedcomputinggroup.org/developers/trusted_network_connect/`

[34] TCG Trusted Network Connect: TNC IF-T: Binding to TLS, May 2009. Specification Version 1.0

[35] VeriSign: Introduction to Digital Certificates. `http://www.verisign.com.au/repository/tutorial/digital/intro1.shtml`

[36] Bin Xiao, Dr.; et al: Autonomic and Trusted Computing. 4th International Conference, ATC 2007, Hong Kong, China, July 11-13, 2007. Proceedings.