Policy-Driven Adaptive Protection Systems.

Diaz Tellez, Yair Hernando

Policy-Driven Adaptive Protection Systems

by

Yair Hernando Diaz Tellez

Submitted in partial fulfillment of the requirements of the
Degree of Doctor of Philosophy

School of Electronic Engineering and Computing Science
Queen Mary University of London
2015

Declaration

I, Yair Hernando Diaz Tellez, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature:

Date:

Abstract

The increasing number and complexity of security attacks on IT infrastructure demands for the development of protection systems capable of dealing with the security challenges of today's highly dynamic environments. Several converging trends including mobilisation, externalisation and collaboration, virtualisation, and cloud computing are challenging traditional silo approaches to providing security.

IT security policies should be considered as being inherently dynamic and flexible enough to trigger decisions efficiently and effectively taking into account not only the current execution environment of a protection system and its runtime contextual factors, but also dynamically changing the security requirements introduced by external entities in the operational environment.

This research is motivated by the increasing need for security systems capable of supporting security decisions in dynamic operational environments and advocates for a policy-driven adaptive security approach.

The first main contribution of this thesis is to articulate the property of specialisation in adaptive software systems and propose a novel methodological framework for the realisation of policy-driven adaptive systems capable of specialisation via adaptive policy transformation.

Furthermore, this thesis proposes three distinctive novel protection mechanisms, all three mechanisms exhibit adaptation via specialisation, but each one presenting its own research novelty in its respective field. They are:

1. A Secure Execution Context Enforcement based on Activity Detection;

2. Privacy and Security Requirements Enforcement Framework in Internet-Centric Services;

3. A Context-Aware Multifactor Authentication Scheme Based On Dynamic Pin.

Along with a comprehensive study of the state of the art in policy based adaptive systems and a comparative analysis of those against the main objectives of the framework this thesis proposes, these three protection mechanisms serve as a foundation and experimental work from which core characteristics, methods, components, and other elements are analysed in detail towards the investigation and the proposition of the methodological framework presented in this thesis.

Acknowledgements

I want to thank my parents, Gloria and Hernando, for their continuous support. Also, I would like to express my special appreciation and thanks to my advisors, Dr Eliane L. Bodanese and Dr Theo Dimitrakos, for encouraging my research and for allowing me to grow as a research scientist. Your advice on both research as well as on my career have been priceless.

Publications

Conference papers

- ***"An Architecture for the Enforcement of Privacy and Security Requirements in Internet-Centric Services"*** TrustCom, pp.1024-1031, 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, 2012.

- ***"Secure Execution Context Enforcement Framework based on Activity Detection on Data and Applications Hosted on Smart Devices"*** The 5th ASE/IEEE International Conference on Information Privacy, Security, Risk and Trust, 2013.

- ***"Context-Aware Multifactor Authentication Based On Dynamic Pin"*** 29th IFIP TC-11 SEC 2014 International Conference, ICT Systems Security and Privacy Protection Marrakech, Morocco, 2-4 June 2014.

Patents

- ***"Processing device and method of operation thereof"*** Publication number: WO2014102526A1, Filed December 31, 2013, Europe

- ***"Processing device and method of operation thereof"*** Publication number: WO2014102523A2, Filed December 31, 2013, Europe

- ***"Client/Server Access Authentication"*** Publication number: WO2014102522A1, Filed December 17, 2013, Europe

Contents

List of Figures

List of Tables

List of Abbreviations

Abbreviation	Meaning
APA	Automated Policy Adaptor
API	Application Program Interface
BPEL	Business Process Execution Language
BPMN	Business Process Modelling Notation
BPT	Business Process Template
CV	Curriculum Vitae
DC	Data Controller
DF	Data File
DFD	Data Flow Diagram
DH	Data Host
DP	Data Provider
DPA	Data Protection Act
DPPP	Data Protection Property Policy
DRM	Digital Rights Management
DS	Data Subject
DSL	Domain-Specific Language
EC	European Commission
EHM	Event Handler Module
EHR	Electronic Health Records
GPS	Global Positioning System
HS	Host System
HSR	Host System Resource
HTML	Hyper Text Mark-up Language
ICT	Information & Communication Technologies
IDS	Intrusion Detection System
IFS	Information Flow Security
ION	Inter-Organisational Networks
IP	Internet Protocol
IPC	Inter-Process Communication
ISM	Implementation-Specific Model
IT	Information & Technology
MAC	Mandatory Access Control
MAPE-K	Monitor Analise Plan Execute - Knowledge
OECD	Organisation for Economic Co-operation and Development
OS	Operating System
OSL	Obligation Specification Language
OWL	Web Ontology Language
PAP	Policy Administration Point
PBM	Policy-Based Management
PCC	Policy Combination Component
PDP	Policy Decision Point

PEM	Policy Enforcement Module
PEP	Policy Enforcement Point
PET	Privacy-enhancing Technologies
PII	Personal Identifiable Information
PIM	Platform-Independent Model
PIP	Policy Information Point
PMA	Policy Management Agent
PMM	Policy Manager Module
POLDF	Data File Policy
POLPU	Processing Unit Policy
PPL	PrimeLife Policy Language
PR	Policy Rule
PSB	Privacy & Security Broker
PSM	Platform-Specific Model
PU	Processing Unit
RBCA	Role-Based Access Control
SAS	Software Adaptive System
SCM	Security Context Monitor
SEC	Security Execution Context
SLS	Service-Level Specification
SOA	Service Oriented Architecture
UGC	User-Generated Content
UML	Unified Modelling Language
URI	Uniform Resource Identifier
VPN	Virtual Private Network
WSDL	Web Services Description Language
XACML	Extensible Access Control Markup Language
XML	Extensible Mark-up Language
BP	Business Process
PIN	Personal Identification Number
OTP	One Time Password
IMEI	International Mobile Station Equipment Identity
IMSI	International mobile subscriber identity
SIM	Subscriber Identity Module
AES	Advanced Encryption Standard
PRNG	Pseudo-Random Number Generator
XOR	Excusive OR
DES	Data Encryption Standard
MDA	Model-Driven Architecture

Chapter 1

Introduction

This chapter gives an overview of the thesis. First, it presents the motivation and the problem statement; then, the research main contributions. Finally, the overall structure of the thesis is outlined.

1.1 Motivation

The increasing number and complexity of security attacks on IT infrastructure demands for the development of *protection systems* capable of dealing with the challenges of today's highly dynamic, distributed, multi-sourced, collaborative, and even virtualised environments.

In many systems the behaviour of the security infrastructure is driven by the governing *security policies,* which specify the actions to be taken given certain circumstances or events. Traditionally, security infrastructure is governed by static policies defined by a central administrator. This static approach works with acceptable results in fairly closed and static environments where there is complete information available and policies can be defined in advance and are not expected to change often; however, in today's reality, static defences are not enough. Reasons as to why policies might need to *adapt* dynamically [1] include increasingly sophisticated attacks, the unpredictability of (un)known vulnerabilities and threats, unanticipated changes in the environment – for example, new exploits may appear that compromise a system; changes in operating conditions and organisational goals – e.g. due to privacy laws and regulations or due to the creation of virtual organisations that come together to collaborate temporarily; also, policies may be required to adapt depending on the current system state – e.g. a system under attack may strengthen its defences to mitigate damage.

The definition of what means to be secure for a system, an organisation, an individual, or any other entity changes over time as these entities, their requirements, and the environments in which they operate all evolve. Thus, IT security policies should be considered as being inherently dynamic in this sense and, in addition, be flexible enough to trigger decisions efficiently and effectively taking into account not only the current *execution environment* of a protection system and its runtime *contextual factors* but also dynamically changing the security requirements introduced by external entities in the *operational environment*. The term *execution environment* is used to refer to the self-contained context in which the execution of a system occurs, and includes all the (logical and physical) entities that together constitute a system, their relationships, and the structure of the overall system and its properties. The term *operational environment* is used to refer to the system as *situated* in its operational context, and to refer to the external entities and their interactions observable by the situated system but out of its control of execution.

This research is motivated by the increasing need for security systems capable of supporting security decisions in dynamic *operational environments* and advocates for a *policy-driven adaptive security approach.*

1.2 Problem Statement

An adaptive system is one that dynamically changes its behaviour or structure at runtime in response to environmental changes [2]. Applying this to policy-driven security, an adaptive protection system is one that dynamically adapts to better enforce the security policies on its execution environment in order to protect the confidentiality, integrity, availability, authenticity and accountability of the information and workloads being protected [3]; and in order to achieve this end goal, the protection system requires to be both context-aware and self-aware. MacDonald [3] defines context-aware security as *"…the use of supplemental information to improve security decisions at the time the decision is made, resulting in more accurate security decisions capable of supporting more-dynamic business and IT environments"*. Self-awareness refers to the ability to observe and act according to the current own states and behaviours.

Different aspects of adaptive software systems have been extensively studied and applied in a wide spectrum of research fields and application areas such as autonomic computing, embedded systems, dependable computing, multi-agent systems, networks, service-oriented architectures, control theory, artificial intelligence, ubiquitous computing, and security. However, the proper realisation of software adaptation still remains an important intellectual

challenge. Drivers for reaching maturity levels of adaptive systems include requirements such as automation, robustness, security, quality assurance, and the management of complexity; all with associated research challenges. For a comprehensive summary of the state-of-the-art and a roadmap of research challenges see [4-6]. This research focuses on aspects of adaptive systems and their function in the dynamic adaptation of policies.

Complementary to the function of adaptive systems and relevant to this thesis is the research area of policy-based management (PBM), which emerged to address the need for reducing system management complexities. As it will be presented throughout the thesis, PBM provides the flexibility required for adaptation via the mechanism of policies while adaptive systems provide the constructs required to dynamically transform, execute, and enforce security policies.

1.2.1 Problem definition

As mentioned before, security policies are the mechanism used to drive the behaviour of policy-based protection systems. The system that the policies are applied to is referred to as the target system. By changing its security policies dynamically it is possible to reconfigure the target system's behaviour at runtime.

Most software adaptation approaches assume a single centralised administrative entity entirely responsible for the definition of the policies that govern adaptation decisions. However, in many cases such assumption limits the adaptation process by making its management highly centralised and restrictive, and not suitable for more flexible and open scenarios. Few adaptive security approaches consider scenarios such as privacy, security customisation, organisational collaborations, etc., in terms of and from the perspective of adaptation, when in such scenarios external entities not only need to express their changing requirements dynamically at runtime, but also those requirements are not always known beforehand to be considered by or predefined in the policies of the adaptation logic.

For example, consider a software system that processes personal identifiable information according to an IT management security policy. At the same time, the person, whose data is about may have changed his/hers privacy and security requirements (which can be expressed as policies) on how the data should be handled. In such a case, both the management policy and the privacy policies somehow need to be combined and transformed into a more specialised and conflict-free policy adequate to the current operational context that needs to be enforced by the system at implementation level.

This research focuses on such scenarios, investigates how to overcome the above limitation, and argues that an adaptive system should be capable of *specialisation*. That is, be able not only to monitor and detect, but also to *take into account* and *incorporate* dynamically changing external requirements (introduced by external entities and behaviour captured in the operational environment at runtime) into an adaptation process, and to be able to accommodate and enact such external requirements within a flexible management structure in a securely controlled manner. In other words, specialisation is a property of the system that consists in the adaptation process being driven by external entities' requirements (via the operational environment) but scoped by baseline administrative policies.

1.2.2 Research objectives

This research concentrates on the problem of how to develop and manage policy-based software systems capable of adapting to changing operational contexts and environments by *specialising* their behaviour in response to the changing security requirements of external entities/actors. The primary goal is to understand and articulate the property of specialisation in adaptive software systems and propose a general methodological framework for the realisation of policy-driven adaptive software systems capable of specialisation via adaptive policy transformation. The following are the main objectives of the research:

Obj. 1. To study the current state of prior literature in the areas of software adaptive systems and policy-based management, in order to extract and analyse their fundamental concepts, properties, and limitations, in order to establish a theoretical foundation and to articulate and formulate the necessary extensions towards the realisation of the proposed general framework.

Obj. 2. To develop and analyse specific policy-based protection mechanisms in scenarios where external entities have dynamically changing security requirements in order to articulate the property of specialisation.

Obj. 3. To propose and develop a policy transformation mechanism that implements the property of specialisation where the adaptation logic integrates external entities' requirements with administrative policies, and produces dynamically enhanced adaptation decisions.

Obj. 4. To propose and develop a general baseline architecture that provides the structural components and layers required to enable policy management, policy transformation and policy enforcement.

Obj. 5. To propose and develop the policy-based mechanism required for the management and actual (runtime) operationalisation across architectural layers of policy-driven adaptive systems.

1.2.3 Research challenge

Designing policy-based adaptive software systems faces important research challenges. In attempting to accomplish the above objective the following research challenge is addressed:

- It is not always possible to anticipate specific adaptations for all the set of possible operational environment conditions that may arise. Incomplete information about the operational environment implies incomplete information about how to specify the goals of the system. The first question is how to deal with uncertainty. In addition, an adaptive software system will have a set of high-level goals (e.g. management policies) that must be followed at all times while, at the same time, allowing for other goals such as those expressed by external entities to be incorporated. Therefore, the second question is how to model the system to allow for flexibility in expressing different types of goals.

The hypothesis of this research is that a policy-based management approach can provide the management structured required in adaptive systems in order to enable the property of specialisation via policy transformation. The methodological framework proposed in this thesis validates this hypothesis.

To the best of the author's knowledge a methodology that integrates PBM and (self-) adaptive systems concepts and provides a framework for the design and management of adaptive protection systems (as described above) capable of specialisation is new.

1.2.4 Adaptive system realisation criteria and scope

The study of adaptive systems covers a wide spectrum of aspects in diverse research fields and application areas. This research concentrates on the aspect of adaptability expressly enabled via policies as the adaptation mechanism and software as the provisioning mechanism. The realisation criteria and scope for this research is based on the discussion presented so far in this chapter, it is influenced by the *modelling dimensions* presented in *Software Engineering for Self-Adaptive Systems*[4], and includes only the aspects of those dimensions that, in the view of the author of this thesis, are relevant to the primary research objective of this thesis.

The following aspects define the system criteria and scope for the realisation of specialisation via adaptive policy transformation.

Goals: The goal(s) that the system should achieve may refer to specific functionality of the system for particular scenarios or may refer to adaptability aspects of the system. Regarding the latter, this research concentrates on the main goal of specialisation as a required property. Regarding the former, goals are associated to the policies in order to drive adaptive behaviour.

Evolution: As the system evolves, goals (or policies) may change themselves or in number. The system is expected to handle changes in policies for adaptation dynamically (as opposed to predefined static policies).

Flexibility: The goal(s) of the system can be either rigid or flexible. Rigid goals are prescriptive – "the system must do this", while flexible goals allow dealing with uncertainty – "the system may do this". The system should support expressing flexible goals that allow incorporating requirements of external entities.

Duration: The goal(s) of the system can be either persistent or temporary. The goal of specialisation should be persistent and a design choice. Goals associated to policies should be temporary and dynamically dependent on changes of administrative and external entities' requirements.

Change: Causes for adaptation may be triggered by external entities and the environment with which the system interacts, or by the system itself. This work focuses on the first. The system should respond to external entities' actions and associated events in order to adapt.

Mechanisms: Adaptation can be related to parameters of the system, its structure or a combination of both (e.g. web services compositions). The adaptation process here considered is scoped to parameterisation via dynamically generated policies.

Autonomy: Adaptation can range from assisted to autonomous. Since the system is expected to handle and incorporate requirements from external entities it should be treated as assisted.

Organisation: The adaptation process can be centralised or distributed. The type of system here considered is expected to handle adaptation by a centralised single component.

1.3 Research Main Contributions

The following are the main contributions of the thesis:

- Firstly, three different policy-based protection mechanisms (Chapter 4, Chapter 5, and Chapter 6) have been developed. The three mechanisms are motivated by scenarios where external entities require expressing constraints in the form of policies at runtime, and where such constraints are combined with administrative or baseline policies. Each

mechanism provides its own novelties in its corresponding fields. The mechanisms and associated main contributions are:

- A secure execution context enforcement framework based on activity detection for protected data and protected applications hosted on smart devices [7]. The framework acknowledges different types of policies issued by different administrative entities. Policies are associated to data and applications dynamically. The framework defines an active secure execution context that integrates, combines, and enforces the applicable policies based on system state, contextual information, and baseline policies. The framework provides a novel policy integration and combination mechanism that allows the dynamic runtime configuration of dynamically generated security profiles, and also proposes an enforcement architecture.

- A framework for the enforcement of privacy and security requirements in internet-centric services [8] that enables the data providers (DP) the definition of data protection policies and enforces them on the infrastructure of the data consumer (DC). The framework provides a mechanism that adds constraints on the DC's execution workflows based on contextual information and user preferences (i.e. DP's external constraints). The main novelty of the framework is that it allows the DP to specify not only privacy and access control-related constraints before any data disclosure but also to specify additional security and assurance-related constraints to be enforced after private data is disclosed.

- A multi-factor dynamic pin authentication mechanism [9] of the challenge-response family of authentication protocols that generates a pseudo-random dynamic pin based on the user input, different authentication factors, and past successful authentication attempts. The main novelty is the cryptographic transformation function that generates the dynamic pin. For each authentication attempt the crypto-function changes itself dynamically based on external contextual constraints provided by the user and the operational environment. The crypto-function provides pseudo-randomness to the authentication process since the crypto-algorithms it uses changes dynamically making it more difficult to perform cryptanalysis attacks.

– Secondly, the three protection mechanisms above mentioned were developed and used as experimental foundation in order to understand and describe the property of

specialisation. Two generic methods for achieving specialisation are presented: instantiation and integration. From this analysis the concept of policy transformation was articulated, abstracted into an architectural component, and incorporated into a general architecture of an adaptation engine. The integration of policy transformation at the adaptation-logic level as a mechanism to enable specialisation is novel.

- Finally, a methodological framework for the realisation of policy-driven adaptive protection systems is proposed. The main contribution of the framework consists of two stages: development and operational. The development stage provides an engineering-based approach for designing and building policy-based adaptive systems consisting of a 3-layered architecture that separates management, adaptation, and low-level concrete implementation concerns. The development stage provides configurable functions at each layer of abstraction. The operational stage provides a policy hierarchy tied to functions corresponding to each of the abovementioned concerns. The policy hierarchy is used as the mechanism to operate and control management functions, adaptive policy transformation functions, and low-level configuration functions. In addition, a methodological stepwise guidance covering different aspects for using and applying the framework is provided.

1.4 Thesis Outline

Chapter 2 is divided in three main sections. Section 2.2 (Adaptive Software Systems) introduces fundamental concepts on adaptation and self-adaptation, the adaptation process, and generic adaptation architectures. Section 2.3 (Policy-based Management) discusses earliest works on policy-based systems, the evolution towards policy-based management (PBM) as a paradigm, and other important concepts and definitions in the field of PBM. Finally, Section 2.4 (Related Work) discusses important works and relevant approaches on the areas of Policy-based Management and (Self-) Adaptive Software Systems that specifically relate to the methodological framework proposed in this thesis.

Chapter 3 presents an overview of the proposed model for a policy-driven adaptive protection system and defines its system requirements.

Chapter 4, Chapter 5, and Chapter 6 present three policy-driven protection mechanisms with their own motivations and contributions in specialised areas of security. For clarity, related work for each of the mechanisms has been placed in the corresponding chapters. In Chapter 4, a secure execution context enforcement framework based on activity detection on protected data and applications hosted on smart devices is proposed. In Chapter 5, a framework for the enforcement of privacy and security requirements in internet-centric services that enables data

providers the definition of data protection policies and enforces them on the infrastructure of the data consumer is proposed. In Chapter 6, the author proposes a multi-factor dynamic pin authentication mechanism of the challenge-response family of authentication protocols that dynamically (re-)configures a crypto-function which in turn is used to generate a pseudo-random dynamic pin based on the user input.

Chapter 7 presents the methodological framework for the realisation of policy-driven adaptive protection systems. First, the protection mechanisms of Chapter 4, Chapter 5, and Chapter 6 are discussed and analysed (section 7.1) with respect to the concepts on adaptation and policy-driven behaviour presented in the introductory chapters (chapters 1 to 3). Second, based on the analysis of the common core elements and aspects identified in the adaptive protection mechanisms studied, the discussion is further extended in order to conceptualise, generalise, and describe how these elements and aspects fit in a general methodology (section 7.2). Finally, the methodological framework consisting of the two main stages, development stage (section 7.3) and the operational stage (section7.4) is presented, along with a methodological stepwise guidance (section 7.5). Section 7.6 presents a final analysis of the systems requirements and how the proposed framework and stepwise guidance of the methodology fulfil those requirements.

Chapter 8 presents the research conclusions and future work.

Chapter 2

Fundamental Concepts and Related Work for Modelling Policy-Driven Adaptive Systems

2.1 Introduction

The proposed methodological approach builds on top of concepts and principles from autonomic computing and self-adaptive systems including the principle of separation of concerns between adaptation logic and application logic, the notion of adaptation manager and managed elements, and the concept of adaptation loop (i.e. a closed feedback control loop) enabled via sensors and effectors. The proposed approach is also based on concepts taken from policy-based management systems including the principle of separation of concerns between security and mechanism and its use as management paradigm, and the concepts of policy hierarchy, policy refinement, and policy transformation. All these concepts are briefly described in sections 2.2 and 2.3.

Section 2.4 discusses several important approaches in the area of policy-based adaptive systems. Different aspects and concepts are discussed and differences between those approaches and the approach proposed in this thesis are contrasted.

2.2 (Self-) Adaptive Software Systems

Adaptive software refers to systems with the ability to change their behaviour to suit particular circumstances or conditions. In the research literature, the concept of *adaptation* is largely presented using different terminology in the areas of *autonomic computing, self-adaptive systems, self-managing systems, self-governing systems, software adaptive systems (SAS)*, etc.

There is not a general consensus among researchers and practitioners, but rather an inconsistent use of terminology and strong overlaps of fundamental concepts when attempting to clearly distinguish, for example, *adaptive* vs. *self-adaptive* [10]. For instance, Norvig defines adaptive software as one that "uses available information about changes in its environment to improve its behaviour" [11]. Oreizy defines self-adaptive software as software that "modifies its own behaviour in response to changes in its operating environment. By operating environment, meaning anything observable by the software system, such as end-user input, external hardware devices and sensors, or program instrumentation" [12]. Similarly, the terms *self-adaptive system* vs. *autonomic computing* or *self-managing systems* [6, 13, 14] are based on the same central concept of software adaptation; although autonomic computing considers a broader scope of study and sees adaptation situated in a scale of software maturity levels, from basic to managed to predictive to adaptive to autonomic (see Figure 1). Refer to [15] for an interesting discussion on software systems and (self-) adaptation. Here the terms adaptive and self-adaptive are used interchangeably as this work is not concerned with debating terminological distinctions. Instead, fundamental concepts of (self-) adaptive systems that resemble the proposed work are presented together. This work adopts Salehie's definition of self-adaptive software as an encompassing definition of the terminology above described: software systems that "aim to adjust various artifacts or attributes in response to changes in the *self* and in the *context* of a software system. By *self*, meaning the whole body of the software, mostly implemented in several layers, while the *context* encompasses everything in the operating environment that affects the system's properties and its behaviour. Therefore, in this view, self-adaptive software is a closed-loop system with feedback from the *self* and the *context"[6]*.

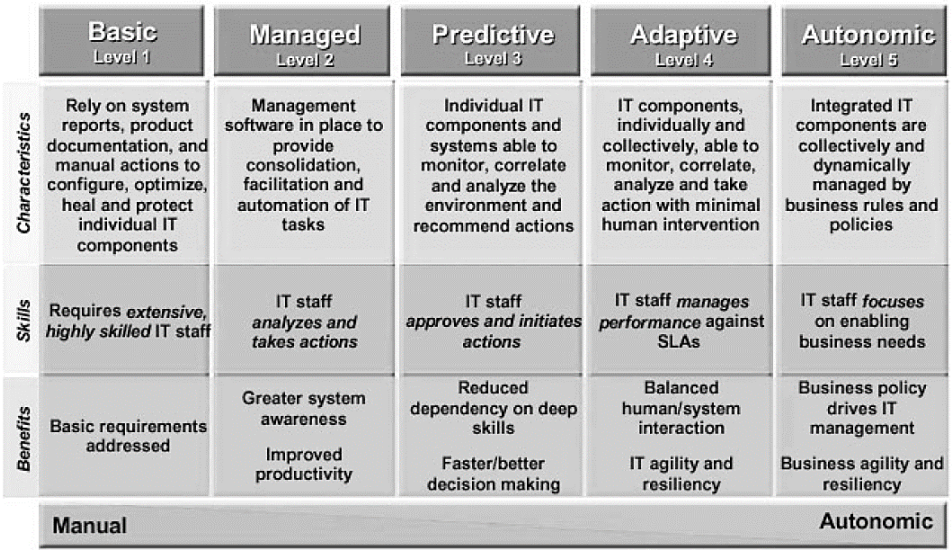


Figure 1 Autonomic Maturity Levels[16]

2.2.1 Conceptual architecture and adaptation loop

In 2001, IBM proposed the vision of autonomic computing [17] as a new paradigm to tackle the increasing difficulty in managing the complexity of modern computing environments: diverse, integrated, distributed, interconnected, etc. The objective of autonomic computing is to create computing systems that are self-managed (or self-governed) according to high-level goals defined by their administrator(s).

IBM proposed the *autonomic element* [18] as the core architectural component of self-managing systems. An autonomic element consists of two primary abstractions, namely an *autonomic manager* and *managed element(s)* – see Figure 2. The autonomic manager monitors the managed elements and their external environment using sensor interfaces and controls them using effector interfaces. This creates a closed control feedback loop that is decomposed into four main functions: *monitor, analyse, plan, and execute*; all supported by a *knowledge component*. This is known as the *MAPE-K loop*, or more generally, the adaptation loop. The *monitor function* collects, aggregates, correlates, and filters information from the managed elements, via the sensor interfaces, in order to determine potential symptoms that must be analysed. The *analyse function* takes as input these symptoms and determines if a change in the system is required; for instance, it may correlate a symptom with information from the knowledge component to determine if a policy is being violated. Complex data analytics algorithms can be implemented by this function. When a change is needed, the *plan function* produces a change plan, potentially from many possible alternatives, that needs to be applied to the managed elements. The *execute function* executes the actions required to fulfil the change plan via the effector interfaces. The *knowledge component* refers to a set of knowledge sources that are shared and used by the four functions of the control loop. It can consist of databases, repositories, etc., that contain information about policies, symptoms, change plans, and so on. The knowledge component can be instantiated with external knowledge defined by, for example, administrators but also the autonomic manager itself can update it.

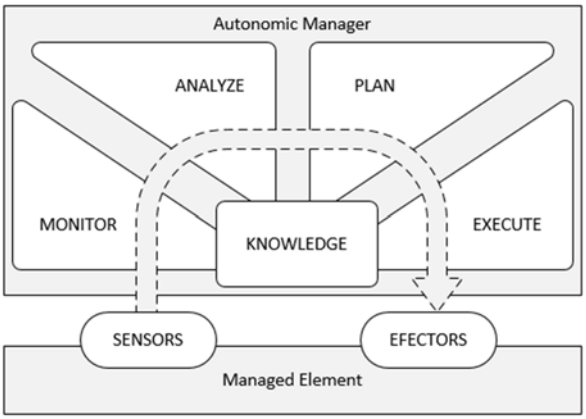


Figure 2 Autonomic Element

The MAPE-K loop is (arguably) the de-facto conceptual model used to describe the four functions (*monitor, analyse, plan, and execute*) of an adaptation process in general. The same four functions are presented in research literature with different terminology; for example, *monitoring, detecting, deciding, and acting*, in self-adaptive systems [6].

2.2.2 The Self-* properties

IBM cites four main attributes that emerge from self-management systems: self-configuration, self-optimisation, self-healing, and self-protection. More generally, Salehie et al. [6] categorises the properties of self-adaptive systems, known as the self-* properties, in a view hierarchy of three levels: *general, major, and primitive*. At the general level, self-adaptiveness relates to properties such as self-management and self-control. Self-management aims at freeing system administrators from low-level operational details of a system [17]. Self-control enables software to control itself during operation [19]. At the major level, Salehie situates the properties of autonomic computing [17]: "*Self-configuration* is the capability of reconfiguring automatically and dynamically in response to changes by installing, updating, integrating, and composing/decomposing software entities." [6] "*Self-healing* is the capability of discovering, diagnosing, and reacting to disruptions. It can also anticipate potential problems, and accordingly take proper actions to prevent a failure. Self-diagnosing refers to diagnosing errors, faults, and failures, while self-repairing focuses on recovery from them" [6]. "*Self-optimisation* is the capability of managing performance and resource allocation in order to satisfy the requirements of different users. End-to-end response time, throughput, utilisation, and workload are examples of important concerns related to this property" [6]. "*Self-protection* is the capability of detecting security breaches and recovering from their effects. It has two aspects, namely defending the system against malicious attacks, and anticipating problems and taking actions to avoid them or to mitigate their effects" [6]. Finally, at the

primitive level, there are two properties for self-adaptation: *self-awareness* – systems aware of their states and behaviours; and *context-awareness* – systems aware of their operational environment.

2.3 Policy-based Management

Policy-based management (PBM) is an administrative approach aimed at simplifying the management of a system. The main goal is to separate the policies governing the behaviour of the system from its functionality in order to improve the manageability, flexibility, maintainability, and runtime adaptability of the managed system [20].

The use of policies can be traced back to early seminal policy-based security models starting with access control models. Access control refers to the selective restriction of access to resources by enforcing authorisation decisions according to security policies. This essentially implies a management aspect. One of the earliest security models (proposed in the 60's) was access-control matrices [21] based on policies associating subjects, objects, and access rights, via N-dimensional arrays. Multiple important security models followed: the Bell-LaPadula model [22] consisting on the mapping of *security labels* to objects and *security clearances* to subjects through confidentiality policies; the Biba model [23] consisting of the same concepts of labels and clearances of Bell-LaPadula but used to enforce integrity policies; the Clark and Wilson model [24] that contributed to security policies the concept of access triplets (user, program, and files) for the first time; the Chinese Wall security model [25] that addressed *conflict of interest* issues through policies, e.g. subject A cannot do task B if A has previously done task C; and the Role-based access control (RBAC) model that associates *access permissions* to *roles* (i.e. not directly to users) and *roles* to *users* providing flexible access control management of organisational activities.

The security models cited above are some of the earliest models that used policies as a general paradigm. Although all of these models correspond to the area of security, the concept of policies is extensively used in the management of any resource allocation problem in general. Policies provide behavioural guides to systems about potential actions improving the system's behaviour [26]. The great significance of this paradigm is that it realises *the separation of mechanism and policy* as a design principle, i.e. the behaviour of a mechanism can be modified without requiring to recode the implementation of the mechanism itself.

During the 80's and 90's, as IT infrastructure was becoming increasingly and widely adopted, large networks and distributed systems started to be deployed, the internet expanded, and as a result the complexity in managing IT resources exploded. Diverse scenarios and management

requirements, including managing scalability, optimisation of resources, cost, revenue, performance, and quality of service (QoS), started to appear in areas such as security, business, and networks. These developments and the complexities they brought with them led to consider *the use of policies and their management* in more abstract ways.

In this direction, initial works include: the inter-organisational networks (ION) access control model by Estrin [27] who recognised the need of structuring policies at different abstraction levels such as high-level access policies in addition to the traditional network traffic policies, with the former defined based on attributes with semantics above the routing layer (e.g. file name at the application layer); and the contributions of Sloman et al [28, 29] who explicitly proposed policy-based management (PBM) as an scalable paradigm for the management of complex distributed systems. In their work, PBM is based on the concepts of *domains* and *management policies*. A domain is a set of objects with common attributes to which the same management policies apply. The domain model allows representing transitive, reflective, inheritance, joint, and disjoint associations among objects. Domains (and subdomains) allow structuring policies for the management of resources in organisations with different types of structures, i.e. hierarchical, supervisory, and inter-organisational. These seminal works established the initial foundations for policy-based management (PBM) as a paradigm, and since then research efforts in the PBM area have progressed and specialised in different directions covering diverse aspects. In particular, research efforts have concentrated in the areas of policy specification languages, management architectures, standardisations, tools and test-beds, and others.

Today, PBM is a central notion of many management models including networks [30], business-driven [17] and self-adaptive [6]. In the following subsections, important aspects and concepts of PBM of relevance to this thesis are reviewed.

2.3.1 Generic policy-based system management architecture

Figure 3 depicts the model of a policy-based system management architecture as defined by the IETF Policy Framework [31]. This model is the most commonly used today. It consists of 4 components: policy management tool, policy repository, policy decision point (PDP), and policy enforcement point (PEP).

The administrator inputs the policies into the system using the policy management tool. Then the policies are sent to and stored in the policy repository. The PDP searches and retrieves policies from the policy repository, interprets them and communicates them to the PEP. The PEP is responsible for applying and executing the policies.

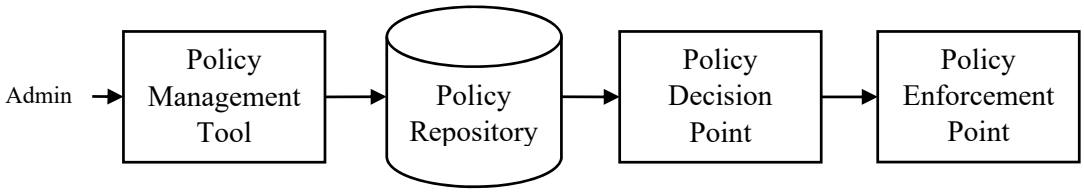


Figure 3 Policy-based Management General Architecture

This simple but general architecture partially mirrors and is consistent with the MAPE-K loop where the "a*nalyse"* and "*plan"* functions have a correspondence to functional aspects of a PDP and the "*execute"* functions to functional aspects of a PEP. Also, notice that the PEP would correspond to the configuration of a *managed element*.

Figure 4 shows a PBM architecture enhanced with a monitoring module that mirrors more closely the MAPE-K loop. In this case, the monitoring module dynamically provides information either to the policy repository in order to add, update and modify policies, or to the PDP for context-based evaluation.

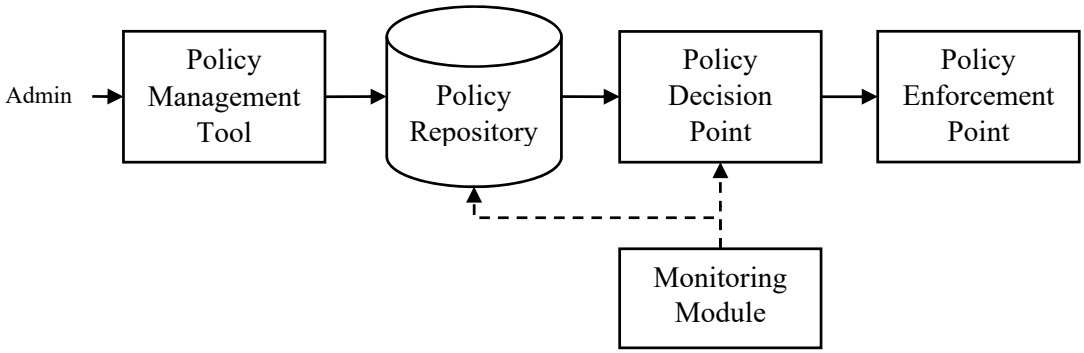


Figure 4 PBM architecture with monitoring module

2.3.2 Policy frameworks

This sections briefly reviews four important policy-based management frameworks: Ponder, Rei, KaoS, and XACML. Important aspects of the Rei, Ponder, and KaoS policy models and the XACML policy management architecture are highlighted.

2.3.2.1 Ponder

Ponder is a policy management framework consisting of the Ponder specification language, a general architecture, a policy deployment model, and extensions for QoS and access control management. The Ponder language is declarative, object-oriented, and allows specifying security and management policies for distributed object systems [32]. It supports specifying authorisation, delegation, information filtering, refrain policies, and obligation policies; and also supports the concept of domains for the grouping and partitioning of objects. In addition,

Ponder supports specifying meta-policies, or i.e. policies about policies. Meta-policies allow specifying management policies that scope other policies.

2.3.2.2 Rei

Rei [33] is a policy management framework developed by Hewlett Packard (HP). It is aimed at providing domain-independent policy specification. The policy language uses constructs based on deontic concepts. These constructs allow the domain-independent specification of rights, prohibition, obligations and dispensation policies, and the flexibility to easily add domain-dependent information without additional modifications. The Rei framework is ontology-based and incorporates a reasoner based on the F-OWL inference engine [34]. Ontology subclasses can be created and added to the existing hierarchical structure of Rei classes. Additionally, Rei supports meta-policies to describe the default behaviour of policies and constraints between meta-policies, e.g. precedence rules.

2.3.2.3 KaoS

KaoS provides an ontology-based language [35, 36]. In KaoS, system concepts are defined in ontologies and KaoS policies are expressed using the Web Ontology Language (OWL). This enables runtime extensibility, adaptability of the system, and policy analysis relating to entities at different levels of abstraction. KaoS supports 4 types of policies, each associated to an ontology class: positive authorisations, negative authorisations, positive obligations, and negative obligations; also new policy types can be created by defining new ontology classes. KaoS provides domain services for the hierarchical grouping and management of entities. Policies can be associated to and grouped by management properties such as e.g. priority.

Policy Frameworks / Features		Ponder	KAoS	Rei
General Characteristics	Policy specification, analysis and enforcement	✓	✓	*partial*
	Language's formal semantics and extensibility.	*partial*	✓	✓
	Domains and other forms of grouping	✓	*partial*	*partial*
	Distributed policy enforcement	✓	✓	✓
	Meta policy	✓	*partial*	✓

Figure 5 Policy frameworks comparison[37]

Each of the above frameworks provides its own policy language and covers several general PBM aspects, as shown in Figure 5, that include policy specification, analysis and enforcement; language's semantics and extensibility; domains for structuring and grouping, distributed policy enforcement, and meta-policies to ease management. The following subsection describes the XACML framework.

2.3.2.4 Extensible Access Control Markup Language (XACML)

The eXtensible Access Control Markup Language or XACML 3.0 [16] is a XML-based declarative access control policy language that provides interoperability for decentralised cross-domain policies. In addition to the language, XACML defines both an architecture for policy evaluation and a message exchange communication protocol. The primary architectural components are: the *Policy Enforcement Point (PEP)* that performs access control, makes decision requests and enforces authorisation decisions; the *Policy Decision Point (PDP)* that evaluates applicable policies and makes an authorisation decision; the *Policy Administration Point (PAP)* that creates/writes policies or policy sets; the *Policy Information Point (PIP)* that acts as a source of attribute values; and the *context handler* that translates decision requests and authorisation decisions into the XACML canonical format and into the native response format, respectively. The data-flow model is shown in Figure 6 and operates as follows:

- The requester makes an access request to the policy enforcement point (PEP) component (step 2), which enforces decisions made by the policy decision point (PDP).

- The PEP sends the request to the context handler component (step 3), whose function is to convert the request into a canonical format and passes it to the PDP (step 4).

- The PDP obtains from the policy administration point (PAP) component the applicable policy/policies. The PAP writes policies and makes them available to the PDP (step 1)

- The PDP requests any additional attributes required for policy evaluation from the context handler (step 5), which in turn requests the attributes from the policy information point (PIP) (step 6). The PIP obtains the requested (resource/environment/subjects) attributes (steps 7a, 7b, and 7c), and then sends them to the context handler (step 8).

- Optionally, the resource is included in the context (step 9). The context handler sends the attributes to the PDP (step 10).

- The PDP evaluates the policies against the attributes and sends a response context with the authorisation decision to the context handler (step 11), which in turn converts the response context into the native format of the PEP and send it to the PEP (optionally) including some obligations (step 12).

- The PEP fulfils the obligations (13) and then grants/ denies access to the resource.

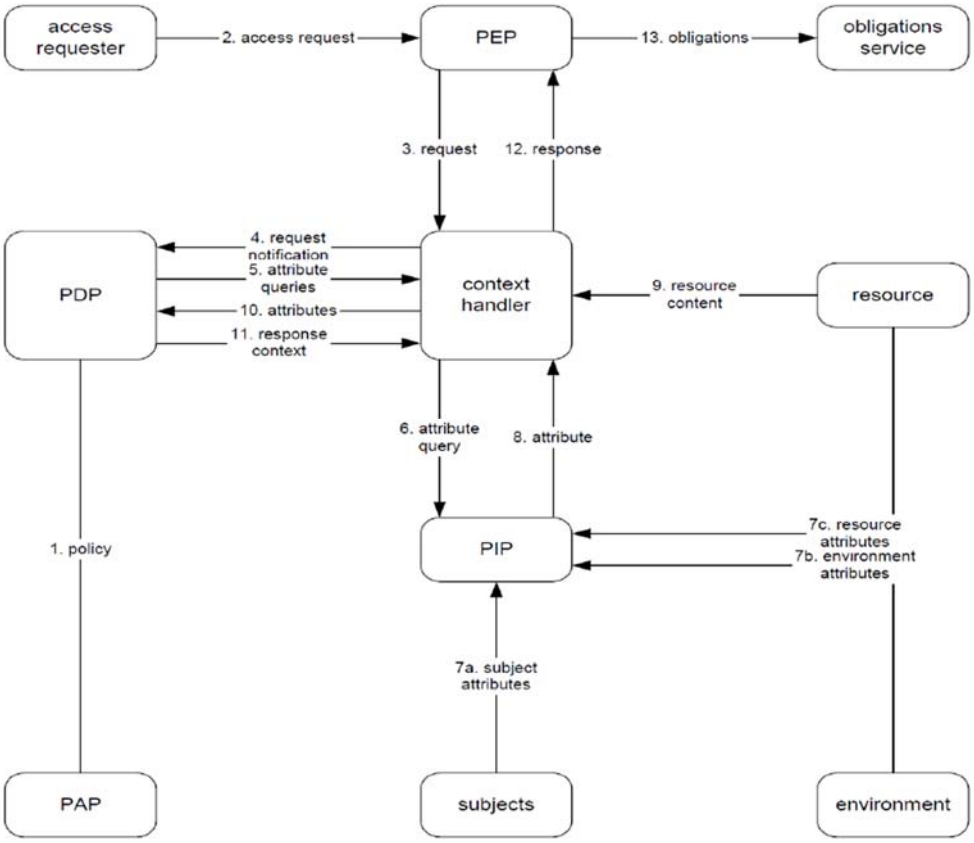


Figure 6 XACML Data-flow diagram

Ponder, Rei, KaoS, and XACML are representative frameworks in the area of PBM. However, research in PBM encompasses more than the aspects covered by the abovementioned frameworks. Looking into the lifecycle of management policies provides a broader view of what PBM entails.

2.3.2.5 Management policies lifecycle

In [38], the authors present a methodology for the management of management policies. They propose a policy lifecycle model that aligns with software engineering concepts – see Figure 7. It consists of several phases that include: *system requirement analysis* to support systematic policy definition and the identification of constraints; *policy definition and specification* at a high-level of abstraction written in natural language; *policy refinement* to transform high-level policies into low-level enforceable policies; *policy analysis* covering syntactic and semantic validation, conflict detection and resolution, and completeness and feasibility validation; *policy distribution and enforcement*; and *policy monitoring and maintenance* covering logging, policy violations checking, auditing, and policy recovery.

Figure 7 depicts the relationships that exist among the different phases above mentioned in terms of process flows, data flows, and reverse flows.

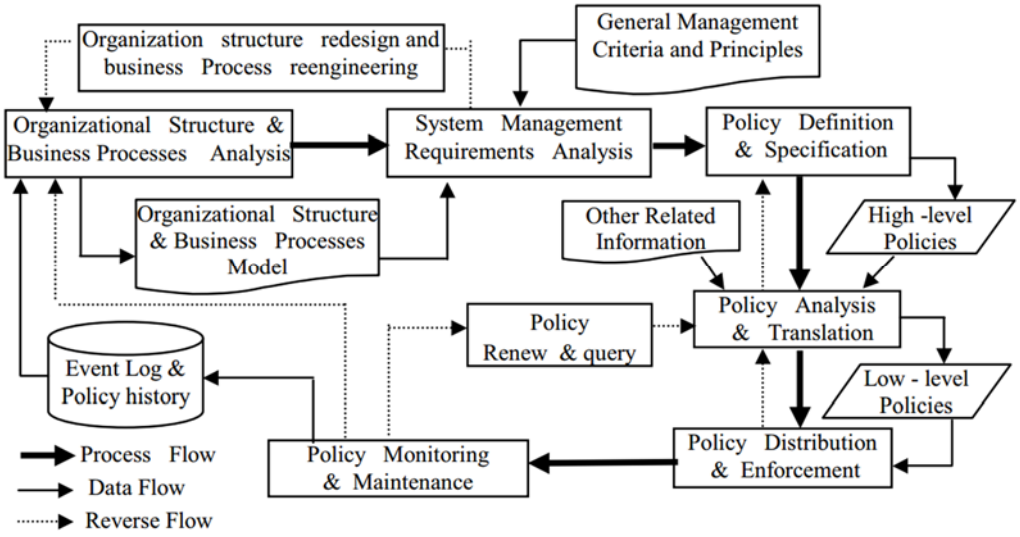


Figure 7 A Policy lifecycle model[38]

This research focuses primarily (in varying levels of detail) on the operational aspects (rather than semantics and syntax) of policy definition and specification, policy refinement, policy enforcement, and policy monitoring (in Chapter 4) phases; and proposes a method for policy transformation for adaptation.

2.3.3 Policy-based Management Terminology

The following is a list of PBM-related terms used in the thesis.

2.3.3.1 Policy

The IETF RFC3198 [39] defines *policy* from two perspectives:

- "A definite goal, course or method of action to guide and determine present and future decisions. "Policies" are implemented or executed within a particular context (such as policies defined within a business unit)" [39] ;
- "Policies as a set of rules to administer, manage, and control access to network resources" [39].

2.3.3.2 Policy abstraction

According to the IETF RFC3198, "policies can be represented at different levels, from business goals to implementation-specific configurations. Translation between different levels

of "abstraction" may require information other than policy. Various documents and implementations may specify explicit levels of abstraction"[39].

2.3.3.3 Policy hierarchy

PBM systems use the concept of policy hierarchies primarily to facilitate the automated processing of policies. A policy hierarchy defines the hierarchical relationships between policies in order to decide the requirements necessary that satisfy the policies at different levels of abstraction [40]. For example, if a high-level policy changes then the policy hierarchy can be used to determine what lower-level policies must change or be created. Policy hierarchies are also used for policy analysis, for example, to determine if low-level policies do not violate high-level policy goals.

2.3.3.4 Policy translation

Policy translation consists in the transformation of a policy from one level of abstraction or representation into another level of abstraction or representation [39]. Policy translation is also referred to as policy mapping or policy conversion. For example, consider a mobile device communicating with different policy decision points (PDP) for the allocation of resources while roaming. The mobile device may be required to convert the format of its policies to particular PDP's technologies.

2.3.3.5 Policy refinement

Policy refinement consists in the transformation of a policy from a high-level of abstraction or a high-level representation into a lower-level of abstraction or a lower-level representation. For example, in business driven policy management, goal policies target the long term high level business and can be refined into low-level policies for components [41, 42]. A low-level policy is usually in the form of *event-action rules*.

2.3.3.6 Policy Transformation

The concept of policy transformation is more general and encompasses the definitions of policy translation and policy refinement. An additional concept that falls under policy transformation is that of *policy integration* in which policies from different issuing sources are combined while maintaining consistency across the set of systems being managed. According to [43], a policy transformation process must consider different properties of the language such as lexicon, syntax, semantics, and the context.

In [41], policy transformation techniques are discussed to simplify and automate the administration of IT infrastructure. The simplification is obtained by allowing a system

administrator to specify only the system goals that are to be met, instead of having to specify detailed low-level system configuration parameters. This is achieved with different types of policy transformations and methods for the mapping from objectives to system configurations. Policy transformations are classified as static or real-time. The former correspond to predetermined static rules that determine policy transformation while the latter correspond to dynamic rules dependent on system behaviour. Moreover, solutions to the problem of policy transformation are distinguished between: analytic models, online adaptive control, and simulation approach. The first case relies on the existence of an analytic model that can be used to determine business objectives as a function of configuration parameters. The online adaptive control makes use of concepts of control theory [44] for controlling and tuning configuration parameters, or neural network models [45] for determining the impact of configuration parameters. The simulation approach is used to simulate the behaviour of the system and to build a scheme for transforming policies.

The concepts described in the previous two sections on adaptive systems and policy-based management constitute the theoretical foundation on which the proposed methodological framework is built on.

2.4 Related Work

In this section different approaches in policy-based management and policy adaptation directly related to the proposed framework are discussed and compared. Although the works selected cover diverse research areas and may in some cases address different research problems, they also cover the most fundamental concepts for the realisation of policy-based adaptation. The objective is to describe these works within their topical context and to contrast them against the work in this thesis in order to elucidate requirements in the light of policy transformation as the mechanism for adaptation.

2.4.1 Real-time policy transformation techniques for PBM systems

In [41], a PBM approach is proposed to simplify and automate the administration of IT infrastructure. The simplification is obtained by allowing a system administrator to specify only the system goals that are to be met, instead of having to specify detailed low-level system configuration parameters. The authors propose a real-time policy transformation approach based on *case-based reasoning* that consists in using a case database or history of system behaviour to provide an experimental basis for transforming from high-level policies into low-level configurations. This is represented in Figure 8. A monitoring module senses the behaviour of the system and a policy transformation module uses the sensed behaviour to

compare against a case database in order to select an appropriate policy configuration. Configured policies are then fed into the policy repository to be used by a policy decision point.

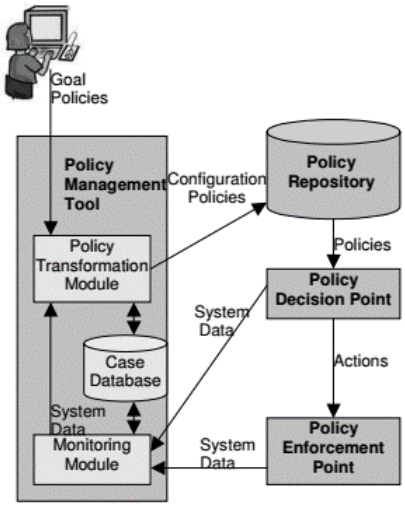


Figure 8 A policy-based management system[41]

This work uses of a policy transformation module to transform high-level policies into low-level executable policies. However, there is not a policy hierarchy explicitly defined. Although this work is mainly a PBM approach, it considers aspects of adaptive systems such as the use of a monitoring module. The approach is limited to the selection of policy configurations to be applied based on the input from the monitoring component, but such input is not used to change (i.e. specialise) the policy transformation process itself, i.e. dynamic runtime generation of newly created deployable policies. The framework proposed in this thesis incorporates runtime specialisation of policies.

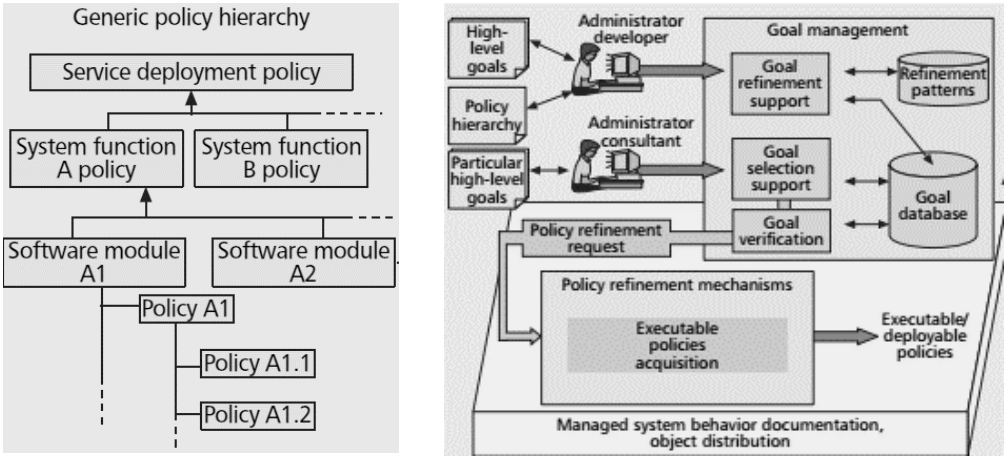
2.4.2 Methodological approach to the refinement problem in PBM systems[46]

This work [46] proposes a methodological approach to policy refinement consisting of three main aspects:

- *A policy hierarchy* that mirrors the system architecture (see Figure 9(a)). From top to bottom, the hierarchy consists of the following levels: a "service deployment policy" that corresponds to the service provided by the system, "policy-controlled system functions", "policy-controlled software modules" that correspond to system components, and "policy-controlled sub-functions" that specify specific types of policies.
- *The identification of high-level goals.* Two actors are considered: the developer and the administrator. The developer defines the high-level goals and derives different policies

that can achieve them. The administrator interprets those high-level goals to define particular goals during operation.

- *A framework that performs the refinement process* consisting of two main functions: goal management and policy refinement mechanisms (see Figure 9(b)). The first function allows refinement driven by high-level goals (explained above in identification of high-level goals), and as prescribed by the policy hierarchy. The second function refines the high-level goals selected by the administrator into executable policies and deploys them.



(a) (b)

Figure 9 (a) Generic policy hierarchy, and (b) Goal-oriented policy refinement framework

Here, the policy hierarchy is primarily used to associate policies to functions and software modules of the system. At the different levels of the hierarchy, the approach is only concerned with a top-down refinement of management goals and policies; therefore, the approach exploits *is-a* and *composition* associations of the hierarchy. In the approach proposed in this thesis, a policy hierarchy is used in similar fashion, however extended to consider the concern of adaptation in addition to the potentially existing associations of the system hierarchy. The work in [46] is not intended to consider adaptation aspects nor external entities introducing constraints in the refinement process.

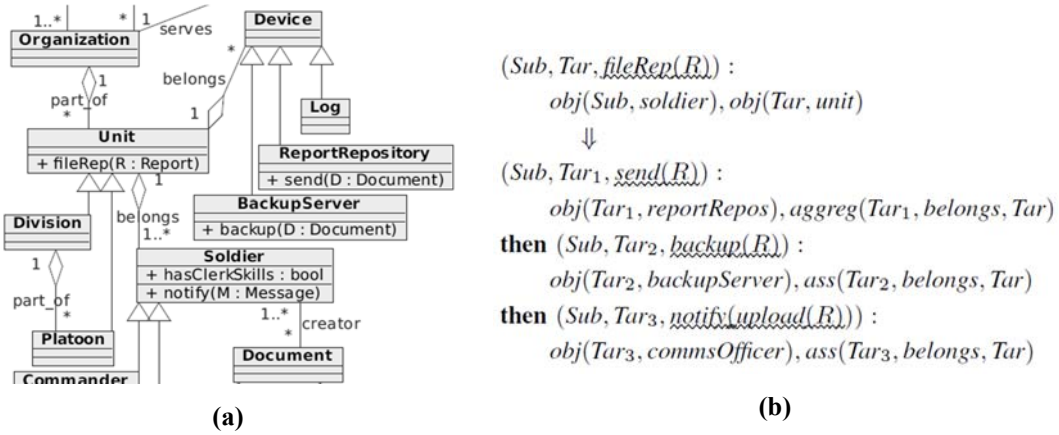
2.4.3 A model based approach for policy tool generation and policy analysis

Barrett et al. propose in [47] an information model-based approach for policy specification and analysis applied to an integrated suite of languages and tools supported by an ontology. The suite supports separate domain-specific languages (DSL) and editors for policy specification (e.g. Ponder, Rei, and KaoS). Regarding policy analysis, the approach addresses policy conflicts (i.e. detection and resolution) and policy transformation using ontologies. The use of ontologies allows reasoning about and comparing different languages' syntax and semantics [43] and involves identifying similarities and differences between policies.

However, in [47] policy transformation only considers the mapping aspect between different representations of policies and is not intended to address adaptation of policies per se nor the incorporation of external requirements via operational environment in the policy transformation process. Moreover, different levels of abstraction during policy transformation are not treated. In the proposed approach of this thesis, three levels of abstraction are specified: management, adaptation, and implementation. Policy transformation is encapsulated as the concern of the adaptation level.

2.4.4 Decomposition Techniques for Policy Refinement [48]

This paper [48] presents a policy refinement framework for authorisation and obligation policies. The problem of policy refinement is described as consisting of three aspects: decomposition, operationalisation and distribution. In decomposition, high-level policies are mapped to low-level policies using policy-independent *refinement rules*. Operationalisation associates abstract policy classes with specific subjects, targets, and actions obtained from the system model. The system model specifies and scopes the policies that can be defined. Distribution covers aspects such as confidentiality and policy conflicts, particularly for distributed scenarios.



(a) **(b)**

Figure 10 (a) UML system model[48], (b) Refinement rule example[48]

Policies and refinement rules are represented using event calculus [49] – a subset of first order logic. The system model is represented using UML class diagrams. Figure 10(a) shows an example of a military scenario. Refinement rules are defined in a process referred to as *action decomposition* from high to low-level actions and exploiting the UML class associations (i.e. aggregation, generalisation, composition, etc.) of the system model. Figure 10(b) shows an example of a refinement rule where the action *send(R)* is decomposed into *backup, notify,* and *upload* (see underlined actions) where R stands for *report*. The policy decomposition process consists of two phases. The *matching phase* verifies if a refinement rule is applicable to a

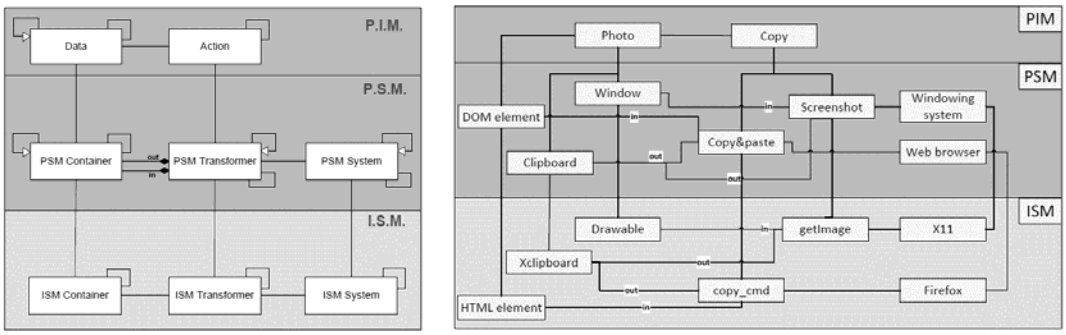
policy (e.g. an authorisation policy) while the *decomposition phase* performs the refinement as defined by the applicable refinement rule.

Similar to the decomposition of policies based on refinement rules in this work, as it will be presented, this thesis' proposed framework allows for top-down policy refinement by introducing semantics in the refinement process at three levels of policy abstraction, namely, management, adaptation, and implementation, with mapping of policies between levels. However, the proposed framework also ties functions to each level of abstraction (e.g. adaptation-level functions), which can be dynamically configured by their corresponding policies. Moreover, the adaptation policy that configures its corresponding adaptation-level functions is the result of a policy transformation process performed at runtime taking as input baseline policies of the system and policies introduced by external entities in the operational environment.

2.4.5 Model-Based Usage Control Policy Derivation [50, 51]

This work [50, 51] presents a model-based usage control policy derivation (i.e. policy refinement) method. Usage control refers to how data is used after access to it has been granted. Usage control policies are translated from specification-level to implementation-level by applying refinement on *data* and *action* parameters. For instance, an end-user discloses some document and expresses the following usage policy: "delete document after 7 days". Such policy is expressed using high-level semantics at the specification-level. However, in order to enforce such policy, the action "delete" and the data element "document" need to be mapped to their technical counterparts at the system implementation-level, for example, *file_xyz* and the command *remove*.

The authors propose a domain meta-model consisting of the following 3 layers as shown in Figure 11(a). The *platform independent (PIM) layer* corresponds to end-user high-level actions on data such as "copy photo". Below the PIM layer data and actions are represented and referred to as *containers* and *transformers,* respectively; the *platform-specific (PSM) layer* corresponds to implementation-independent transformers on containers such as "take screenshot". These are technical-level abstractions but not tied to specific implementations; finally, the *implementation-specific (ISM) layer* corresponds to low-level transformers such as "getImage()" in the X11 windowing system and low-level containers such as "HTML element". Figure 11(b) shows an example. The domain meta-model provides the semantics for each layer and also the mappings required between data and containers and between actions and transformers by means of UML associations.



(a) (b)

Figure 11 (a) Domain meta-model[50], (b) Meta-model example[50]

For policy specification the method incorporates a usage control model based on OSL (Obligation Specification Language) [52] extended with constructs for handling containers and transformers [53]. The model constructs allow semi-automated refinement and the configuration of enforcement mechanisms at the ISM layer.

Although belonging to the area of usage control, the work describe in [50] is very influential in the framework proposed in this thesis, i.e. it clearly defines a hierarchy of abstract levels with clear semantics and associated functions expressed as *actions* and *action parameters*. However, it is not intended to address dynamic adaptation of policies. As explained at the end of the previous subsection, the framework proposed in this thesis encompasses a more complex policy transformation process at the adaptation level that goes beyond the mere mapping of policies between different levels of abstraction.

2.4.6 An Adaptive PBM Framework for Network Services Management [54]

This paper [54] proposes a framework for the dynamic adaptation of policies in response to changes within a network-managed environment. Figure 12 depicts a PBM system for service management. The technical parameters of service level agreements (SLAs) are specified as service level specifications (SLSs) in terms of network information. To support dynamic service management, according to the particular adaptation strategies, a *management system* receives the network information from the SLS processing component and modifies the relevant policies when changes are required, and applies them to the managed network.

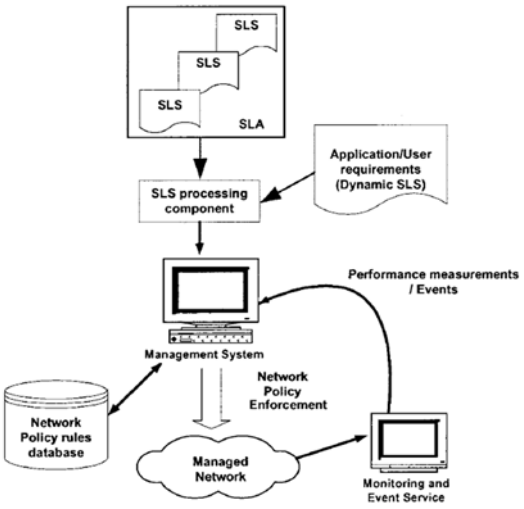


Figure 12 Service management with a PBM system[54]

The framework considers the following scenarios in which policies may need to be changed (see Figure 12): a user or an application requests changes to the QoS provided, performance degradation is detected by service monitoring components, and network failure events are triggered. For policy definition, the framework uses the Ponder policy language [32] and considers *obligation policies* of the form of event-triggered condition-action rules. The authors use the term *policy adaptation* to refer to the ability of the PBM system to alter the behaviour of the managed network in two ways: by dynamically changing policy parameters and by selection and enabling/disabling a policy from a set of predefined policies:

- Dynamic modification of policy parameters. *Parameter events* trigger the calculation of new parameters which in turn are used to configure the relevant policy actions on the target network objects.
- Dynamic selection/enabling of policies from a set of predefined policies. *High-level control policies* triggered by reconfiguration events determine what low-level policies to enable or select. By modifying or adding high-level policies it is possible to introduce new adaptation strategies without requiring to modify the low-level ones.

The framework provides an enforcement architecture for policy adaptation (see Figure 13). The functionality is as follows. First, the *Service PMA* (Policy Management Agent), which interprets service management policies, receives an adaptation request from the *event service* requesting a change (step 1). The Service PMA requests the current policy database from the *Policy Service* (steps 2 and 3) and uses it to select a selection algorithm. The selection algorithm determines what network policy to apply to the Network Level PMA, which interprets network device configuration policies (step 4). Finally, the Event Service receives the corresponding *obligation event* from the Service PMA (step 5) and distributes it to all network-level PMAs (step 6).

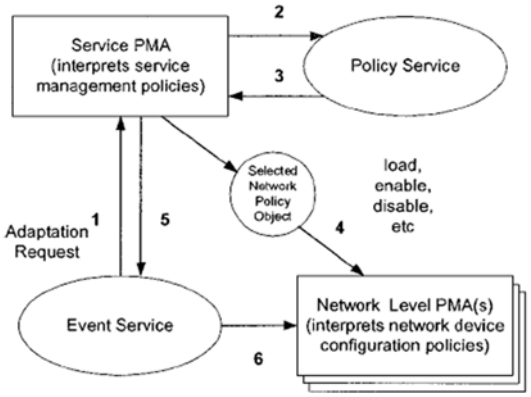


Figure 13 Enforcement architecture for policy adaptation[54]

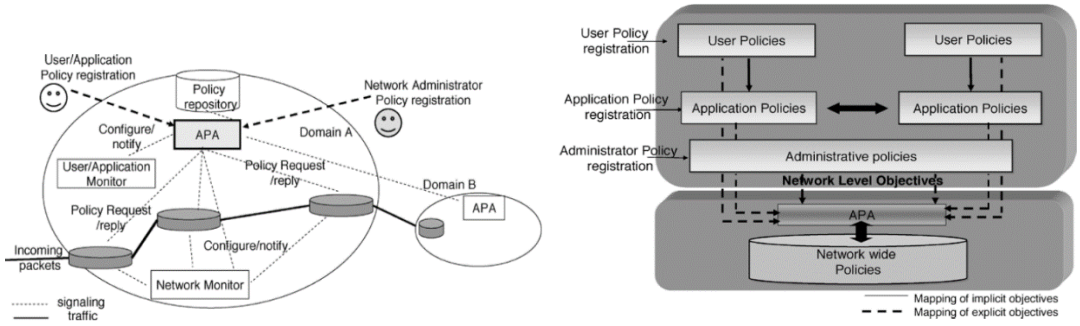
The work in [54] is very relevant to this thesis, however the framework proposed in this thesis differs in one main aspect. It goes beyond the notion of an *adaptation request* made by external entities. Instead it treats such "adaptation requests" as external policies that reflect behaviour initiated by external entities that may correspond to a direct request to the component that performs the adaptation logic or instead correspond to behaviour observed by adaptation component in its operational environment. For example, in Chapter 4, different administrative external entities define policies on their own software applications. Such applications can interact among them but within a scope defined by the baseline policies of the adaptation component. When two applications interact, the adaptation component dynamically generates an executable policy resulting from the combination of the two applications' policies, but still constrained by any applicable baseline policy of the system. This is different to the type of *adaptation request* described in [54]. The proposed framework in this thesis allows incorporating external requirements in the form of policies during the adaptation process.

2.4.7 An Automated Policy-Based Management Framework for Differentiated Communication Systems [55]

This framework [55] is intended to automate the management of differentiated communication systems. It focuses on the adaptation of policies in response to changing requirements of users and applications. The authors highlight the limitations of PBM techniques that require a priori policy configurations to manage network devices, and propose policy adaptation by using learning techniques. The adaptation of policies is not based on adapting policy parameters, but on the dynamic creation of policies at runtime instead.

The framework proposes a key component called *automated policy adaptor* (APA) that decouples two tasks:

– The mapping between high-level goals and low-level network objectives. Users and applications specify requirements at different layers by means of high-level policies.
– The functionality of adapting the behaviour of network components.



(a) (b)

Figure 14(a) Proposed PBM framework[55], (b) Proposed APA policy hierarchy model[55]

For the mapping a policy hierarchy model is used, see Figure 14(b). The model allows network administrators to define business objectives, and users and applications to specify requirements related to service parameters, e.g. cost. Business objectives and requirements are expressed in the form of policies. The result of the process are *network level objectives*.

The adaptation functionality is an automated process where the APA interacts with edge and core routers to create, adapt, and enforce existing policies according to the network level objectives, see Figure 14(a). Three types of policies are supported. Admission control policies identify the ingress and egress points of network traffic and check that the resources requested can be accommodated. Traffic conditioning policies are assembled at runtime by the APA with the configurations for each particular request and applied to the corresponding routers. Provisioning policies ensure no bottlenecks are created in the network. The adaptation process is event-triggered either periodically or via monitoring components according to the specified network level objectives, and the network traffic requests.

Although in the communications area, the work in [55] provides an important insight of dynamic policy transformation. Similar to the work proposed in this thesis, this approach allows for dynamic runtime creation of new policies to be applied to low-level elements (i.e. network components) based on the combined input of management-level policies and external entities' policies. However, in the approach proposed in this thesis the problem of policy transformation, although in the security area, is treated at a more general level following a model-driven approach that considers a methodological framework that covers design, implementation, and operational aspects. This allows modelling the policy transformation process, in addition to the mapping between policies and the integration of policies

(management and external entities), to be extended by incorporating additional complementary security models at the adaptation-level.

2.4.8 Dynamic, context-specific SON management driven by operator objectives [56]

This work[56] proposes a framework for Self-Organising Networks (SON). SON is a mobile networks management approach based on the use of independently acting functions, each one dedicated to specific management tasks such as optimisation and configuration. SON functions act autonomously by adjusting network parameters in order to optimise the Key Performance Indicators (KPI) of the network (e.g. capacity). SON functions are configurable depending on predefined the KPI targets. This usually requires manual intervention. The framework described in this work is aimed at closing this manual intervention gap. It proposes the use of a SON Objective Manager to perform an automated transformation of context-dependent KPI targets into SON function configurations. The approach is based on the mapping of all the possible KPI contexts to the best possible SON function configurations. During runtime a Policy System selects the most appropriate configuration.

This approach implicitly defines a policy hierarchy where KPI are expressed as high level technical objectives mapped to low-level SON configurations. Both the KPI and the SON functions can change dynamically based on contextual information creating a closed control loop. However, different to the framework described in this thesis, the framework does not allow for the explicit introduction of external policies and constraints by external actors. Also, KPI and SON functions, although reconfigurable, are predefined. In this thesis, new functions and high level objectives can be created dynamically at runtime.

2.5 Concluding Remarks

This chapter presented an account of important definitions and concepts related to the areas of (self-) adaptive systems and policy based-management (PBM) systems. The separation of concerns between the adaptation logic and application logic aligns with the concept of *autonomic element* proposed by IBM, i.e. autonomic manager and managed element(s).

The motivations for the evolution from policy-based systems to policy-based management has been studied along with the reasons for the need to consider the use of policies and their management in more abstract and structured ways. In the area of policy-based management, the concepts of policy hierarchies, policy translation, policy refinement, and policy integration, are fundamental to understand how a PBM system is structured and how it is managed at different levels of policy abstractions. One of the most studied topics in PBM is

the problem of policy refinement which has gained increasing interest in the last decade. However, policy refinement is primarily concerned with the mapping from high-level goals to low-level policies. A more general concept is that of policy transformation (although in literature sometimes purely policy refinement approaches are also referred to as policy transformation) which implies much more complex transformation processes.

The approaches discussed in section 2.4 (Related work) focus mainly in mechanisms for policy refinement and the use of policy hierarchies. In addition to supporting such similar mechanisms, the work described in section 2.4.7 is the one that closest addresses the problem of runtime dynamic generation of new policies to drive adaptation (which is the primary focus in this thesis). However, the generation of adaptive policies is seen from the perspective of flexible business objectives capable of accommodating requirements introduced by external entities and aiming to satisfy them in terms of business metrics. This perspective is valid and complementary to the one proposed in this thesis but not the same. In this thesis, the dynamic generation of new policies is seen from a security angle where baseline (i.e. management) policies are intended to provide flexibility to external entities for the definition of requirements, but at the same time are intended to enforce a secure context of execution by constraining external entities' behaviour. This difference introduces an additional requirement (and certainly an opportunity) to the adaptation process. Modelling an adaptation process that encompasses security aspects at an abstract level allows the introduction of security models and functions in the policy transformation process resulting in low-level policies that reflect better security decisions on the target system. This is the focus and a novelty of this thesis.

Chapter 3

The Requirements of a Policy-Driven Adaptive Protection System

This chapter provides an overview of the proposed model for a policy-driven adaptive protection system and defines its system requirements. These requirements are extracted from the definition and analysis of the research problem and address the main research objectives of this thesis (see section 1.2.2). The last part of this chapter describes the interrelation of the following chapters in providing experimental foundation for the detailed proposition of the methodological framework and how the presented requirements will be addressed in the framework.

3.1 Overview of the Proposed Model for a Policy-Driven Adaptive Protection System

This research investigates *policy-driven protection systems* and a methodological framework for the realisation of policy-based systems capable of *specialising* their behaviour by *adapting* their *execution* in response to constrains introduced by external entities in the *operational environment*. Figure 15 shows the conceptual model proposed.

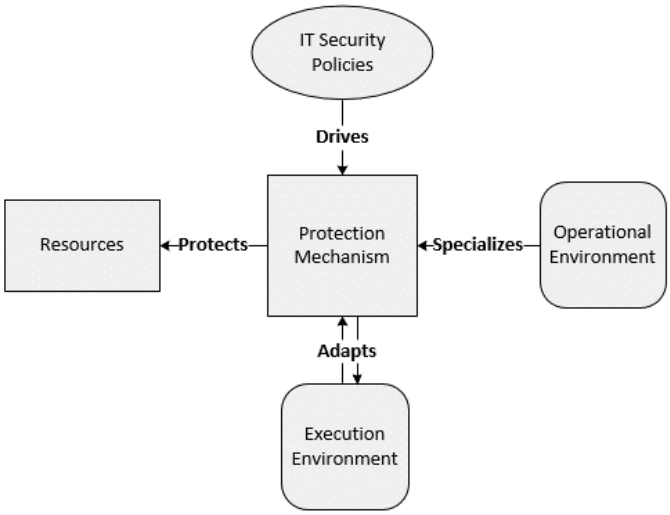


Figure 15 Overview of a Context-aware Adaptive Protection System

The proposed methodological framework builds on top of fundamental concepts and experimental work in two main areas of research: (self-) adaptive systems and policy-based management (PBM).

Most of the research efforts in areas related to (self-) adaptive systems concentrate on approaches that take the view of a system with certain level of autonomy that goes through a decision-making process in order to adapt. Such system is decomposed into two components, the *application logic* and the *adaptation logic*. This is shown in Figure 16.

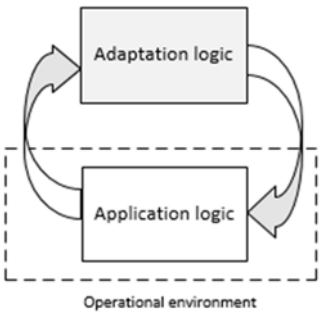


Figure 16 Adaptive system

The adaptation logic uses sensors and effectors forming a closed feedback loop that monitors the operational environment, detects relevant behaviour, makes adaptation decisions, and triggers configurations and actions to be enforced by the application logic [57]. For example, a network administrator defines administrative policies and makes them available to the adaptation logic. The adaptation logic, in turn, monitors the environment and uses the policies to decide how to dynamically configure network devices.

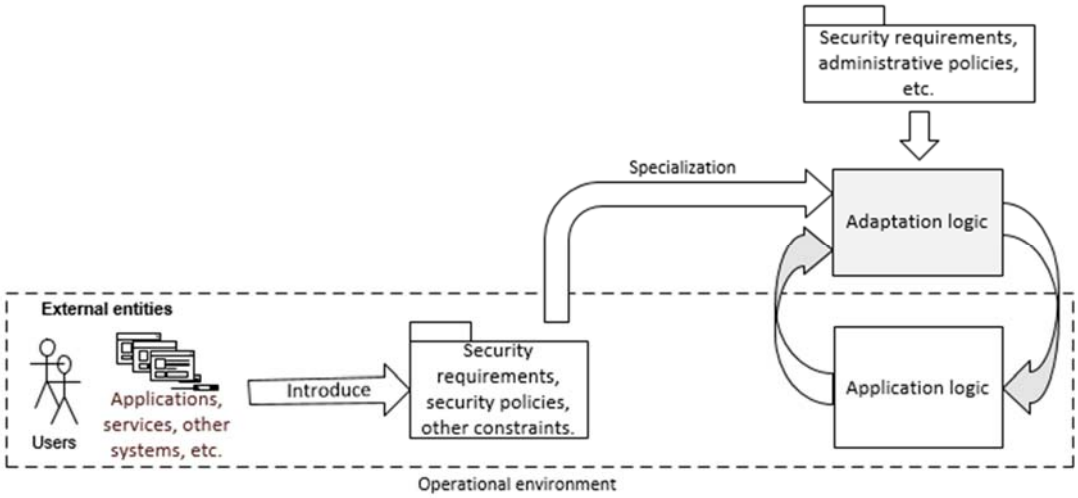


Figure 17 Specialisation in the adaptation logic

Figure 17 depicts this type of scenarios. Here, in addition to the traditional closed feedback loop, there is an influence caused by the security requirements, security policies, or any other constraints introduced (explicitly or implicitly) by external entities. Such constraints have a direct impact on the decision-making process at the adaptation logic level, that is, on the adaptation decision as to what strategy, configuration, or policy to apply to the application logic. This is very different to evaluating and enforcing, for example, a (context-based) security policy already applied at the application logic level.

In order to achieve specialisation, the adaptation logic must be able to somehow integrate the security policies, security requirements, or any other constraints being introduced into the system by both system administrators and external entities. As it will be described in detail in chapter 7, this thesis proposes a policy-based management approach for the integration and harmonisation of constraints by means of *policy transformation*. In essence, the policy transformation process takes as input both management and external constraints (in the form of policies) and outputs a resulting executable policy to be applied to the application logic.

At first, the property of specialisation appears not to be substantially different over a traditional adaptive system that can already detect external constraints via its monitoring components to process them. There is a subtle but overlooked core difference. In the case of specialisation, the transformation process requires to be modelled and designed in such a way that it understands and treats external constraints as policies and drivers for adaptation.

3.2 System Requirements Definition

The following subsections present the research questions related to the creation of a framework to build adaptive protection security systems. Based on results of previous work

presented in the literature and in the novelties aimed for the systems (e.g. systems capable of adapting to changing operational contexts and environments by specialising their behaviour in response to the changing security requirements of external entities/actors), seven system requirements are extracted in order to address the research questions. The research questions define the overall research problem this thesis addresses.

3.2.1 Policy Hierarchy and translation / mapping of policies

The research objective *Obj. 5* (section 1.2.2) addresses the need for a policy-based mechanism for the management and operationalisation of the proposed system. In order to fulfil this research objective, a policy hierarchy and the corresponding translation and mapping of policies across the hierarchy have been identified as necessary requirements. These requirements have been identified based on the study of the current state of prior research (*Obj. 1* - section 1.2.2). Objective *Obj. 5* addresses the following research question:

Can a general method be devised that enables adaptation in response to dynamically changing management and external constraints at the adaptation logic level, and the mapping of those constraints to executable policies at the application logic level?

Analysis:

Generally speaking, a security infrastructure consists of two main elements: a *security policy element*, which drives the decision-making process; and a *protection mechanism element*, that executes the security policy enforcement process. At enforcement level, protection mechanisms are concrete or implementation-specific components that are situated within the IT infrastructure and are usually operated via domain-specific low-level security policies. For example, at the network layer, a protection mechanism may consist of a firewall and/or an intrusion detection system (IDS) components, each one with their own type of policies: firewall rules and intrusion prevention/detection rules, respectively. This type of policies, hereafter executable policies, are extremely important because they can depend on a protection mechanism's state along with the semantics of that protection mechanism's abstractions [58].

However, working with executable policies presents several difficulties such as maintenance, consistency, and verification [59]; it requires domain-specific expert knowledge to define them; and it can be challenging to align them with high-level IT, business, or organisational security goals. In addition, because executable policies are constrained to the implementation level it is difficult to make them adaptive and dependent on contextual information sourced from outside the scope of the protection mechanism itself. One approach to overcome these difficulties is to devise a translation process from high-level abstract security policies,

hereafter abstract policies, into executable policies. By *abstract policies* is meant policies that cannot be specified directly at implementation level because of their high-level semantics. However, abstract policies allow defining the overall goal(s) and behaviour that the protection mechanism must meet, they can be extended or augmented with contextual information in order to improve security decisions, and they can be used to (re-)configure executable policies and, directly or indirectly, to improve the expressivity of the latter. For example, consider the executable policy of a firewall component "allow network traffic for port 3389 and IP address 192.168.0.1" and an abstract policy of the overall system "deny network traffic for ports 1000-5000 when the risk level in the system crosses a certain threshold". Here, the executable policy is valid as long as the risk level is below the threshold and can be enforced at firewall level. Importantly, the risk parameter is defined outside the scope of the firewall by the abstract policy, allowing for better expressiveness and fulfilling security decisions. However, for the translation process to be practical, the adaptive protection mechanism should be able to react dynamically to changes in abstract policies and reflect those changes in the way executable policies are enforced; additionally, it could be the case that abstract policies do not change, but runtime changes in the execution context might trigger different executable policies that should be consistent with the abstract policies.

Requirements:

1. To provide a policy hierarchy that facilitates system management, provides different policy abstraction levels, and enables different semantics for expressing concerns separately, i.e. implementation-specific, adaptation and management.
2. To provide consistent transformations and mappings between the policies of the policy hierarchy.

3.2.2 Three-Layer Architecture: management, adaptation, and implementation

The research objective *Obj. 4* (section 1.2.2) addresses the need for a general architecture that provides the components required to structure the adaptive system at different levels of abstraction. In order to fulfil this requirement, a layered architecture has been identified as a necessary requirement. From the study of the current state of prior research (*Obj. 1*), two levels of abstraction have been identified in the area of (self-) adaptive systems, i.e. adaptation and implementation; and two levels of abstraction in the area policy-based management, i.e. management and implementation. This research proposes a three-layer architecture, i.e. management, adaptation, implementation, in order to enable separation of concerns. Objective *Obj. 4* addresses the following research questions:

Where does the decision making process take place the across layers of a (self-) adaptive system? and how can this be achieved in a coordinated and practical way?

What mechanism(s) can enable the harmonisation and scoping of policies across the different layers?

Analysis:

In order to analyse how adaptation is introduced into the overall system, the adaptive protection mechanism is decomposed into two components, the protection mechanism logic (or application logic) and the adaptation logic. There are different facets to how to incorporate adaptation into a protection mechanism [6]. Some of them are considered below:

Static/Dynamic Decision Making. This refers to how the decision making process can be modified. In the static option, the decision-making process is hardcoded and requires recompiling and redeploying the protection mechanism or some of its components. In the dynamic option, policies [1, 60], rules [61] or QoS definitions [62] are externally defined and can be modified during runtime to produce new behaviour. This research is concerned with protection mechanisms where the decision-making process as to how to adapt can be achieved dynamically. This is a natural choice since it is policy-driven mechanisms being considered.

External/Internal Adaptation (see Figure 18). This refers to the separation of the *adaptation logic* from *the application logic*. In the internal approach, the application logic and the adaptation logic are combined together [63]. The external approach uses an external *adaptation engine* that deals with the adaptation logic and can be implemented by means of middleware [63], a policy engine [64], or other application-independent mechanism. From a security perspective, internal adaptation can be an effective approach for handling local adaptations. However, it can also be difficult to scale, and, in addition, adaptation often requires global information about the execution environment and correlation between different external events. On the other hand, external adaptation is scalable, and the adaptation engine is reusable and can be used to integrate adaptation in legacy systems.

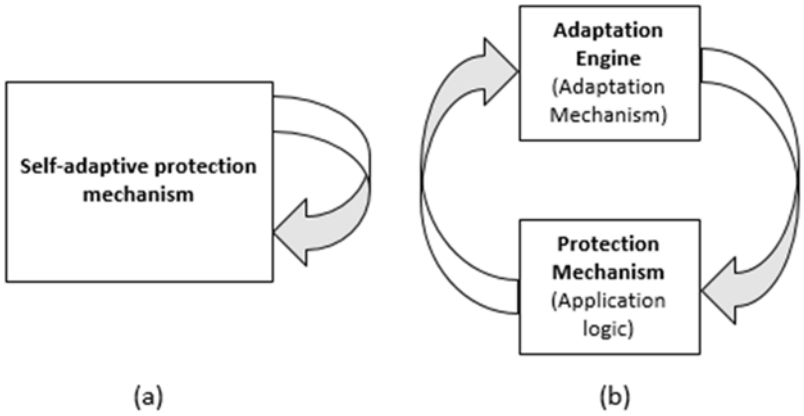


Figure 18 Internal adaptation (a) and external adaptation (b) in the decision making process

From an architectural point of view, both adaptation approaches have their own merits, but more importantly in the search for an adaptive protection mechanism, the two approaches could co-exist and serve to interesting practical uses. Consider a simplified access control system enhanced with an adaptive engine component as shown in Figure 19.

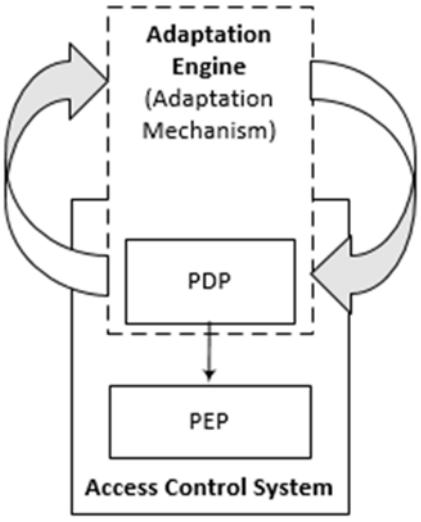


Figure 19 Access Control System enhanced with an Adaptation Engine

This access control system consists of two subcomponents: a policy decision point (PDP) and a policy enforcement point (PEP). The PDP evaluates and issues authorisation decision while the PEP intercepts access requests, forwards them to the PDP for evaluation, and enforces the authorisation decisions issued by the PDP. The interesting aspect is that the addition of the adaptation engine component introduces adaptation in two different ways.

First, the adaptation engine (i.e. external adaptation logic) can incorporate external adaptation and drive the behaviour of the access control mechanism by dynamically transforming abstract policies into executable policies and then adding, modifying, and removing the latter during the PDP's evaluation process.

Second, the adaptation engine can be used to control internal behaviour in the system (i.e. management aspect) by dynamically scoping executable policies (e.g. their applicable context, domains, etc.) that may be defined into the access control mechanism, ultimately influencing the decision that occurs at the PDP. This is a desired feature in scenarios where, for instance, a security policy administrator is authorised to define access control policies targeted to the protection mechanism, i.e. the access control system; but where at the same time such access control policies must comply with global policies (i.e. abstract policies) such as high-level privacy laws, compliance and regulatory policies, and so on, expressed as abstract policies; and in scenarios where, due to local contextual changes, it is more sensible to let the PDP evaluate decisions locally.

Requirements:

To structure the system based on a three-layered architecture that enables:

3. The separation of concerns across the following levels: management, adaptation, and application; and the externalisation of the adaptation logic.
4. The management and control (i.e. management level) of internal behaviour of the application logic (i.e. implementation level) via policy scoping while providing flexibility in the definition of executable policies (i.e. adaptation level).

3.2.3 Modelling the operational environment

The research objective *Obj. 3* (section 1.2.2) addresses the conceptual articulation of the property of specialisation and its implementation via policy transformation. In order to fulfil this requirement, proposing a model of the operational environment has been identified as a fundamental requirement. Policy transformation is based on the concept of enabling the integration of external (i.e. operational environment entities) and internal (i.e. execution context and management policies) constraints. Objective *Obj. 3* addresses the following research questions:

What approach can be taken that allows modelling the execution environment?

Analysis:

Having introduced the concept of an adaptation engine where the main adaptation decision process takes place, another important aspect to consider is what information can be used to support security decisions. Supplementing information should happen at runtime, identifying *relevant actions and contextual factors* in the operational environment. Figure 20 shows a layered IT model and examples of actions that entities from one layer may perform on other

entities from another layer. The dotted line represents the operational environment. Notice that contextual factors not only include variables such as time and location, but also entities' context (e.g. network context including entities such as packets, communication protocols, and so on). How to determine what "relevant actions and contextual factors" means requires understanding and modelling the operational environment including the runtime conditions of execution, actions among the entities involved, as well as relevant and potential security concerns, i.e. security goals, countermeasures, vulnerabilities, threats and the risks associated.

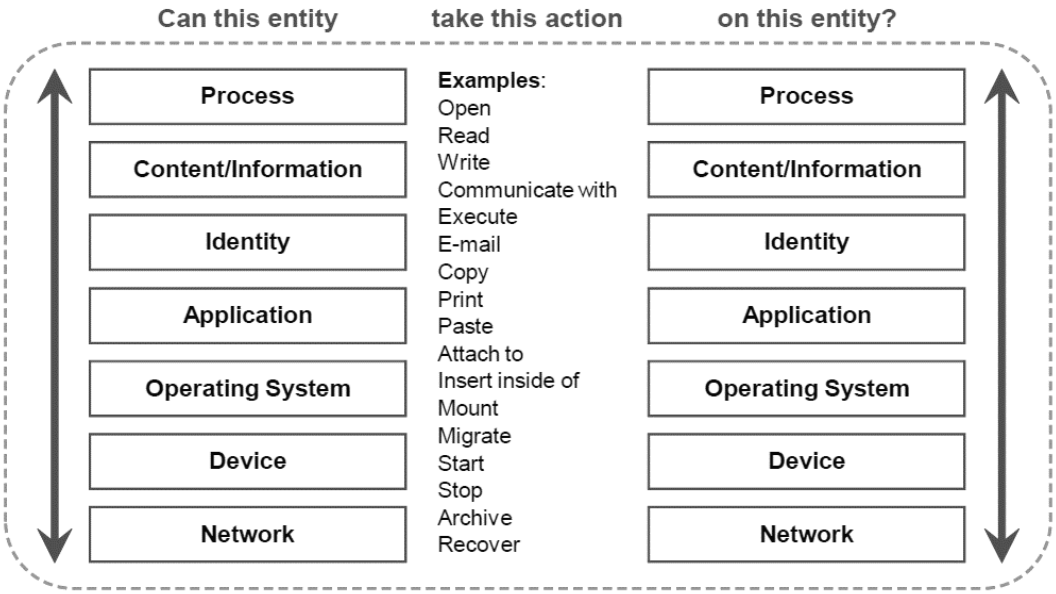


Figure 20 Layered IT model[3]

As mentioned in previous sections, most policy-driven protection mechanisms are based on predefined static policies. Trying to define fine-granular policies to represent each possible state of the execution context is not practical, if not unfeasible, as it would require complete information and dealing with large numbers of possible combinations of contextual variables; and, in addition, protection mechanisms with limited ability of policy expression lack extensibility support when new contextual variables become relevant or known depending on particular situations and over time. Therefore, understanding and reasoning about the operational environment is a complex task that should be dealt with at the adaptation logic level.

Requirement:

5. To provide a model the operational environment in a way that allows reasoning about its entities (including their characteristics) and behaviour.

3.2.4 The property of Specialisation

The research objective *Obj. 3* (section 1.2.2) addresses the conceptual articulation of the property of specialisation and its implementation via policy transformation. In order to fulfil this requirement, the articulation and development of a general policy transformation mechanism is the main requirement of the framework proposed by this thesis. As already mentioned, the policy transformation process is based on enabling the dynamic integration of external and internal constraints taken as input into the adaptation logic in order to produce enhanced adaptation decisions in the form of executable policies. Objective *Obj. 3* addresses the following research question:

How to enable the property of specialisation by means of a policy transformation process at the adaptation level?

Analysis:

In multiple scenarios, there is a need for protection systems able to dynamically support customised security capabilities that reflect and enforce security, privacy, and quality of service (QoS) requirements expressed by diverse external entities, including human users and other software systems, that the protection system is required to interact with.

Ideally (as mentioned before), a policy-driven protection system should be capable of specialisation: i.e. be able to identify and incorporate dynamically changing security requirements and operational constraints. This is of great importance since the constraints introduced by the operational environment shape and scope the structure and behaviour of the protection system.

A system that enables specialisation extends adaptation by providing the system with the mechanisms to allow external constraints, requirements, policies, etc., to be integrated into the adaptation logic.

Requirement:

6. To dynamically capture constraints and requirements in the operational environment and to use them to specialise the adaptation process.

3.2.5 Enhancing adaptation with security models

The incorporation of a policy hierarchy (*Obj. 5*) into a three-layer architecture (*Obj. 4*) that enables the separation of levels of abstraction allows the encapsulation and modularisation of the policy transformation process (*Obj. 3*) at the adaptation level. See the objectives in section

1.2.2. These characteristics of the proposed framework create an opportunity to introduce security-enhancing models as a requirement in the design of the policy transformation process. By allowing reasoning about and the incorporation of security models into the policy transformation process is possible to extend the adaptation logic with security capabilities that otherwise would not be possible at the concrete implementation level due to the different semantics. Objective *Obj. 3* addresses the following research question:

Can the policy transformation process be constructed in a way that allows incorporating security-enhancing models to improve security decisions?

Analysis:

Security is concerned with protecting valuable or vulnerable assets from harm. The usual approach is to identify the value or vulnerability associated to assets, determine *appropriate* levels of security, and define security policies accordingly. However, in dynamic environments, the value, vulnerability, and level of security associated to an asset continually evolves. This makes extremely difficult to predefine an optimal security policy due to the intrinsic unpredictable and uncertain nature of such environments.

An adaptive protection mechanism should be able to incorporate different security models directly at the adaptation level in order to support policy transformation.

For example, consider the concepts of trust and risk. These two concepts rely on unpredictability and uncertainty [65]. Risk is the probability of loss or damage resulting from a given action, inaction, or event; while trust is the willingness of an entity to depend on another with a certain sense of security, although negative consequences are possible. Using models based on risk and trust would provide better support and improve the security decisions made by the adaptation logic.

Requirement:

7. To incorporate and make use of security models to enhance the adaptation logic to dynamically support security decisions.

The above seven requirements address the research objectives of this thesis and form the basis of the research problem to be addressed.

3.3 Concluding Remarks

This chapter presented an overview of the proposed model for a policy-driven adaptive protection system. The model not only takes into account characteristics of successful models

of previous experimental work presented in the literature, but also by incorporating the novelties desired for dynamic adaptation to changing operational contexts and environments by specialising their behaviour in response to the changing security requirements of external entities.

Section 3.2 presented the definition and analysis of the main research questions that constitute the main research problem of the thesis. From the analysis, seven requirements were extracted and they form the basic system requirements of any adaptive protection system to be built following the framework proposed in chapter 7. The stepwise approach proposed in this thesis takes into account all these seven requirements allowing the blueprint framework to fulfil each one.

The following chapters present the proposition of three different policy-based protection mechanisms. All three mechanisms allow dynamic adaptation and specialisation due to changes in external entities constraints. The proposed mechanisms have their own research contributions in their own right, but also they serve as important foundation and experimental work because common core characteristics, methods and components emerge. These common core characteristics will be analysed in detail towards the proposition of the framework for the realisation of policy-driven adaptive protection systems in chapter 7.

Chapter 4

Secure Execution Context Enforcement based on Activity Detection

A mechanism that takes into account the combination of security requirements of independent administrative entities over a set of interacting resources on a smart device requires the ability to provide some sort of execution context control. The proposed framework consists of an architecture and a policy model. The architecture detects different events and activities (i.e. user, system, applications) and based on them enforces applicable policies and constrains the execution context for a given set of resources. The policy model offers a method to dynamically create a secure execution context by combining different types of policies (e.g. access, usage, execution) issued by different entities on protected resources.

4.1 Introduction

Smart devices such as laptops, tablets, smart TVs, game consoles, smartphones, and similar others are increasingly built with more memory, processing power, networking interfaces, better performance, and powerful hardware and software integrated platforms. These capabilities enable rich-functionality applications and services. Currently, smart devices outnumber traditional PCs. This trend will continue to accelerate in the next years and as a result security and privacy threats to individuals and organisations at large are expected to grow substantially [66]. Security incidents caused by known types of cyber-attacks traditionally targeted to PCs, workstations, and mainframes including viruses, malware, impostor updates, phishing, data leakage, and many others, are now being increasingly targeted to smart devices [67] and have the same potential to seriously impact businesses:

service outage, financial loss, data leakage, etc.; as well as individuals: identity theft, privacy issues, leakage of sensitive data such as personal and banking information, etc. However, smart devices present different and sometimes unique privacy and security challenges due to characteristics such as platform openness, form factors, multi user-enabled platforms, and sensor-equipped devices able to collect and process contextual information as in the case of smartphones and tablets, just to give two examples.

In addition, techniques and infrastructure are being developed and deployed which make it increasingly common for such devices to be able to host and use data, (smart) applications, and services often owned or controlled by different entities (e.g. corporate vs. personal vs. third parties) with their own security requirements and under different security contexts. This kind of scenario raises several questions such as how to protect hosted resources against malicious entities (e.g. code downloads), how to satisfy and harmonise security and privacy requirements of different independent entities, who defines what security features must be in place, what contextual conditions must exists, what security policies must hold true that guarantee a secure execution context for a piece of data to be accessed by a given application, or for an application or service to be installed or used, and who is to enforce security policies.

Traditionally, security models for smart devices often assume the existence of a single security administration entity that defines security policies and execution constraints on how users, data, and applications can be processed on a device, be it the network carrier, or a corporate administrator owner of such device, or more often the user itself. The problem with this security assumption is that privacy and security requirements of entities, other than the single security administrator, are ignored by default or simply cannot be defined and enforced due to the lack of mechanisms to do so; making difficult the use of hosted resources in an execution environment that is too restrictive while hindering the flexibility and openness of today's smart device platforms.

Moreover, a mechanism that takes into account the combination of independent administrative entities' security requirements over a set of interacting resources on a smart device requires the ability to provide some sort of secure execution context control. Some authors have proposed mechanisms based on the idea of enforcing predefined *security profiles* under different contexts (e.g. home vs. work) as a way to provide flexible context isolation [68, 69]. It is acknowledged that these approaches are in the correct directions. However, a scenario where independent administrative entities' security requirements need to be enforced implies that not only these requirements must be accounted for when determining what is considered as a secure execution context, but also that such execution context has to be defined dynamically at runtime when the involved entities become known and interact.

Here, a framework for the enforcement of such execution context, for resources hosted on smart devices, defined dynamically by multiple administrative entities and expressed via policy is proposed. The proposed framework consists of an architecture and a policy model. The architecture detects different events and activities (i.e. user, system, applications, services, etc.) and based on them enforces and monitors the execution context. The policy model provides a method to dynamically create a secure execution context by combining different types of policies (e.g. access, usage, execution) issued by different entities on their protected resources.

This chapter is organised as follows. Section 4.2 presents the related work. Section 4.3 further elaborates on the scenario presented in the introduction and other related scenarios in order to define the shortcomings found in such scenarios and to elicit the requirements of the proposed solution. Section 4.4 provides an overview of the framework. Section 4.5 introduces the proposed policy model. In section 4.6, the proposed system architecture is presented. Section 4.7 presents a use case scenario including a design and implementation based on the proposed framework. Finally, Section 4.8 presents the conclusions.

4.2 Related Work

Operating systems such as Android [1] and iOS [70] use the concept of application privilege separation [71]. In the Android OS applications run in separate processes in order to provide isolation from each other. An inter-process communication (IPC) mechanism allows sharing data between applications but operations are constrained by permissions, thus providing Mandatory Access Control (MAC). At install time, applications statically request the permissions they require and the user is prompted to decide whether to grant them all or none; the user cannot select a subset. Permissions are coarse-grained and enforced during the whole lifecycle of the application. The Android OS enables applications to declare custom permissions that isolate application-level features. Other applications may request these permissions but whether they are granted is decided solely by the user, not by the application that declared the permission itself. Furthermore, the granting decision is not augmented based on contextual factors.

A considerable amount of research has focused on making security rules, of the kind of Android permissions, more flexible and selectable to the user, and also on adding mechanisms to define and enforce such security rules based on contextual information [68, 69, 72].

In [69] a policy-based framework for enforcing isolation of software applications and data on the Android platform is proposed. It consists of predefined *security profiles* each one associated

with a set of policies that control the access to applications and data, and the dynamic switching between security profiles. In this thesis, the framework proposed introduces the concept of "active execution contexts", similar to that of security profiles as a way to impose execution constraints. However, active execution contexts are created dynamically (i.e. not predefined) at runtime based on the combination of policies declared by the host system, hosted applications, and data files interacting.

In [73] the authors identify the lack or limitation of mechanisms for applications to protect themselves. A framework is described where: applications can provide installation-time policies, inter-application interactions are governed by runtime policies asserted between caller and callee applications, and (optionally) application and system policies can be overridden by the end user. Similar to the framework proposed in this chapter, their approach allows for multiple entities (i.e. applications) to define their own policies. However, it does not consider the situation of policy-protected data files by different entities; and their proposed use of overrides is only given to the end user at system level. The framework proposed in this chapter uses the concept of overrides as a mechanism for authority and trust delegation from any entity to another, and at different levels of granularity (data files, applications, and host system).

In [74] a mandatory access control (MAC) system for smart devices that uses input from multiple stakeholders to compose policies based on runtime information is presented. This work falls into the area of distributed access control and it is related to the framework proposed here as it provides a mechanism for combining independent stakeholders' policies; however, the resulting policy is determined targeting a given resource, i.e. application. In essence, this is a mechanism applicable to distributed access control which is different to the problem controlling the execution context for a set of entities expressing different requirements on different objects and for different reasons.

4.3 Scenario

In this section a simple corporate scenario is analysed, also some additional but relevant use case scenarios are considered, and then the proposed solution requirements are defined.

Consider a scenario where an employee accesses corporate data using a smart application provided by a third party and hosted on a mobile device administered by an IT Department whilst the device is connected only to an unsecure external network such as the Internet from home; or a third party application being downloaded on a corporate device and used by an employee in a corporate environment.

As mentioned in the introduction, typically, security models for smart devices assume a single security administration entity to define global coarse-grained security policies. Moreover, some

organisations prohibit the option to install third party applications on devices. At first glance, this policy may serve to its security purpose but it also limits enterprise collaboration, networking, and productivity [75]. To overcome this limitation, it is common that employees opt for using a second device for work-related activities, which in turn increases security vulnerabilities and breaks the corporate security policy.

Moreover, an application provider may declare what system resources (of the device) the application intends to use. One way to achieve this is using a permissions model like the one of the Android operating system in which, as mentioned before, during installation the system asks the user to accept a set of permissions defined by the application and the user is required either to accept all of them or to decline from installing the application. This solution has several shortcomings: it makes security inflexible; once the user authorises certain permissions they hold until the application is removed; and users tend to forget over time what they consented to. In addition, the application provider may define a security policy restricting the use of the application, but the enforcement decision is user-centric, it does not take into account contextual information, and does not provide fine-grained application-level enforcement.

Finally, a single-administrator security model does not integrate well in smart devices where external data and application providers may want to define policies to be enforce on the device environment as in the following scenarios: Digital Rights Management (DRM) where copyrighted digital content is protected by licensing policies and requires certain restrictions on the accessing applications, user-generated content (UGC) where end-users want to protect or restrict access to content, and parental control where home devices such as PCs, laptops, smart TVs and game consoles host data owned and used by applications and users with different access and usage rights. In such scenarios it is not clear how data and application policies can be integrated with the device's global policies. For instance, in DRM the data provider would require to define usage policies on the device environment and on the applications that intend to access protected content.

There is therefore a need to provide security measures for use with such devices to ensure that any hosted data, applications, and services are accessed and used in secure manner. Data must be handled securely, especially to prevent devices from having confidential data accessed by unauthorised third parties either maliciously or inadvertently. Such unauthorised access could result from malware operating on the device in question, or from user actions performed on the device either as a result of some sort of malware (e.g. some sort of social engineering attack) or simply by the user using a device at wrong location or performing an unwise action.

4.3.1 System solution requirements

Based on the abovementioned shortcomings the following system requirements are considered. The system solution should:

1. Be flexible enough to allow different entities with different security and privacy requirements to express their own security policies; and be able to enforce them.
2. Allow expressing fine-grained policies associated to the device's resources as well as to hosted resources: data, applications, and services.
3. Augment policy definition based on monitoring activity on the host system, its state and contextual information.
4. Be able to integrate and combine entities' policies in a way that harmonises their independent requirements by enforcing a secure execution context.
5. Allow users, data, applications, and services to operate and interact under flexible conflict-free security contexts.

4.4 Proposed solution

In this section, the proposed conceptual model (see Figure 21) is presented. A smart device consists of hardware components, an operating system (OS), and hosted external resources. The OS provides libraries and interfaces to implement core services and to allow interaction and operation of external hosted applications and services, and of system resources: software and hardware. Software system resources include the file system, processes, network sockets, etc. Hardware resources are managed via a kernel. The kernel provides the hardware abstraction layer for the management of memory, processing power, network stack, security, etc.

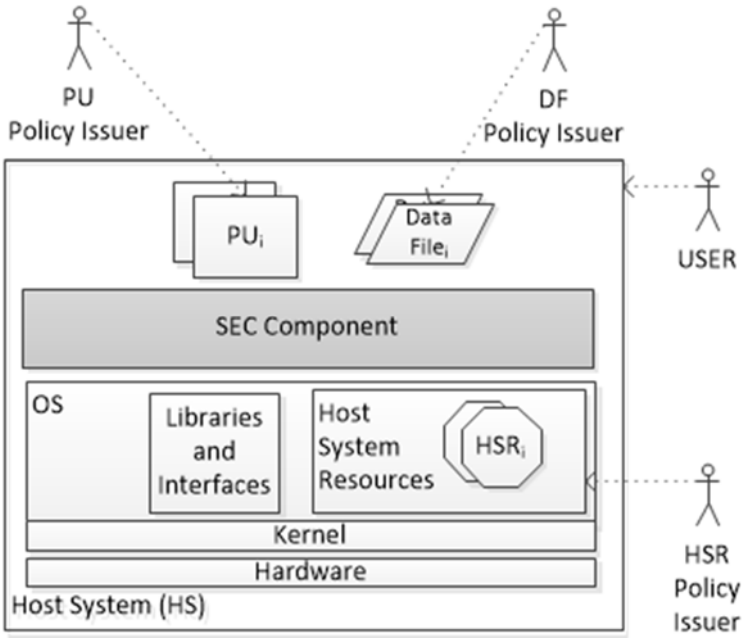


Figure 21 Conceptual Model

In the proposed framework, a smart device is referred as the "host system (HS)", in order to clearly emphasise the fact that a smart device effectively acts as a host of both data and processing entities such as application and services. Three types of protected (by policy) resources are defined:

1. Data File (DF): a container entity that stores data to be used by processing units or the host system. It refers to information seeing as input to or output from a processing unit.

2. Processing Unit (PU): any software entity with behaviour, e.g. (fore- and background) (smart) applications, services, processes, etc., able to consume or interact with any of the following: host system resources (HSR), DFs, and other PUs.

3. Host System Resources (HSR): any entity that falls into the definitions of DF and PU but that strictly belongs to or is part of the HS itself. For instance, OS data files, OS processes, sensors, network connections and interfaces, etc. Here, this type of resources are referred as HSR, or separately as DF_{HSR} and PU_{HSR},

As shown in Figure 21 each protected resource on the HS is protected by a policy. This policy is defined by a policy issuer entity and is associated to each resource instance of different the types: DF, PU, and HSR. Notice that the device user can be a policy issuer himself. Policy issuers ($POL_{ISSUERs}$) have the authority to declare security and privacy requirements on their own resources, and, in addition, those resources interact by performing actions among them, and as consequence an execution context is created on the HS environment. Therefore, a secure execution context is one that is generated by real-time interactions between policy-protected

resources and the user, and that is define by their policies. Then, it is necessary to integrate, combine, and harmonise the different policies that ultimately will define a secure execution context to be enforced. This is the primary requirement of the proposed solution and is achieved by means of the proposed policy model and the Security Execution Context (SEC) architectural component of Figure 21. In the subsequent sections the policy model and SEC architecture are described in detail.

4.5 Policy model

As explained in the previous section, protected resources have associated $POL_{ISSUERS}$. There are three different types of POL_{ISSUER} entities that correspond to DF, PU, and HSR, respectively. The concept of a *policy issuer* is used in the design of the proposed system in order to describe how multiple independent entities may control and define different policies with different semantics. By different semantics is meant a set of policies expressing different requirements that not always can be logically matched, but that must be integrated and combined without violating any policy of the set. Consider policies semantically different in the illustrated corporate scenario: for example, the IT Department (HS-POL_{ISSUER}) may define policies on the host system such as compliance policies, execution policies restricting when and how applications may run, usage policies defining what system resources (e.g. an interface) an application can use, security policies defining what ports are available, data policies defining under what circumstances a type of data can be accessed by a type of application. The third party provider of the smart application (PU-POL_{ISSUER}) may define his own policies about access to data (e.g. allow access to contacts list), usage of resources (allow usage of GPS module), execution (e.g. the application can be launched only if Wi-Fi is enabled), and security (e.g. anti-virus software must be running). Finally, the corporate email data owner ((DF-POL_{ISSUER})) may define additional policies: access policies (e.g. allow access if user role is employee and the application is company owned or from a trusted third party), security policies (e.g. VPN must be enabled), etc. Although semantically different policies may exist they can be expressed using similar policy syntax, and, in addition, since resources of type HSR consist of DF_{HSR} and PU_{HSR}, the latter share the same structure as DF and PU policies.

The first step in the design of the proposed policy model is to identify the objects to be protected, the subjects that perform actions on those objects, and the policy issuing entities of objects and subjects.

There are two types of subject entities that execute or trigger actions on objects on the host system: user (U) and processing unit (PU). U can be any person e.g. employee, device

administrator, etc. And there are three types of object entities that are policy protected or restricted: DF, PU, and HSR. Notice that a processing unit can be both subject and object at a given time. For instance, during an interaction a processing unit PU_1 trying to access a data file DF_1 acts as the subject but if the same processing unit PU_1 is being executed by another one (e.g. PU_2) the former becomes the object.

The following two types of policies are considered: POL_{DF} and POL_{PU}, associated to data files and processing units, respectively. Before formalising the definition of policies, first the basic structure of policy rules and the concept of overrides are introduced and defined.

4.5.1 Policy Rules

DEFINITION 1 (PR). Policy rules PR of the form *(subj, act, obj, cond, eff)* are defined to express that subject *subj* performs action *act* on object *obj* under (optional) conditions *cond* with effect eff (i.e. permit or deny).

For instance, consider the rule *(alice, read, data_file1, {purpose=work_activity, anti-malware=ON}, allow)* where this is the policy of data_file1 (or data_file1 is the target of the policy). This rule states that user Alice can read data_file1 if the anti-malware is enabled and for work-related activities. It was opted for an attribute-based policy model to enable flexibility and fine-granularity in policy expression.

4.5.2 Policy Overrides

Additionally, in order to provide flexibility of policy combination policy rules were extended with the concept of overrides.

DEFINITION 2 (Override-Rule). An override is defined as a "policy evaluation decision delegation given by a policy issuer to another and granted to a specific PU for a specific resource". An override rule (Override-Rule) is of the form (POL_{ISSUER}, Override-type, PU-target) meaning POL_{ISSUER} is granted an override where Override-type can be permit-override, deny-override, or all-override, and where PU-target defines a set of required conditional attributes that characterise and categorise the PU targeted.

Consider the rule *(any-PU, write, data-file1, –, deny)* extended with the override rule *(POL_{ISSUER}=company_xyz, permit-override=true, PU_{TARGET}=email_app))*. This policy states that no PU can write to data_file1 (i.e. the specific resource); however, any PU of type email_app and with policy by POL_{ISSUER} company_xyz can have an associated policy

applicable to data_file1 that overrides the effect and may evaluate to *permit*. This exemplifies policy combination using overrides.

Notice that overrides are granted (implicitly via $POL_{ISSUERs}$) by DFs and/or PUs to PUs only. This is simply because DFs do not have behaviour (DFs are always objects) so it would not make sense to define a DF as the receiving target of a granted override. Also, notice that here the concept of overrides is used in a totally different way to other policy models such XACML [76]. XACML uses overrides as part of their policy and rule combination algorithms as a means to arrive to an authorisation decision given a set of policy rules with different effects but controlled by a single administrative entity. Instead, here overrides are used as a means of authority (and trust) delegation to another issuing entity possibly belonging to a different administrative domain.

4.5.3 Data File policies (POL_{DF})

DEFINITION 3 (POL_{DF}). A policy POL_{DF} defines the conditions under which a data file DF can be accessed or used by users U and PUs. A POL_{DF} is a tuple (POL_{ISSUER}, PR, Override-Rule) where POL_{ISSUER} is the issuing entity of the policy, PR is a policy rule as per definition 1 and Override-Rule is a rule as per definition 2.

The object obj of a POL_{DF} rule is always the DF that the policy applies to and on which actions *act* are performed by subject *subj*. Conditions *cond* determine different semantics for the rule, however, the evaluation process remains unchanged. As an example consider the previous one in the overrides subsection.

4.5.4 Processing Unit Policies (POL_{PU})

POL_{PU} are policies associated to processing units. A policy POL_{PU} consists of two subtypes of policies: $POL_{PU\text{-}AS\text{-}OBJ}$ and $POL_{PU\text{-}ON\text{-}OBJ}$. $POL_{PU\text{-}AS\text{-}OBJ}$ refers to policies where PU is the object in the policy on which actions can or cannot be performed. $POL_{PU\text{-}ON\text{-}OBJ}$ refers to policies where PU is the subject in the policy and which can or cannot perform an action on another given resource(s).

DEFINITION 4 ($POL_{PU\text{-}AS\text{-}OBJ}$). A policy $POL_{PU\text{-}AS\text{-}OBJ}$ defines the conditions under which the processing unit PU can be accessed or used by other PUs or users U. A $POL_{PU\text{-}AS\text{-}OBJ}$ is a tuple (POL_{ISSUER}, PR, Override-Rule) where POL_{ISSUER} is the issuing entity of the policy, PR is a policy rule as per definition 1 and Override-Rule is a rule as per definition 2.

Policies $POL_{PU\text{-}AS\text{-}OBJ}$ are exactly equal in structure to POL_{DF} and can be distinguished because the object obj of the rule PR is the processing unit PU itself: the PU that the policy applies to. For example, consider the following policy defined for the PU Camera-App: (HSR-POL_{ISSUER}, (Picture-Editor, launch, Camera-App, -, permit), -). This is a policy issued by the HSR-POL_{ISSUER} that allows application Picture-Editor to launch Camera-App (i.e. the object in the policy). Note: A dash in a policy declaration indicates the absence of optional elements (e.g. no conditions and overrides defined).

DEFINITION 5 ($POL_{PU\text{-}ON\text{-}OBJ}$). A policy $POL_{PU\text{-}ON\text{-}OBJ}$ defines the conditions under which a processing unit PU_i can access or use other hosted resource(s), either data files DF or processing units PU_j. A policy $POL_{PU\text{-}ON\text{-}OBJ}$ is a tuple (POL_{ISSUER}, PR) where POL_{ISSUER} is the issuing entity of the policy and PR is a policy rule as per definition 1.

Policies $POL_{PU\text{-}ON\text{-}OBJ}$ do not define overrides because the object is not the PU the policy applies to but another specified resource as in the following policy defined for the PU Camera-App: (HSR-POL_{ISSUER}, (Camera-app, read, Photo-Gallery, {loc=home}, permit), -). In this example, Camera-app is the subject and the policy states Camera-app can access Photo-Gallery if the location is "home". Notice that since $POL_{PU\text{-}ON\text{-}OBJ}$ always has another resource as object (Photo-Gallery), $POL_{PU\text{-}ON\text{-}OBJ}$ may serve as overriding policies to the specified resource object. For instance, the policy in the previous example could be an overriding policy to another policy defined for DF Photo-Gallery granting an override for Camera-App.

4.5.5 Policy combination and evaluation behaviour

In this section, the proposed policy combination model and the policy evaluation behaviour of the system are presented. One of the main functions of the Secure Execution Context (SEC) component (see Figure 21) is to evaluate policies defined by different entities that become applicable. By applicable policy is meant a policy that applies to a particular resource (DF or PU) during a particular interaction such as a PU or a user U accessing a DF or using another PU, or consuming host system resources HSR under a given context. For example, a DF may contain two policies: 1) Alice can access data_file_1 at home, and 2) Alice cannot access data_file_1 at her office. If Alice were at home, then only policy (1) would be evaluated. Policy (2) would not be applicable in this context (home) but only if Alice were at her office. Applicable policies are evaluated depending on Us, DFs, and PUs involved in a given context.

The combination of different policies is achieved by means of evaluation rules that produce different evaluation behaviours depending on the interaction taking place and the applicable policies that are available for evaluation. Table 1 summarises the evaluation decision process.

Table 1 Policy combination rules

Rule ID	POL_{DF}/ $POL_{PU\text{-}AS\text{-}OBJ}$		POL_{PU}		Baseline Evaluation Rules
	POL_{XX}	Override Rule	$POL_{PU\text{-}AS\text{-}OBJ}$	$POL_{PU\text{-}ON\text{-}OBJ}$	"eval()": evaluation function
1					ALLOW
2			✓		"eval($POL_{PU\text{-}AS\text{-}OBJ}$)"
3				✓	"eval($POL_{PU\text{-}ON\text{-}OBJ}$)"
4			✓	✓	"eval($POL_{PU\text{-}AS\text{-}OBJ}$) EXCEPT eval($POL_{PU\text{-}ON\text{-}OBJ}$)"
5	✓				"eval(POL_{XX})"
6	✓		✓		"eval(POL_{XX}) AND eval($POL_{PU\text{-}AS\text{-}OBJ}$)"
7	✓			✓	"eval(POL_{XX}) AND eval($POL_{PU\text{-}ON\text{-}OBJ}$)"
8	✓		✓	✓	"[eval($POL_{PU\text{-}AS\text{-}OBJ}$) EXCEPT eval($POL_{PU\text{-}ON\text{-}OBJ}$)] AND [eval($POL_{XX}$)]"
9	✓	✓		✓	"[eval(POL_{XX})] OVERRIDE [eval($POL_{PU\text{-}ON\text{-}OBJ}$)]"
10	✓	✓	✓	✓	"[eval(POL_{XX})] OVERRIDE [eval($POL_{PU\text{-}AS\text{-}OBJ}$) EXCEPT eval($POL_{PU\text{-}ON\text{-}OBJ}$)]"

A tick (✓) indicates what type of policy is found for evaluation during an interaction and (in the case of POL_{DF} and $POL_{PU\text{-}AS\text{-}OBJ}$) whether it defines an override or not. Table 1 shows the policy evaluation rules between two entities only but it can be easily extended to any number of interacting entities, and the 10 evaluation rules are the baseline of the proposed model and suffice to deduce and derive other interaction cases not shown on the table (due to space constraints). Since POL_{DF} and $POL_{PU\text{-}AS\text{-}OBJ}$ have exactly the same syntactic structure and are semantically the same in essence (i.e. DF and PU are the object in the policy) they are represented together on columns 2 and 3, and are labelled as POL_{XX} whenever they appear in the evaluation rules meaning the rule applies equally to both cases.

The evaluation rules use the following evaluation function and operators:

- "eval()": it takes as parameter the policy to be evaluated. Returns PERMIT or DENY
- Operator "AND": it is a standard logical AND operator
- Operator EXCEPT: it is used in the form eval("pol1") EXCEPT eval("pol2") and it gives priority to the evaluation decision eval("pol2") over eval("pol1") if and only if policy pol2 is an exception to policy pol1.
- Operator OVERRIDE: it is used in the form eval("pol2") OVERRIDE eval("pol1") and it overrides the evaluation decision eval("pol1") with eval("pol2") if and only if policy pol1 grants override privilege to policy pol2.

4.5.6 Expanded evaluation behaviour

In the case when a processing unit PU_1 initiates an interaction to consume a second processing unit PU_2 that, in turn, attempts to consume another processing unit PU_i or a data file DF_i, and so on, regardless of whether PU_2 by itself attempts to consume the next resource or as a result of PU_1 accessing such resource indirectly through PU_2, the evaluation process includes all the interacting entities' policies: PU_1, PU_2, PU_i, DF_i, etc. All policies are combined as per the evaluation rules of Table 1. The evaluation process is performed incrementally revaluating for each policy added until the result of an evaluation results in "deny" or until the chain of evaluations completes and a final decision is made.

4.5.7 Policy Integration and Host System Permissions

Policy integration refers to the concept of global rules that determine how POL_{DF} and POL_{PU} expressed in an abstract way can be integrated semantically at system level. Consider the policy POL_{DF1}: (email-app, write, contacts-list, {WiFi=off}, allow); and the policy POL_{PU1}: (user=Alice, execute, PU1, {loc=home}, allow). POL_{DF1} says that any PU of category "email-app" can write to "contacts-list" if the WiFi interface is switched off. POL_{PU1} says that user Alice can execute PU1 if the location is home. The concept of categories (or labels) is introduced to characterise policy elements when defining policies as in the case of POL_{DF1}, where any PU of category "email-app" is referred to and not necessarily to a specified PU. Therefore, if both POL_{PU1} and POL_{DF1} were to evaluate to true and if PU1 were of category "email-app" then it could write to "contacts-list". In fact, in the proposed solution, hosted resources PUs and DFs declare their own categories (or labels) and the host system HS decides whether to accept such categories as valid. In other words, the concept of categories is used to define "permissions" at HS level. If PU1 is accepted as an "email-app" then it can access the contacts-list.

DEFINITION 6 An $HS_{PERMISSION}$ is a set of accepted categories declared by a PU or a DF that determine a set of permissions for that PU or DF at host system level. An $HS_{PERMISSION}$ is of the form (RES_{HOSTED}, {cat1, cat2 …}), where RES_{HOSTED} is a hosted resource (i.e. PU or DF) and {cat1, cat2 …} is a set of categories associated to that resource.

4.6 Secure Execution Context Enforcement Architecture

The core objective of the architecture is to enable the ability to define and enforce Secure Execution Contexts for a multiplicity of entities (DF and PU) with different intends and actions

where inter-relations may require to be constrained at different levels. For example, if PU_i is accessing DF_i and PU_{i+1} is then launched, it is possible this will have an impact on the execution context already established between PU_i and DF_i. The purpose is to satisfy execution conditions for given contexts (defined as/by the combined policies of the entities interacting), manage possible inter-context conflicts, and provide execution context isolation.

Figure 22 depicts the system architecture. The main modules are: the Events Handler Module (EHM), the Policy Enforcement Module (PEM), the Policy Manager Module (PMM), and the Security Context Monitor (SCM). The EHM handles different types of messages generated by different components of the architecture and mediates communication between them. The PEM performs activity detection and policy enforcement functions. The PMM resolves, integrates, combines, and evaluates applicable policies. When the result of policy combination evaluates to "true" as per Table 1 the PMM allows an execution context. The SCM monitors such execution context and notifies when the conditions change triggering policy re-evaluation (at the PMM). Each module will be described in detail in the rest of this section.

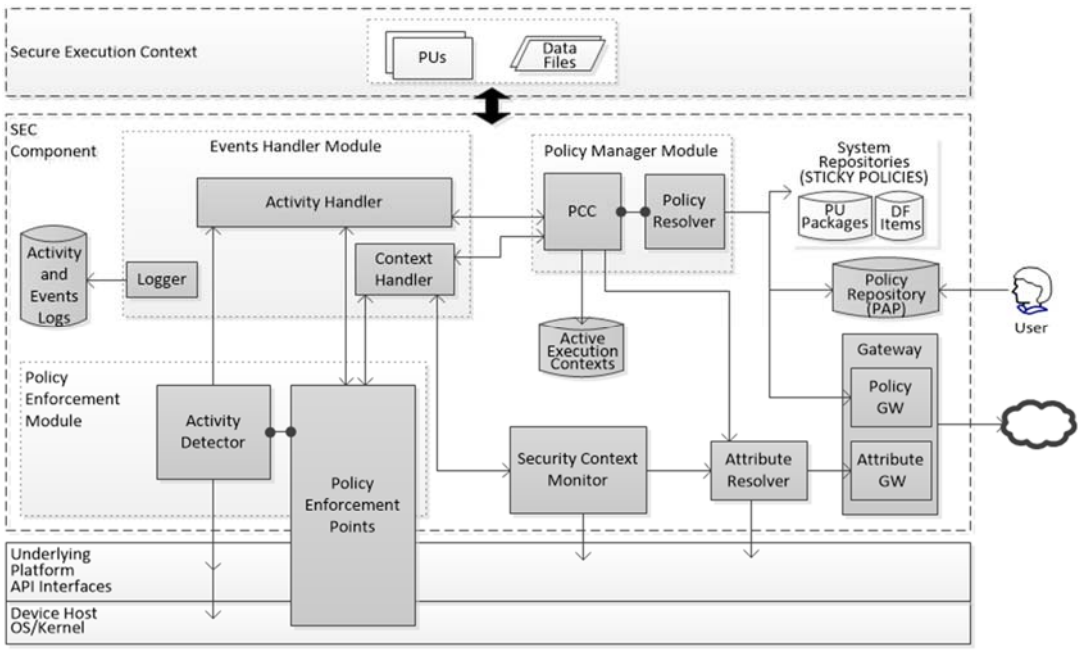


Figure 22 Architecture

The Policy Enforcement Module (PEM) consists of an Activity Detector component and one or more Policy Enforcement Points (PEPs). The Activity Detector detects real-time actions performed on (hosted and host system) resources, i.e. DF, PU, DF_{HSR} and PU_{HSR}. Activities can be triggered by users (e.g. launching an application), PU_{HSR} (e.g. killing a background service), and hosted PUs (e.g. accessing a DF). When an activity is detected, the Activity Detector checks who is triggering the activity and does the following: if triggered by the SEC component (i.e. our architecture itself) it is considered safe and ignored; if triggered by the host system (i.e.

by a PU_{HSR}) the Activity Detector passes control to the Events Handler Module (EHM) by sending an activity message; and finally, if triggered by the user or a hosted PU the Activity Detector signals the appropriate PEP(s) to momentarily halt the PU's execution and sends a corresponding activity message to the EHM. In the latter case the PEM goes to a pending state waiting for a return call from the EHM indicating how the PEP(s) should proceed. For a given activity detected, the actionable PEP(s) depend on the type of actions attempted by the triggering entity, e.g. blocking a graphical user interface or pausing, stopping, or even killing a background service, etc. As shown in Figure 22, PEPs can be integrated at the kernel layer and/or at platform API interfaces layer (depending on implementation requirements).

The Events Handler Module (EHM) consists of three subcomponents: Activity Handler, Context Handler, and Logger. The Activity Handler processes activity messages coming from the Activity Detector. Activity messages are of the form (subject, action, and object) where "subject" is the entity that triggers and executes an "action" on the target "object". Depending on the object type (PU or DF) adequate actions may be defined, e.g. for DF: create, write, update, delete, etc.; for PU: access, use, start, stop, pause, kill, update, install, uninstall, etc. Upon receiving an activity message the Activity Handler forwards it as an evaluation request to the Policy Manager Module (PMM). The Context Handler processes context messages coming from the SCM component generated when the conditions of an active execution context change, and signals PEPs and the PMM. The Logger keeps records of all events generated by the Activity Detector and the SCM. It is used to resolve queries about previous activity behaviour and state such as how many times an application has launched. It is also used for auditing purposes.

The Policy Manager Module (PMM) consists of a Policy Combination Component (PCC) and a Policy Resolver. The PMM receives evaluation requests from the Activity Handler when a new activity is detected on the system. Upon receiving an evaluation request, the PCC first contacts the Policy Resolver to retrieve the policies associated to the entities involved in the activity detected. The Policy Resolver fetches policies locally either from the Policy Repository or as sticky policies attached to DF items and PU packages from the System Repositories; or remotely (through the policy gateway) resolves from $POL_{ISSUERS}$ servers if policies are URI-referenced. If associated policies are found, the PCC determines all applicable policies, and requests from the Attribute Resolver component all attribute values required to initiate the policy evaluation process. The Attribute Resolver implements attribute resolution routines that make calls to the available Underlying Platform API Interfaces on the host system and resolves for contextual, system state, and resource attribute values, both locally and remotely (through the attribute gateway). During policy evaluation applicable policies are integrated and combined as describe in section 4.5. If the evaluation result is "deny" the PCC sends a deny

response to the appropriate PEPs via the Activity Handler. The Activity Handler maintains session information between the PCC and PEPs for each activity detection event. If the evaluation result is “permit” then the PCC generates an “execution context” defined by the resulting combined policies (see evaluation rules of Table 1). However, it is possible that this new “execution context” could conflict with other(s) currently “active execution contexts”. For instance, there could be an active execution context enforcing that the Wi-Fi interface must be turned off when the new execution context requires to enforce the opposite. In such case the PCC needs to resolve such conflict. After evaluating to “permit” the PCC checks for conflicts in the Active Execution Contexts Repository where “active execution contexts” are stored. Conflict resolution is achieved by re-evaluating the stored policy combinations under the assumption of the conditions of the new combined policy generated. If no conflict is found the PCC adds the new execution context to the repository and signals the PEPs to enforce it. Otherwise the new execution context is not authorised, or a resolution strategy is followed. For example, if two conflicting execution contexts were generated as a result of actions performed by the same user then the PCC notifies the user and asks him what execution context to discard. A more advanced strategy can be that the PCC makes a decision based on predefined priority rules.

In addition to receiving activity detection-related evaluation requests from the Activity Detector (mediated via the Activity Handler), the PMM also receives evaluation requests (via the Context Handler) triggered by the Security Context Monitor (SCM) when the allowed conditions of currently active execution contexts change. As mentioned before, conflict-free security execution contexts become active and must be monitored. That is the purpose of the SCM. The SCM monitors events that are specific to each active execution context and that are not originated by direct intended actions of users, PUs, or the host system on targeted resources (called activities), but instead, it monitors specific per-execution context changes of conditions that have an impact on any currently active execution context. Conditions may include, for example location, time and date, elapsed time, active network interfaces and connections, protocols, etc. This is done as follows: once a conflict-free execution context is authorised by the PCC and is stored in the Active Execution Contexts Repository, the PCC notifies the Context Handler and passes to it the new active execution context (i.e. the new combined policy). The Context Handler processes the combined policy and determines the required conditions that must hold true for the execution context to remain active, i.e. in a valid state; and registers these conditions with the SCM for their monitoring. The SCM consists of predefined monitoring modules that communicate with the Attribute Resolver and Underlying Platform APIs components in order to retrieve information about attributes, state, and context; and determine if a registered condition becomes valid or invalid for the given execution context.

When a registered execution condition changes to invalid state, the SCM fires a corresponding event and notifies the Context Handler. The Context Handler signals the PEM and forces the PMM to re-evaluate the corresponding execution context.

4.7 Controlling Access and Usage of Files and Applications: Use Case Scenario

This section presents the architecture, design, and implementation of the application "User behaviour Locker" based on the framework described in this chapter. User behaviour App Locker is a security application that restricts access to installed applications and data files residing within a mobile device based on contextual information.

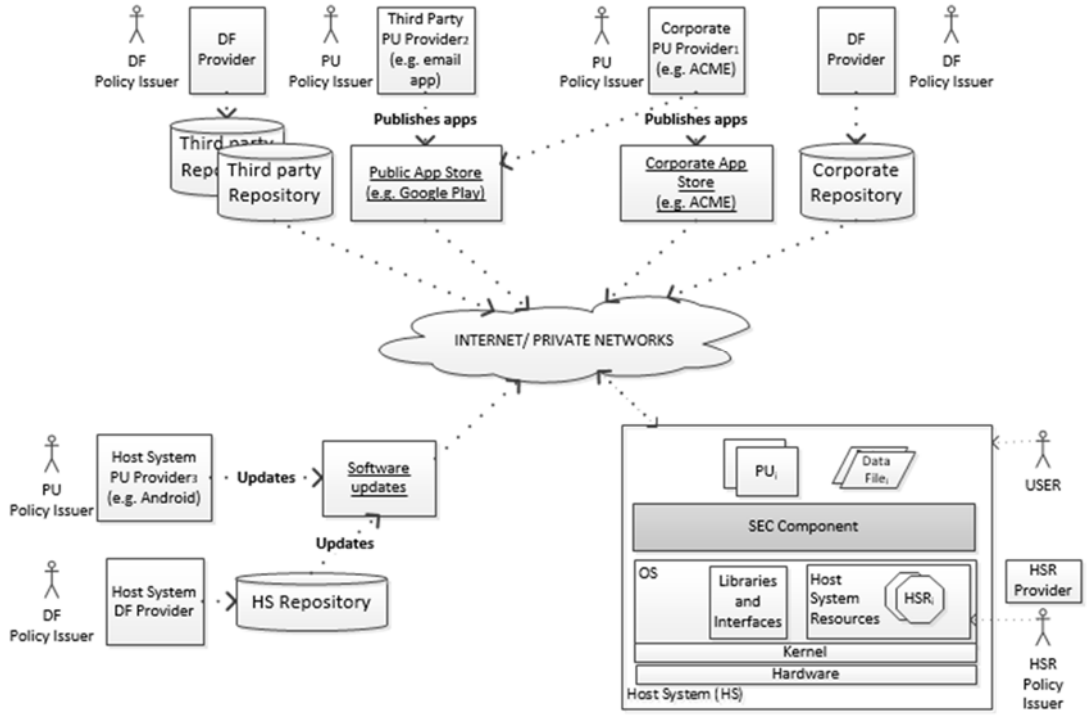


Figure 23 User behaviour App Locker Use Case Scenario

Consider the following scenario. Alice is given a mobile device by her employer, company ACME. ACME wants Alice to be able to use the same device in two domains: professional and personal, while adhering to the company security policies. For example, Alice should be able to access a data file at her work location but not at home. As shown in Figure 23, the mobile device is enabled to connect via Internet to third party application stores, such as Google Play, and download data files and Apps (i.e. Processing Units – PUs). Similarly, ACME provides a corporate Apps store to enable employees downloading Apps and data files for business reasons. The mobile device can also download host system resources and updates

(e.g. Android OS). Moreover, data files and Apps providers (ACME, Third parties, and Host System) define access and usage policies on their resources. The User behaviour App Locker application implements the SEC (Security Execution Context) Component shown in Figure 23 and provides the required security functionality, policy-based access and usage control and separation of domains.

4.7.1 User behaviour App Locker Policies

In User behaviour App Locker, access restrictions are represented as security policies. These security policies are "sticky" meaning they are physically or logically attached to the application or data file that they protect.

4.7.1.1 Protected Resources and Security Sticky Policy

Protected applications and protected data files are resources which are protected by a security sticky policy.

4.7.1.1.1 Protected applications policies

When a protected application is installed in the mobile device, its installation package contains an associated security (sticky) policy file. This policy file, named *app_policy.xml*, defines security and contextual constraints specifying the conditions under which the application can be launched/used/accessed and by whom (see Figure 24 – left). Figure 24 (right) shows an example of a protected application policy.

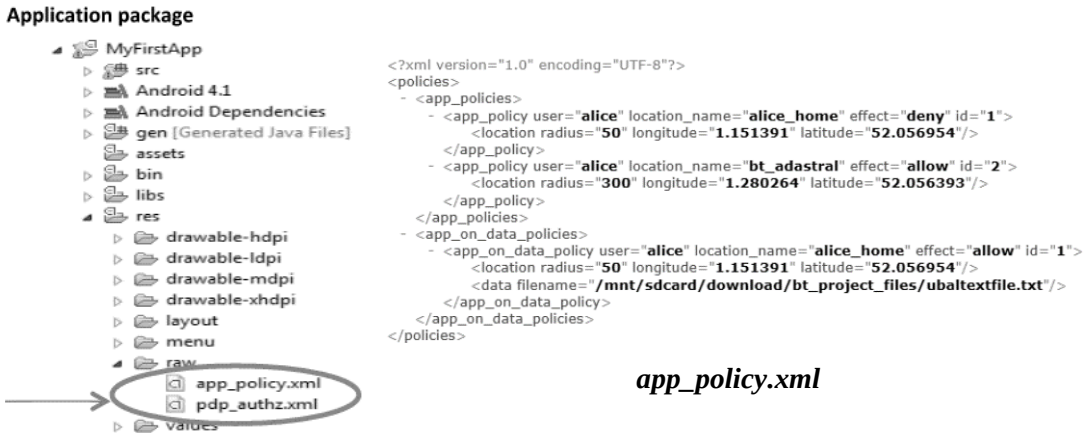


Figure 24 Protected application policy

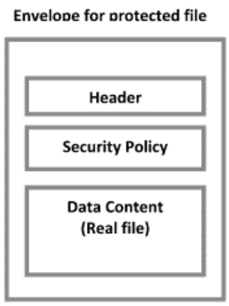
4.7.1.1.2 Protected data files policies

Protected data files are files with file extension *"ubal"*. A *"*.ubal"* is an envelope file which contains inside the "real" file it protects (e.g. an image file) and a security policy that controls access to the real file. Non-protected files are normal files without an *"ubal"* envelope.

Data Policy: A policy that defines how data can be accessed or used.

Overrides: An override is a "policy evaluation decision delegation given by a data file to a specific application". For example consider the following scenario:

- A Data File 1 is associated to a policy saying that user "Alice" cannot access this data file at location "alice_home". HOWEVER, the policy defines a "permit override" set to true applicable to an application called "BT App".
- The BT App is associated to a policy saying that user "Alice" can access Data File 1 at location "alice_home" when connected to secure WIFI_BT46.
- Result: The policy of BT_App overrides the policy of Data File 1 and access is granted.




```xml
<?xml version="1.0" encoding="UTF-8"?>
<ubal_envelope>

    <header>

        <metadata extension=".png" mime_type="image/png"/>

        <policies>

            <data_policy id="1" effect="allow" location_name="alice_home" user="alice">
                <location latitude="52.056954" longitude="1.151391" radius="50"/>
                <overrides permit_overrides="true" deny_overrides="false" app="com.example.myfirstapp"/>
            </data_policy>

            <data_policy id="2" effect="deny" location_name="bt_adastral" user="alice">
                <location latitude="52.056393" longitude="1.280264" radius="300"/>
                <overrides permit_overrides="false" deny_overrides="true" app="com.example.myfirstapp"/>
            </data_policy>
        </policies>

    </header>

    <file_content>
        iVBORw0KGgoAAAANSUhEUgAAACQAAAAkCAYAAADhAJiYAAAFp01EQVR42tVYbUybVRQeuOmM4jLm
                    ...                    ...          ...
        n5OT05qbmzuW15c3QcK4TGAOz6ADXdhw2wju55z/iXjhgtO1POxRJLEk8SR7ucTzuSius5bbrAiV
        yD82aC5uQn5sqgAAAABJRU5ErkJggg==
    </file_content>

</ubal_envelope>
```


exampleFile.ubal

Figure 25 Protected data file policy

4.7.2 User behaviour App Locker Architecture

In the following background section a brief introduction to the Android architecture is presented. Then, the subsequent sections describe the *User behaviour App Locker* architecture and flowchart design.

4.7.2.1 Android System Architecture

Figure 26 shows the Android architecture (left). Android uses a Linux kernel as hardware abstraction layer for the management of memory, resources, processes, network stack, security, etc. The native libraries are compiled and preinstalled by the vendor according to specific hardware abstractions required. Native libraries include media codecs, graphics, databases, surface manager, and others.

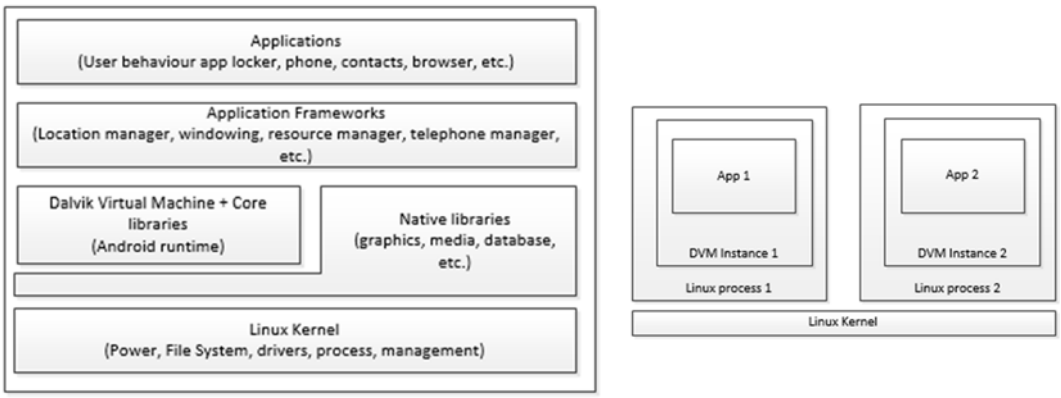


Figure 26 Android Architecture

Applications do not make calls directly to the Linux kernel, but instead they use the (android virtual machine) Dalvik which sits on top of the kernel. The android runtime environment consists of the Dalvik Virtual Machine (DVM) and a set of core java libraries. Dalvik is a Java Virtual Machine optimised for low memory usage and allows multiple virtual machine (VM) instances to be run at the same time. The application frameworks layer provides the high level building blocks to create powerful applications. Application frameworks include: activity manager, content providers, location manager, resource manager, notification manager, etc. The top layer is the applications layer and consists of all installed end-user applications, e.g. phone, contacts, web browser, "user behaviour application-locker", etc.

4.7.2.1.1 Application security sandboxing

Each application installed in Android lives in its own security sandbox. The Android architecture features a multi-user Linux system in which each application is a different user. By default, the system assigns each application a unique Linux user ID. When an application is launched it runs in its own instance of a DVM and each DVM runs in its own Linux process. This is depicted in Figure 26 (right). This security architecture provides application isolation and allows Android to enforce inter-application security constraints. Applications must request specific permissions to access device resources such as files, network, directories, and APIs in general. Also, an application cannot access the data and/or source code of another application living in a different Linux process (unless the latter provides the corresponding permissions).

4.7.2.2 User behaviour App Locker Architecture

Figure 27 describes the system architecture for the application *User behaviour App Locker*. In this section the system components are described.

4.7.2.2.1 Launch Service Detector

This component runs as a thread. It obtains the current runtime environment and accesses, reads and monitors the device's logging system in order to detect when an application or data file is about to be launched. This component informs the *Event Handler* component.

4.7.2.2.2 Event Handler

This component checks whether the application or data file being launched contains an associated (sticky) policy. If it does the *Event Handler* performs the following steps:

- Requests from the *Blocking Interface* to block the launch and requests the ID (i.e. username) of the user attempting the action.
- Upon receiving the ID of the user requesting access, the *Event Handler* makes a policy evaluation request to the *Policy Manager* including the user ID and the name of the application (to be launched). The *Event Handler* waits for the evaluation.
- Once the evaluation decision (allow/deny) is returned from the *Policy Manager*, the *Event Handler* requests from the *Blocking Interface* to allow or deny access to the user accordingly. In the case of data files, the *Event Handler* encrypts/decrypts files via the *OTFE (On-the-fly Encryption)* module.

4.7.2.2.3 Blocking Interface

From a Model-View-Controller architectural perspective, this component acts to some extent as the interface that separates the view layer (Blocking Screen GUI) from the model/controller layers. The Blocking Interface has three main functionalities:

- It receives requests from the *Event Handler* to block/unblock the resource being launched. This component achieves this by opening/closing the Blocking Screen GUI on the device's screen.
- If requested by the end-user (e.g. by pressing back button) or by the *Event Handler* (deny launch), the *Blocking Interface* can also kill the process that hosts the resource being launched.
- It obtains the end-user credentials, via Blocking Screen GUI, and executes identification/authentication tasks. When a user is satisfactorily identified and authenticated, the *Blocking Interface* sends the end-user ID back to the *Event Handler*.

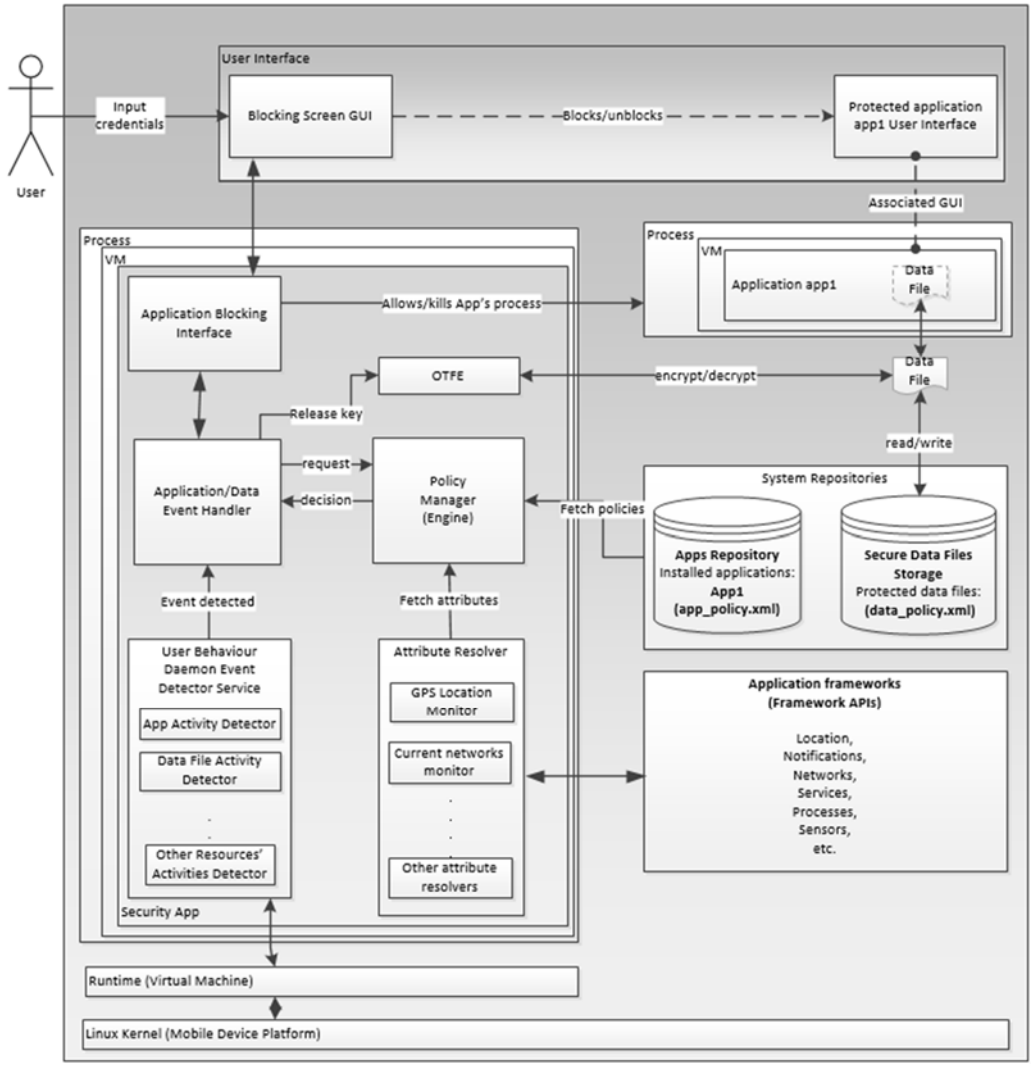


Figure 27 User behaviour App Locker Architecture

4.7.2.2.4 Blocking Screen GUI

This component is the user interface itself (window, widget, etc.) displayed on the device's screen. Its main functionality is to block protected resources (at view level) from being displayed on the device's screen. The *Blocking Screen GUI* provides a conventional login form for the end-user to enter username and credentials to be sent to the *Blocking Interface* for authentication purposes.

4.7.2.2.5 Policy Manager

This component processes policy evaluation requests from the *Event Handler* and returns an evaluation decision (allow/deny). The *Policy Manager* performs the following steps:

- Retrieves the applicable sticky policies (app_policy.xml) from the *System Repositories*.
- Requests the required attributes for policy evaluation from the *Attribute Resolver*.
- Evaluates the policies
- Returns the evaluation decision (allow/deny) to the *Event Handler*.

4.7.2.2.6 Attribute Resolver

This component resolves attributes requested by the *Policy Manager*. The *Attribute Resolver* provides the interfaces needed to communicate with the Application Frameworks APIs. This component consists of different subcomponents dedicated to obtain specific attribute values.

4.7.2.2.7 Application Frameworks (APIs)

This component provides high level APIs to access to/interact with the different resources available in the device's system. For instance in the *Android platform*, application frameworks include: activity manager, content providers, location manager, resource manager, notification manager, etc.

4.7.2.3 Flowchart design

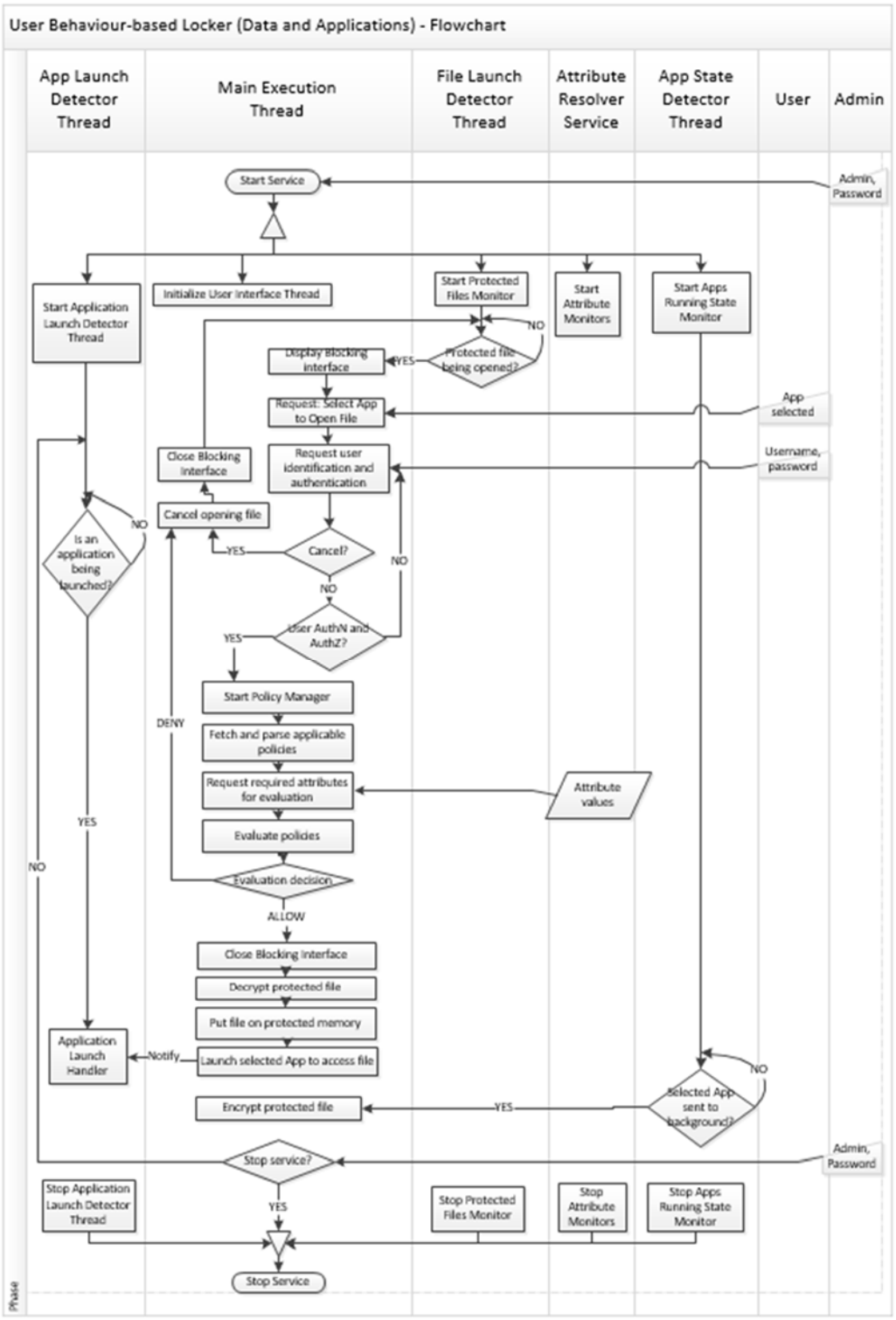


Figure 28 Flowchart design

4.7.3 *User behaviour App Locker* Demonstration of Implementation

The following example demonstrates step by step the capabilities of the *User behaviour App Locker* application.

1. Attempt to access BT_App. Enter the following credentials: ***alice/1234*** and press the login button.

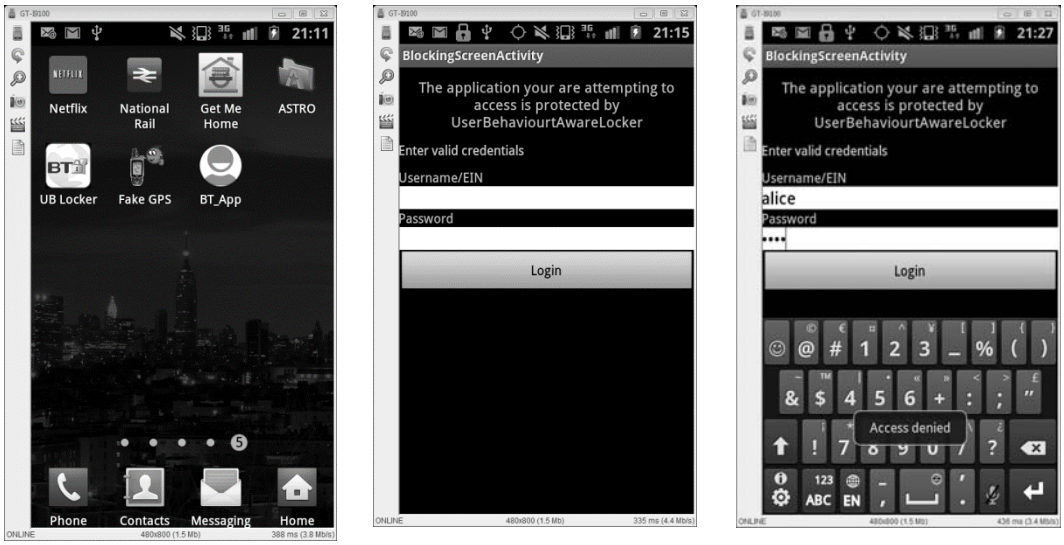


Figure 29 App Locker application screenshots: login

2. Access is denied. This is BT_App policy file:

```
<?xml version="1.0" encoding="UTF-8"?>
- <policies>
  - <app_policies>
    - <app_policy user="alice" location_name="alice_home" effect="deny" id="1">
        <location radius="50" longitude="1.151391" latitude="52.056954"/>
      </app_policy>
    - <app_policy user="alice" location_name="bt_adastral" effect="allow" id="2">
        <location radius="300" longitude="1.280264" latitude="52.056393"/>
      </app_policy>
    </app_policies>
  - <app_on_data_policies>
    - <app_on_data_policy user="alice" location_name="alice_home" effect="allow" id="1">
        <location radius="50" longitude="1.151391" latitude="52.056954"/>
        <data filename="/mnt/sdcard/download/bt_project_files/ubaltextfile.txt"/>
      </app_on_data_policy>
    </app_on_data_policies>
</policies>
```

Figure 30 App Locker application: policy example

Use the second "app_policy" applicable to location bt_adastral (which evaluates to allow). It requires you to set the correct GPS location to grant access. The location is defined by the tuple (latitude, longitude). You can use Google Maps to determine this location graphically. It corresponds to BT Adastral – Orion building.

Now use the application "Fake GPS" to set this location:

- ✓ Open Fake GPS -> point to the location -> Press "Set Location"

Figure 31 App Locker application screenshots: geolocation

3. Now attempt to access BT_App again:

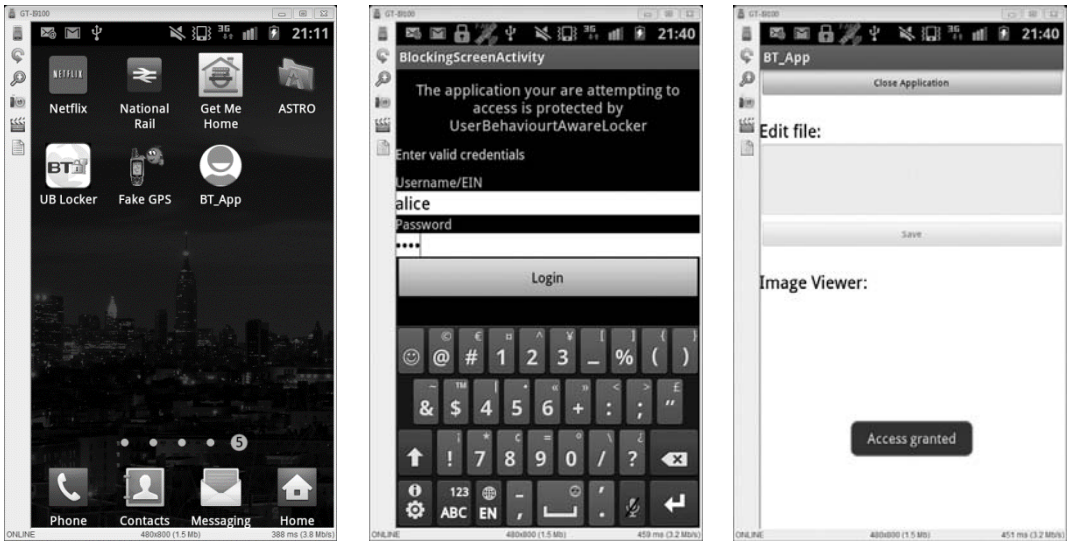


Figure 32 App Locker application screenshots: access granted

This demonstrates:

- ✓ Preventive access based on policy.
- ✓ Authentication to grant access.
- ✓ Context-based access control (GPS location).

4.8 Concluding Remarks

In this chapter it has been presented a policy-based framework that allows multiple administrative entities to define security policies on their protected resources while hosted on smart devices. The framework provides a method for policy integration and combination that

as a result produces an active execution context for a given set of resource interacting. The resulting active execution context is generated dynamically at runtime, depending on the system state and contextual information, and when the interacting entities become known. This is an important and desirable characteristic in highly dynamic, platform-open, and multi-purpose systems, such as smart devices, where predefined security profiles would be inflexible or difficult to define. The framework also provides an architecture that detects fine-granular activities triggered between resources, and that monitors and enforces active execution contexts, thus providing a level of execution context isolation. Actions initiated by entities with behaviour on resources such as services, applications, data files, etc., can be detected and dealt with providing protection against different common types of threats including viruses, malware, and data leakage.

The design and implementation of the *User behaviour App Locker* application demonstrates and validates using the proposed framework in a real scenario.

Chapter 5

Privacy and Security Requirements Enforcement Framework in Internet-Centric Services

This chapter focuses on the problem of how to protect personal data and privacy in the context of internet-centric services. Two main challenges are considered: how to enable individuals to express data protection requirements on their personal data according to the sensitivity of the data in a disclosure request; and how to ensure that personal data is actually protected and processed according to the intended purpose of use after being disclosed. As part of the proposed solution, the notion of a distinctive online service and architectural component is introduced, i.e. the Privacy and Security Broker (PSB), as the entity responsible for the protection of personal data. The PSB enables a user to express their data protection requirements and translates them into "Data Protection Property Policies" (DPPPs). A high level architecture and the corresponding protocols involving the interaction of the main actors of the solution are presented.

5.1 Introduction

Today, individuals unavoidably depend on Information and Communication Technologies (ICT) services to varying degrees in many aspects of their lives. As a result personal data is scattered across hundreds of different service domains that collect, store, process, and transmit data. Two technological advances have contributed to this: i) the proliferation of internet-centric services across all organisational sectors that have redefined the way individuals and organisations interact and exchange information; and ii) the cloud architectural model as enabler of internet-centric services characterised by its outsourcing/provisioning nature that

increases the dependency on third party services. These ICT developments create great opportunities and benefits from the societal, organisational, and technological perspectives; however, they also present complex security and privacy challenges to individuals.

Consider a recruitment company that provides internet-centric services to individuals, and processes personal information of different kinds such as personal details, curriculum vitae, health and qualifications information, etc. This information is sensitive and must be handled in a way that protects the privacy of the individual. To complicate the matter, the recruitment company may outsource parts of its functions to third part services increasing privacy and security risks.

This chapter focuses on the problem of how to protect personal data and privacy in the context of internet-centric services. Two main challenges are considered:

1. How to enable individuals, the Data Subject[1] (DS), to express data protection requirements on their personal data according to the sensitivity of the data in a disclosure request.
2. And once disclosed, how to ensure that personal data is actually protected and processed according to the intended purpose of use of the company, the Data controller[2] (DC), who offers services.

Regarding the first challenge, privacy protection laws acknowledge that not all items of personal data are equally sensitive from the point of view of their dissemination [78]. The 1998 Data Protection Act (DPA) [79] mandates that "appropriate technical and organisational measures shall be taken against unauthorised or unlawful processing of personal data and against accidental loss or destruction of, or damage to, personal data" [79]; and such measures *"must ensure a level of security appropriate to the nature of the data to be protected*" [79] (emphasis added). However, the sensitivity of data does not depend on its nature in itself only. This is because privacy is inherently contextual – "Contextual information includes: the interests of the data controller as well as the potential recipients of the data, the aims for which the data are collected, the conditions of the processing and its possible consequences for the individual and others" [80]; and because the concept of personal data does not necessarily mean that the data is especially sensitive, private, or embarrassing; instead, its significance aspect derives from its privacy value to a human [81]. "Personal information privacy is a

[1] "The Data Subject is the person whose personal data are collected, held, or processed by the Data Controller" [77] "Directive 95/46/EC of the European Parliament and of the Council of 24 of October 1995," *Official Journal of European Communities,* 1995.

[2] The Data Controller can be any service provider (SP) which acts as "an organisational entity which alone or jointly with others determines the purposes and means of the processing of personal data" [77] ibid..

property of personal information that makes it significant" [81, 82]. Additionally, in the context of internet-centric services, organisations execute business processes where flows of personal data are created from activity to activity, and as personal data propagates it is transformed and associated to new data.

It is observed that the sensitivity of data not only depends on its nature, the context in which is used, and on the value given by the data subject, but also it depends on how it is handled and used by the data controller. Sensitivity of data varies dynamically according to these several factors consequently their privacy and security requirements vary too. Moreover, it would be inefficient to protect all personal data at the same levels.

This suggest the need for a mechanism that identifies the level of sensitivity of a given data item before disclosure and once collected and processed by organisations; and one that helps the data subject apply privacy and security requirements accordingly. In this chapter, the notion of a distinctive online service and architectural component is introduced, i.e. the Privacy and Security Broker (PSB), for the protection of personal data and which provides the proposed functionalities. The PSB enables the DS to express their data protection requirements and translates them into "Data Protection Property Policies" (DPPPs). The DPPPs aims to capture the privacy and security properties required from the DC for the protection of the requested data.

Protecting personal data is not a straightforward task to the average user. It would not be realistic to expect users to think and act, for instance, in terms of the DPA principles in a data disclosure request, or to expect them to easily determine the sensitivity of a certain piece of information, or to think in terms of privacy and security properties. They lack expertise in general privacy and security related issues, let alone technical ones. General issues range from lack of meaningful consent to lack of trust to poor usability to privacy unawareness. Also, individuals are risk takers and most often underestimate the magnitude of privacy threats when revealing sensitive data or its impact further in time [83]. Therefore, in the proposed solution the PSB component plays an important role.

The second challenge considers the principle of "purpose of use" of data. This principle is well recognised and articulated in privacy and data protection laws, and it has been included in different privacy-aware access control models and policy specifications as an important factor when evaluating access decisions. However, the actual enforcement of the purpose of use specified in policies still remains a challenge. The concept of "purpose" is very ambiguous, prone to arbitrary interpretations, and therefore difficult to enforce at system level by the DC. In most privacy policies, the "purpose of use" is no more than a self-declaration made by the DC.

The "purpose of use" expresses the reason(s) as to "why" personal data is collected, processed, disclosed, etc., by the DC but it falls short in expressing "how" this is done or implemented. The present work coincides with some authors [84-86] who have taken a different approach by noting that the purpose of use can be associated to activities in a business process. The idea behind this approach is that a business process can more accurately specify the purpose of use at system level.

In the second part of the solution, it is proposed a framework in which the DC includes, as part of the data request, a system's representation of how and where data will flow in its system within the scope of the request. Such system representation consists of a business process template (BPT) and an abstract representation of the "purpose of use" here called the "Business Process Tree" (T_{BP}) and which is introduced in section 5.4. These two artefacts are used by the PSB, who acts on behalf of the DS, to apply fine-grained DPPPs. That is, appropriate data protection requirements are applied to the BPT via DPPPs based on the level of sensitivity of the data requested by the DC. In addition, the BPT-T_{BP} allows applying execution constraints to both the control-flow and the information-flow of the business process, in a natural manner.

This chapter is organised as follows. Section 5.2 presents the related work. Section 5.3 presents a high-level architecture and interaction protocol between the DS, the DC, and the PSB; and introduces a recruitment process (RP) as the motivating scenario. Section 5.4 presents the purpose of use model (T_{BP}). Section 5.5 presents the BTP and the information-flow control model. Section 5.6 describes the DPPPs model. Finally, section 5.8 presents the conclusions and further work.

5.2 Related Work

Many countries recognise the need to protect individuals' privacy and personal data from potential threats and dangers that may result in harm, unfairness, and embarrassment. For instance, the DPA [79] governs the protection of personal data in the UK and defines important privacy principles: data should be processed fairly, collection must be for a specific purpose, data should be accurate, data collected should be made accessible to data subjects, further disclosure must be prohibited unless it is consented by the data subject, data retention should be the minimal required, jurisdictional restrictions must be considered when transferring data, and appropriate security measures should be in place.

In [87], McCullagh discusses the existing categories of sensitive data in international law considering changes in societal structures and advances in technology. The EU Directive

95/46/EC [77] specifically defines categories of sensitive data: racial or ethnic origin, political opinions, religious or philosophical beliefs, trade union membership, and data concerning health or sex life. This approach is criticised as being out-dated and meaningless in some regards. The result of a survey in [80] showed that some of the categories defined as sensitive such as religious beliefs or sexual orientation are not considered that sensitive to certain communities whereas other categories such as financial information, which is not included as sensitive, is actually perceived as highly sensitive by individuals. In contrast, the Organisation for Economic Co-operation and Development (OECD) [6] does not consider designated categories of sensitive data: "...it is probably not possible to define a set of data which are universally regarded as being sensitive." "Sensitivity is no more perceived as an *a priori* given attribute. On the contrary, any personal datum can, depending on the purpose or the circumstances of the processing be sensitive" [88]. Similarly, The Council of Europe (2005) emphasises on whether the underlying purpose of the data processing is intended to reveal sensitive data. They argue that the actual processing of data, rather that the data itself, could be considered sensitive.

In [89], an experimental study was carried out consisting in the identification and measurement of enterprise sensitive data. The sensitivity estimation was based on data analytics and classification tools to categorise the unstructured text documents. Although targeted to the enterprise, the experiment above described demonstrates the potential of data categorisation and classification techniques.

Several authors have taken a different approach by noting the link between *purpose* and associated *activities*. For instance, requesting an "email" for contact purposes may entail a chain of actions such as collecting, storing, processing, and/or disclosing [90]. Another approach is that of [84] that argues that the purpose of access is associated to a workflow in which the access takes place. The idea is that workflows can more accurately specify a purpose.

In [85] a purpose presumes a (business) goal and subsequently it refers or corresponds to an action or a set of actions. A formal semantic model for "purpose of access" is developed for a business system. In their framework a common vocabulary for referring to system's actions is proposed that provides the terminology to be used by policies and business processes in the system. Standard action vocabularies exist for different domains. Additionally, an *action graph* is proposed where action names are taken from the vocabulary to add semantics to the business system, and to define relationships between actions that reflect the behaviour of the system.

PrimeLife introduced the PrimeLife Policy Language (PPL) which allows expressing access and usage policies in XML format. Users express privacy preferences while service providers (SPs) express privacy policies. Before disclosing data, privacy preferences must be matched by privacy policies and the resulting policy is the sticky policy [91] which is bound to the data as it moves across different SPs. The sticky policy defines: i) access control policies based on an extended version of the XACML language and ii) data handling (usage) policies including the following elements: authorisation purpose, downstream usage, and obligations. Relevant to this discussion are authorisation purposes: a type of authorisation to use data for a particular set of basic purposes such as purchase, administration, contact, marketing, etc. Purposes are refer to by URIs specified in agreed-on vocabularies of data usage purposes, that can be organised as flat lists or as hierarchical ontologies.

Technical and conceptual initiatives with distinctive approaches to user-centric data management models have started to appear in the past few years [92-94]. They are still in their early stages of maturity and therefore represent a research opportunity. However, such approaches mainly focus in data-portability and access-control aspect and do not address the protection of data during its full life-cycle including its protection even after disclosure.

This proposal differs from the work above by including the definition of privacy and security requirements based on different factors data sensitivity depends on along with the modelling of an information-flow control policy to be enforced on a business process. The proposed framework includes the typical access control aspects of privacy policies such as purpose. It is also proposed the notion of a new architectural component as mediator between data subject and data controller in the definition of privacy and security requirements. Moreover, it is proposed to represent a data protection policy of a data controller as embedded in a business process template which in turn represents the intended use of data at business process activity level.

5.3 Proposed approach

In this subsection, first a high level architecture and the corresponding protocol involving the interaction of the main actors of the solution is presented; then, a recruitment process scenario is used in subsequent sections as the running example throughout the chapter.

5.3.1 Actors, architecture and interactions

The architecture and actors, i.e. DS, DC, and PSB, are shown in Figure 33. As an example, consider a client (DS) who uses the services provided by a recruitment company (DC) via the mediation of another company that provides the services of the PSB. Here an additional new actor is introduced, a data host (DH), to refer to a component that hosts personal data of the DS, for example, a health centre (DH) which hosts electronic health records about the client.

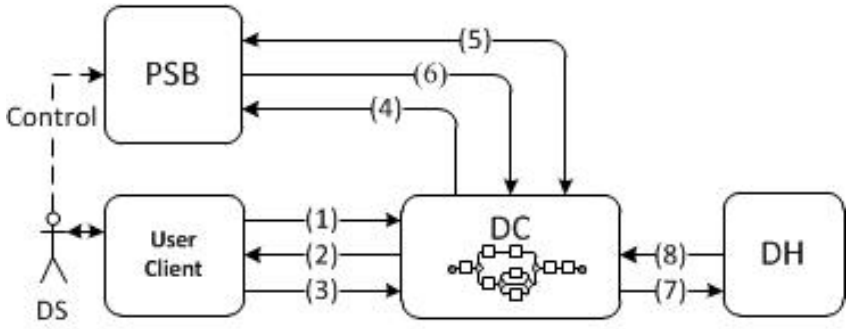


Figure 33 Proposed High-level Architecture

The interaction between the components consists of the following steps: (1) the DS sends a *service request* to the DC; (2) the DC replies to the DS with a *data request* requesting the data resource(s) required for the service provision; (3) the DS redirects the DC to the PSB. The DS relies on the PSB for the protection of personal data; (4) the DC makes a data request to the PSB in the form of a *business process instantiation request* including the BPT, T_{BP}, and associated data handling policies. Note that the BPT-T_{BP} represents the intended use of data by the DC. As mentioned in the introduction, the BPT-T_{BP} represents a well-defined business process specification that the DC intends to execute to provide the service to the DS. The BPT-T_{BP} specifies business process activities with their respective roles, actions, and data input/outputs assignments, and additionally it specifies the control-flow and data-flow within the business process. The T_{BP} is the representation of the "purpose of use" of data in the business process. Therefore, the BPT-T_{BP} includes the typical aspects of a privacy-aware access control policy delimited within the business process. More clearly, the BPT-T_{BP} is the privacy policy of the DC; (5) the PSB, acting on behalf of and under the control of the DS, has two main functions: (i) to perform the *instantiation of the BPT* by evaluating the applicable disclosure policies on the data resource(s) requested, and applying information-flow and control-flow constraints on the BPT. Note that the PSB does not host the requested data, instead, it acts as a policy decision point (PDP) for the evaluation of policies and therefore it is able to produce authorisation tokens which can be used by the business process's activities to access the requested resources wherever they may be hosted; and (ii) to determine the sensitivity of the authorised data to be consumed by the business process, and to define the

enforceable (by the DC) *Data Protection Property Policies (DPPP)*. Additionally, step 5 may include a constraints/policy matching and negotiation sub-protocol in order to resolve conflicting constraints/policies; (6) the PSB then sends to the DC the *authorised BPT Instance*, the DPPP policies, and the corresponding *authorisation tokens* required for granting access to the requested data resources; (7) *later*, during execution of the business process, the activities use their corresponding authorisation tokens to make data requests to the data hosts (DHs). A DH is any entity at any location where the requested resources are hosted; and (8) The DH, acting as Policy Enforcement Point (PEP), grants or denies access to the requested resource(s) according to the authorisation decision. It is assumed that DHs act as PEP to the PSB to simplify things although it may not be the case. However, this is out of the scope of this chapter.

It is important to notice that in step (6) the PSB sends the DPPP policies and the corresponding *authorisation tokens*; however, because the nature of data is contextual and circumstantial, its level of sensitivity may be change over time and by the time the business process is actually executed the DPPP policies will need to be redefined and reissued by the PSB with valid *authorisation tokens*.

5.3.2 Example Scenario: Recruitment Process

As an example scenario, throughout this chapter a recruitment process (RP) is considered in which a client, the DS, registers with a recruitment company, the data controller (DC), in order to find employment. Personal data processed in a typical recruitment scenario includes: curriculum vitae (CV), transcript of records (ToR), and electronic health records (EHR). CVs contain employment history, qualifications and personal identifiable information (PII) such as name, contact details, etc. EHR include information such as demographics, medical history, medication and allergies, immunisation status, laboratory test results, vital signs, etc[3]*. ToR are used to document the academic performance of an individual over a certain period of time by listing the course units or modules taken, the credits gained, and the grades awarded*[4]*. These data can be highly sensitive. It is assumed that the recruitment company requires this information in order to establish the DS's elegibility for available jobs and to produce a report of vacancies.*

The recruitment company specifies RP as a business process. A business process describes processes as structured sequences of activities that must be performed in order to accomplish a given goal. Activities are control and data interdependent and must be executed by

[3] http://en.wikipedia.org/wiki/Electronic_health_record
[4] http://en.wikipedia.org/wiki/Transcript_(education)

authorised subjects on authorised objects, and in the sequence specified by the business process. Therefore, execution and security policy must be synchronised.

Two examples of business process specifications are the Business process Modelling Notation (BPMN) [95] and the Business process Execution Language for Web Services (BPEL4WS) [96], or BPEL. Although very similar in expressiveness as they share a core set of constructs, BPMN is more suitable for modelling business processes while BPEL focuses on process execution in service oriented architectures (SOA). BPEL supports two types of service composition: service orchestration and service choreography. In service orchestration a process acts as the central coordinator to activities (or to other processes) that it interacts with. In service choreography, there is not a central orchestrator process. Instead, each activity knows the operations that it must execute within the business process and the activities it must communicate with. This work consider an orchestration workflow.

The RP as a business process is shown in a BPMN-like notation in Figure 34.

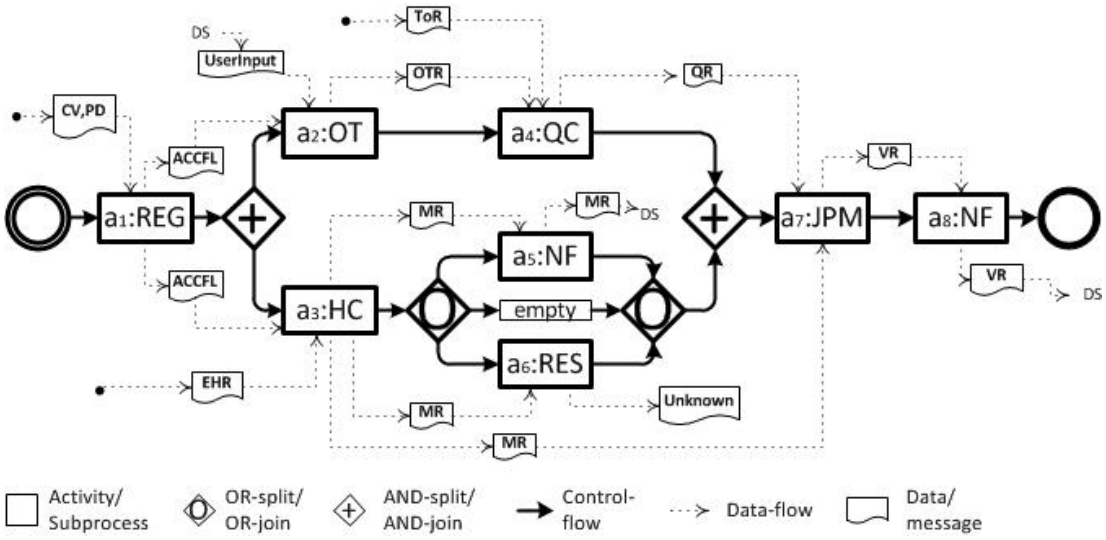


Figure 34 The Recruitment Process in a business process modelling

The diagram includes control-flow (solid lines) and data-flow (dotted lines) dependencies. Here, structured business process modelling [97] is considered where activities are structured between nested control connectors AND-split/AND-join, and OR-split/OR-join[5]. The notation of the form a_i: WS_i means that activity a_i invokes an operation supported by a web

[5] And-join connectors wait for all preceding activities to be finished before the subsequent activity can be started. Or-join connectors wait for any preceding activity to be finished before the subsequent activity is started. And-split create parallel flows. (Inclusive-) Or-split connectors create alternative execution paths.

service WS_i. The Web Service Description Language (WSDL)[98] is an XML-based language used to describe web services. Web services provide interfaces describing the operations that can be performed. Invocations are done by the process orchestrator, i.e. RP, in an orchestrator BPEL engine. In what follows a detailed description of the recruitment process RP. Note that this description is assumed to be encoded into a BPEL specification, which corresponds to the business process template (BPT) intended to be executed by the recruitment company for the provision of its services to the DS.

First, the RP invokes the web service REG (registration) which corresponds to the execution of activity a_1. Activity a_1 takes as input the personal details (PD) and the curriculum vitae (CV) of the client and creates a client account file (ACCFL). Then, the RP executes activities a_2 and a_3 concurrently. This initiates two sub-flows of execution. The first sub-flow consists in the execution of activity a_2 followed by activity a_4 that corresponds to the invocation of web services OT (online test) and QC (qualifications check), respectively. The OT provides the client with an interface to take an assessment test as part of the recruitment process and thus activity a_2 takes as input UserInput and outputs a document with the online test results (OTR). Once this is done, the RP continues the sub-flow of execution with activity a_4. This activity takes the inputs OTR and ToR (transcript of records) of the client that are used by a clerk (here the role QAClerk) to generate a qualifications report (QR). This terminates the first flow of execution. The second sub-flow starts with the execution of activity a_3, which corresponds to the invocation of the web service HC (health check). This activity takes as input the electronic health records (EHR) of the client that is processed by a medical doctor (i.e. the role of a GP) who produces a medical report (MR) within the system. Then, the sub-flow proceeds to three additional multi-choice execution paths. These are activities a_5, a_6, and "empty". Activity a_5 corresponds to the invocation of the web service NF (notification). This activity takes as input the medical report (MR) and sends it to the client if requested. Activity a_6 corresponds to the invocation of the web service RES (medical research). This activity takes as input the medical report (MR) and uses it for medical research purposes and its output is "unknown". Activities a_5 and a_6 are optional. If none of these activities are executed, the activity "empty" is executed in order to guarantee that the second sub-flow terminates. Both sub-flows must terminate before activity a_7 can be executed (the AND-join synchronises them). Activity a_7 corresponds to the invocation of the web service JPM (jobs-profile matching). This activity takes as inputs the medical report (MR) and the qualifications report (QR), which are both used by a recruitment agent (i.e. role of the RecrAgent) who produces a vacancies report (VR). Finally, the RP invokes the web service NF (notification), which corresponds now to the execution of activity a_8 and which takes as input VR and sends it to the client.

5.4 Purpose of use model

The concept of "purpose" in most privacy policies is very ambiguous, prone to arbitrary interpretations, and therefore difficult to enforce at system level. Although included in many privacy policy models, the "purpose" is no more than a self-declaration.

As acknowledged in previous research [84-86], the concept of purpose in privacy policies can be associated to an activity or set of activities performed by a business process in order to accomplish a business goal; for instance, in the running example, it could be said that the recruitment process requires the *electronic health records (EHR)* of the client for the purpose of performing the activity *health check (HC).* Moreover, the activity *HC* can be seen as part of a higher-level abstract activity: the recruitment process. This is natural because organisations are structured hierarchically according to business functions and goals. For example, the activities of the recruitment process (in Figure 34) can be grouped into three abstract activities: *account management* (AM) referring to admin-related activities consisting of *registration (a_1: REG), job-profile matching (a_7: JPM), and notification (a_8: NF)*; *health centre assessment* (HCA) referring to healthcare-related activities consisting of *health check (a_3: HC), notification (a_5: NF), and medical research (a_6: RES)*; and *qualification assessment* (QA) referring to qualification-related activities consisting of *online test (a_2: OT) and qualifications check (a_4: QC)*. This is represented in Figure 35.

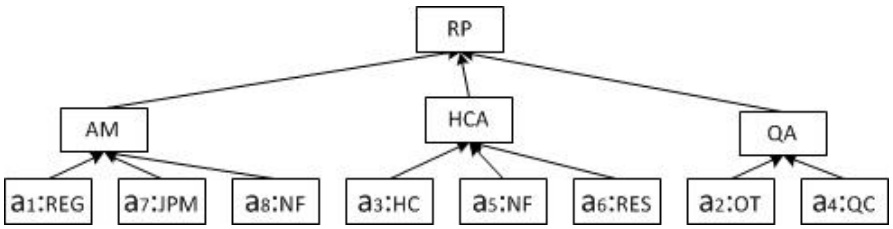


Figure 35 Business Process Tree

The purpose of use in data protection policies expresses the reasons as to why personal data is collected, used and handled. Activities of a business process specify how this is done. Here, the "purpose of use" is represented as activities at different levels of abstraction in a business process.

One advantage of this grouping of activities from high to low level is that it eliminates ambiguities when expressing a purpose. For example, the purpose statement "we collect your email address for the purpose of notification" does not distinguish whether it is referring to activity a_8: NF or a_5: NF. Instead, by using a hierarchy of activities, the distinction can be expressed, for example, by writing notification as RP/AM/NF meaning that *email address* can be used to *notify* (a_8: NF) the client as part of activity *AM*, which is in turn part of RP. Another

advantage is the flexibility to define policies at different levels of abstraction: using the same example, RP/AM would mean that email address can be used by all activities under the abstract activity AM as part of RP. Finally, the most important advantage is the fact that purpose can be linked directly to actual activities of a business process. (Note however the trade-off between the second and the third advantage.)

Definition 1 (Business Process Tree - $\boldsymbol{T_{BP}}$) A business process tree $\boldsymbol{T_{BP}}$ of a business process $\boldsymbol{BP}$ is a set of activities $\boldsymbol{B} = \{\boldsymbol{A_1}, \boldsymbol{A_2}, \dots, \boldsymbol{A_n}\}$ and a relation $<_{\boldsymbol{B}} \subseteq \boldsymbol{B} \times \boldsymbol{B}$ where $<_{\boldsymbol{B}}$ is a partial ordering of $\boldsymbol{B}$ and for any $\boldsymbol{A_i} \in \boldsymbol{B}, \{\boldsymbol{s} \in \boldsymbol{B} | \boldsymbol{s} <_{\boldsymbol{B}} \boldsymbol{A_i}\}$ is well ordered. $\boldsymbol{T_{BP}}$ is a hierarchical representation of a business process BP with its root being BP as the highest level activity, its intermediate nodes being abstract activities, and its leaves being the concrete set of activities in BP. It is said that an activity is part of its higher level activities. $\boldsymbol{BP}$ is used to refer to all activities in $\boldsymbol{B}$; and $\boldsymbol{BP/A_i/}$ ... to refer to $\boldsymbol{A_i}$, or to all the activities under $\boldsymbol{A_i}$.

5.5 Modelling information flows

One functional requirement of the proposed model is the ability to protect data when it propagates from activity to activity. As personal information flows through the system its sensitivity changes, and therefore its privacy and security requirements vary dynamically. In this section, firstly, the concept of static analysis of information-flow security (IFS) systems [99] is briefly presented, followed by a discussion on how it can be used to control information-flow in a business process. Secondly, the formalisms of the proposed information-flow control policy model is presented. The recruitment process is used as an example.

5.5.1 Introduction to Information-flow Security systems

Information-flow security (IFS) systems allow tracing information and the identification of illegal flows in a system. Consider the line of code $\boldsymbol{a = b + c}$, where a, b, and c, are program variables. The "$+$" operation takes as input c and b and assigns the result to variable a. Clearly, there is a flow of information from b and c to a. If $\boldsymbol{a', b', and\ c'}$ are labels (e.g. indicating data types as in conventional programming languages) associated to $\boldsymbol{a, b, and\ c}$, respectively, then, a static IFS system would check whether the flows $\boldsymbol{b' \rightarrow a' and\ c' \rightarrow a'}$ are *legal flows* before allowing to write to memory. In the case of a dynamic IFS system, the system would dynamically compute from the operation (i.e. "+"), b', and c' a new label a' and would

associate it to variable a, and then the system would check whether the flow $\boldsymbol{a' \rightarrow C'}$ is a legal flow before allowing variable $\boldsymbol{a}$ to be written into a channel C, where C' is its label.

5.5.2 Revisiting the recruitment process scenario: the concept of Form

In the running example, the recruitment company requires the client's electronic health records (EHR) to perform different health checks. Let consider the data flow from activity a_3 to activity a_7: Health Check (HC) takes as input the EHRs and outputs a Medical Report (MR), and Job-Profile Matching (JPM) takes as input. This is depicted in Figure 36.

Figure 36 Data-flow from HC to JPM

Suppose HC requires the full EHRs in order to establish the client's physical and mental fitness. Once HC outputs MR, new medical information about the client may be either created and associated to personal identifiable information (PII) of the client, or inferred by aggregating and mining the client's information (EHR) with other inputs (e.g. in the case that the activity takes another inputs to accomplish its functions). From the perspective of the client, determining the set of new information that could possibly be associated to personal data is a complex challenge. One reason is that the organisation that performs the business activity may not be willing to disclose private information about additional inputs or algorithms used by the activity. This approach is not discussed further. Instead, a different mechanism is considered which consists in the encapsulation of the activity output(s) associated to personal data into a predefined document or form. Here, the trust assumption is to consider the activity as a black box where the outputs can be controlled.

As MR is the output from HC, it may contain information of different sensitivity, e.g. *eye test* (low), *chronic condition* (medium), *mental illness* (high), *etc*. From input to output, an activity may modify (increase or decrease) the level of sensitivity of the information that passes through it. Therefore, as information flows from activity to activity, the system should adapt and enforce privacy and security requirements accordingly. On the other hand, JPM, which uses MR as input to its process, may not require all the information contained in MR. For example, the information required by JPM may depend on current available job vacancies (which vary in time): JPM may only require access to *eye test* in order to consider the applicant for a currently available job as *driver*. Therefore, rather than passing directly the output (MR)

from HC to JPM, the system should be flexible enough to allow JPM to take as input information from MR on a need-to-know basis. This would allow the enforcement of privacy and security requirements to be flexible and the utilisation of protecting resources to be more efficient.

In the proposed design, the concept of a *form* is used as an intermediate step in the information-flow (as shown in Figure 37). The assumption is that since activities of a well-defined business process are being considered, it is possible to know beforehand what the output or possible outputs of an activity will be. For instance, the recruitment process can specify the output MR as an XML template to be instantiated by the activity HC once executed. Table 2 shows the conceptual representation of the form MR consisting of a form identifier and a structured list of data categories specifying data category name identifiers and their respective values or references to the values. This is very similar to the concept of static analysis in IFS systems where the output writes to a channel with a predefined security level label. The concept of *form* is also very similar to the concept of asynchronous *channel*[6] in programming.

In the proposed system, a form specifies the binding between an activity output and an activity input has the following functions: (i) to encapsulate the output from an activity; (ii) it is used as a channel for the flow of information between activities associated to the form; and (iii) it is used to logically group the data items an output consists of. In the next subsection, the proposed information-flow control model is presented and it includes the concept of form.

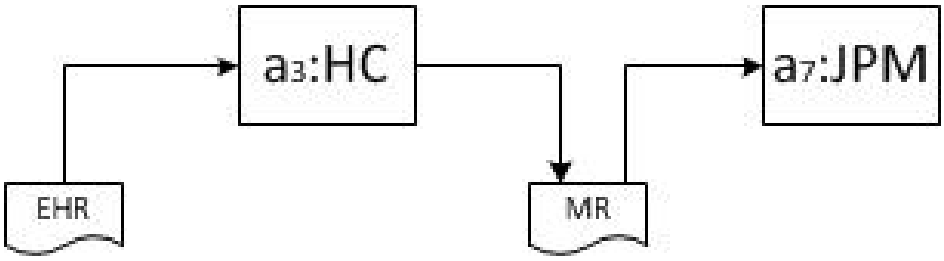


Figure 37 Data-flow from HC to JPM via form MR

[6] "An *asynchronous channel* represents a connection that supports non-blocking operations, such as connecting, reading, and writing". http://www.ibm.com/developerworks/java/library/j-nio2-1/index.html

Table 2 Example MR Form

Form: Medical Report (MR)		
Name	**Type**	**Value**
Client-name	PII	John Smith
Email-address	PII/Contact	aaa@aol.com
Eye test	Medical	Xlink: eye-test
Chronic illness	Medical	Asthma
Disabilities	Medical	NA

5.5.3 Proposed Information-flow control model

This section presents all the formalisms. First, the definition of a *form* is presented followed by the definition of p*ermissions,* which allow expressing restrictions on the *actions* that can be performed on a *form*. Then a *business process* is formalised and the proposed information-flow control model is presented. The policy model is not only an access control policy because it also allows specifying restriction to the output(s) of an activity.

Definition 2 (Form) A form $\boldsymbol{F}$ is a tuple $(\boldsymbol{F_{ID}}, \{\boldsymbol{d_1}, \boldsymbol{d_2}, \dots, \boldsymbol{d_m}\})$ where $\boldsymbol{F_{ID}}$ is the form identifier and $\{\boldsymbol{d_1}, \boldsymbol{d_2}, \dots, \boldsymbol{d_m}\}$ is a set of data items partially ordered and each data item $\boldsymbol{d_i}$ is a tuple $(\boldsymbol{category}, \boldsymbol{name}, \boldsymbol{value})$. A form is any data structure containing a collection of data elements items, for instance, a structured or semi-structured XML document. $\boldsymbol{F_{ID}}$ is written to refer to (the data items in) $\boldsymbol{F}$, $\boldsymbol{F_{ID}/d_i}$ to refer to a data item $\boldsymbol{d_i} \in \boldsymbol{F_{ID}}$, and when appropriate $\boldsymbol{F_{ID}/d_{i1}/d_{i2}/}$ ... is written to refer to the data items under $\boldsymbol{d_{i2}}$, meaning that there exists a partial ordering relation $\leq_{\boldsymbol{F}}$ between data items.

Definition 3 (Permission) A permission *P* is a tuple $(\boldsymbol{F_i}, \boldsymbol{act})$ where $\boldsymbol{F_i}$ is a form as defined in definition 2 and $\boldsymbol{act}$ is the type of action that can be performed on $\boldsymbol{F_i}$. Actions $\boldsymbol{act}$ can be of types create, read (retrieve), update (modify), and delete (destroy). In future this can be extended to include other action types.

Definition 4 (Business Process) Let BP be a business process consisting of a set of activities $\boldsymbol{B} = \{\boldsymbol{A_1}, \boldsymbol{A_2}, \dots, \boldsymbol{A_n}\}$ as defined in $\boldsymbol{T_{BP}}$ (definition 1), a set of control-flow

edges $\boldsymbol{E_{CF}} \subseteq \boldsymbol{A} \times \boldsymbol{A}$, a set of forms $\boldsymbol{\mathcal{F}} = \{\boldsymbol{F_1}, \boldsymbol{F_2}, \dots, \boldsymbol{F_l}\}$, a set of data input edges $\boldsymbol{E_{DInput}} \subseteq \boldsymbol{F} \times \boldsymbol{A}$, a set of data output edges $\boldsymbol{E_{Doutput}} \subseteq \boldsymbol{A} \times \boldsymbol{F}$, and a set of roles $\boldsymbol{\mathcal{R}} = \{\boldsymbol{r_1}, \boldsymbol{r_2}, \dots, \boldsymbol{r_K}\}$ hierarchically organised under a partial order relation $\leq_{\boldsymbol{R}}$. A business process description corresponds to the tuple $< \boldsymbol{B}, \boldsymbol{E_{CF}}, \boldsymbol{\mathcal{F}}, \boldsymbol{E_{DInput}}, \boldsymbol{E_{Doutput}}, \boldsymbol{\mathcal{R}} >$.

Definition 5 (Information-flow Policy) An information-flow control statement is a tuple $(\boldsymbol{A_i}, \boldsymbol{F_{IN}}, \boldsymbol{F_{OUT}})$ where activity $\boldsymbol{A_i} \in \boldsymbol{B}$ is assigned a set of data inputs $\boldsymbol{F_{IN}} \subseteq \boldsymbol{\mathcal{F}}$, a set of data outputs $\boldsymbol{F_{OUT}} \subseteq \boldsymbol{\mathcal{F}}$. An information-flow control policy $\boldsymbol{Pol_{Flow}}$ is a set of information-flow control statements in BP.

Definition 6 (Role-permissions Statement) A role-permissions statement in activity $\boldsymbol{A_i}$ is a tuple $(\boldsymbol{r}, \boldsymbol{P_r}, \boldsymbol{A_i})$ where role $\boldsymbol{r} \in \boldsymbol{\mathcal{R}}$ is assigned a set of permissions $\boldsymbol{P_r} = \{\boldsymbol{P_1}, \boldsymbol{P_2}, \dots, \boldsymbol{P_K}\} \subseteq \boldsymbol{\mathcal{P}}$ where $\boldsymbol{\mathcal{P}}$ is the set of permissions defined for the activities in BP. A role-permissions statement assignment is legal if it respects the information-flow policy: permissions must apply only to $\boldsymbol{F_{IN}}$ in $\boldsymbol{A_i}$ and to $\boldsymbol{F_{OUT}}$ in $\boldsymbol{A_i}$, as authorised by $\boldsymbol{Pol_{Flow}}$. The set of all role-permissions statement in BP is denoted as $\boldsymbol{RP_{BP}}$.

5.5.4 An example

In the proposed model, the flow of information between activities is controlled indirectly via *forms*; for instance, consider the information-flow control statements $(\boldsymbol{HC}, \{\boldsymbol{EHR}\}, \{\boldsymbol{MR}\})$ and $(\boldsymbol{JPM}, \{\boldsymbol{MR}, \boldsymbol{QR}\}, \{\boldsymbol{VR}\})$ (see Figure 34) where health check (HC) has as input *electronic health records (EHR)* and as output *medical report (MR)*; and *job-profile matching (JPM)* has as input *MR* and *qualifications report (QR)*, and as output *vacancies report (VR)*. These two statements allow a flow of information from activity *HC* to activity *JPM* via *MR*. In a way, *MR* can be seen as a data sharing contract between *HC* and *JPM* where only the information specified in *MR* can be part of the flow.

In addition, the role-permissions statements allow controlling the flow of information from activity to form; for instance, consider again the information-flow control statement $(\boldsymbol{HC}, \{\boldsymbol{EHR}\}, \{\boldsymbol{MR}\})$, and the permission statements $(\boldsymbol{GP}, \{(\boldsymbol{EHR}, \boldsymbol{read}), (\boldsymbol{EHR}, \boldsymbol{update})\}, \boldsymbol{HC})$ and $(\boldsymbol{Nurse}, \{(\boldsymbol{EHR}, \boldsymbol{read})\}, \boldsymbol{HC})$. The two statements allow both *GP* and *Nurse* to read from *EHR* in activity *HC*. This implies an inbound information-flow with respect to the activity. However, only the GP can make updates to *EHR* which in turn implies an outbound information-flow respect to activity *HC*.

To end this section, the *business process instantiation request* is defined.

Definition 7 (*Business process instantiation request*) A *business process instantiation request* is a tuple $(BP, T_{BP}, Pol_{Flow}, RP_{BP})$ as per definitions 1-6, where BP corresponds to the business process template (BPT).

In the presented running example, the BPT corresponds to the recruitment process that the data controller intends to execute when handling the client's personal data. The business process instantiation is sent to the Privacy and Security Broker.

The next section presents the data protection property policies.

5.6 Data protection property policy model

The second functional requirement of the proposed framework is the ability to protect data according to its level of sensitivity by requiring the data controller to put/have in place appropriate security measures when handling personal data.

As mentioned in the introduction, the sensitivity of data depends on several factors: the nature of the data itself, for example, financial or medical; the value given to data by a human; contextual factors surrounding a disclosure; and the purpose of use and its further use.

On one hand, regarding to data sensitivity, the nature of data corresponds to different data categories of sensitive data while the value given by a human to data is circumstantial.

On the other, in [100], the author defines context as an interactional problem where contextuality is a relational property that holds between objects or activities, and argues that context is a dynamic feature that arises from activities. In the proposed framework, the purpose of use in a business process is defined as an abstraction to activities. Therefore, there exists an intrinsic relationship among purpose, context, and activity.

Based on these observations, in the proposed model, a relation consisting of category, activity and sensitivity level is considered as a mechanism to characterise data and on which to define data protection properties.

The concept of labels is used to characterise *forms* or data items, and also to characterise the activity requesting access to that *form* or data item.

5.6.1 An example

The DPPP model can be explained in the following steps: (i) in the presented running scenario, suppose the data item *mental illness* is part of the form *electronic health records (EHR)* and

it is requested by the recruitment process to be consumed by the activity *health check (HC)*. Also, suppose *mental illness* is assigned the data label($category = medical,\ activity - type = healthcare,\ sensitivity = high$). This is assigned by the PSB or the client, and can be predefined; (ii) based on the data label, the PSB defines an *activity label* defining data protection properties, e.g. (confidentiality, high) where high means a high level of assurance protection; and (iii) the PSB allows access if the activity can fulfil the data protection properties.

Following, the formalisation of the proposed data protection property policy model is presented.

Definition 8 (Data Labels – DL) A data label DL is a tuple (C, A_i, S) where C is a data category, $A_i \in B$ is an activity in BP, and $S \in \{low, medium, high\}$ is a level of sensitivity value.

Definition 9 (Data Protection Property Activity Label– $DPPActLabel$) A DPP Activity Label is a size-variable $vector$ of the form $\{(property_1, L), (property_2, L), \ldots., (property_n, L)\}$ where $property_i$ is a data protection property such as confidentiality, integrity, authentication, authorisation, audit, unlinkability, etc.; and $L \in \{low, medium, high\}$ is an assurance level that characterises $property_i$.

Definition 10 (Data Protection Properties Policy – DPPP) A Data Protection Property rule is a tuple $(DL,\ DPPActLabel,\ type)$ where DL is a data label assigned to a form $F_i \in \mathcal{F}$ (meaning the data item(s) in F_i), $DPPActLabel$ is a Data Protection Property Activity Label assigned to an activity $A_i \in B$, and $type \in \{allow, deny\}$. We say that activity A_i is granted/denied access to F_i with data label DL if A_i fulfils the data protection properties defined in $DPPActLabel$. A Data Protection Properties Policy (DPPP) is a set of data protection property rules.

5.7 Privacy threat analysis: Validation of the Proposed Framework

To prove that the proposed framework is effective and fulfils the system privacy requirements that it is designed for, a threat analysis approach is used. Threat analysis enables the analysis of a given target system in order to model and elucidate security and privacy threats, and consequently identify and apply adequate countermeasures.

The remaining of the section is organised as follows. First, an existing threat analysis framework for privacy requirements is presented and described in detail; second, the two main challenges of the proposed framework (Privacy and Security Requirements Enforcement Framework in Internet-Centric services – section 5.1) are expressed as high-level system

privacy requirements and used to formulate a set of concrete system privacy criteria; and third, the threat analysis framework presented is used to analyse and show how the work of this chapter can be applied to fulfil the system privacy criteria.

5.7.1 LINDDUN: A Privacy Threat Analysis Framework

LINDDUN[101] is a methodological threat analysis framework for the elucidation and fulfilment of privacy requirements of software-intensive systems and the selection of privacy-enhancing technologies. It has been chosen because of several reasons: it specialises specifically on privacy requirements; second, it is based on Microsoft's STRIDE[102], an industry-graded threat analysis framework (that focuses on security requirements); third, LINDDUN incorporates an information flow oriented model based on Data Flow Diagrams (DFD) – an standardised modelling notation, to guide analysis; and finally, LINDDUN can be integrated into the Secure Development Lifecycle (SDL)[103] – a well-established security analysis methodology in security software engineering.

5.7.1.1 The LINDDUN Methodology

Figure 38 depicts the building blocks of the LINDDUN methodology and the associated supporting knowledge for each step. In what follows a detailed description of the methodology is presented.

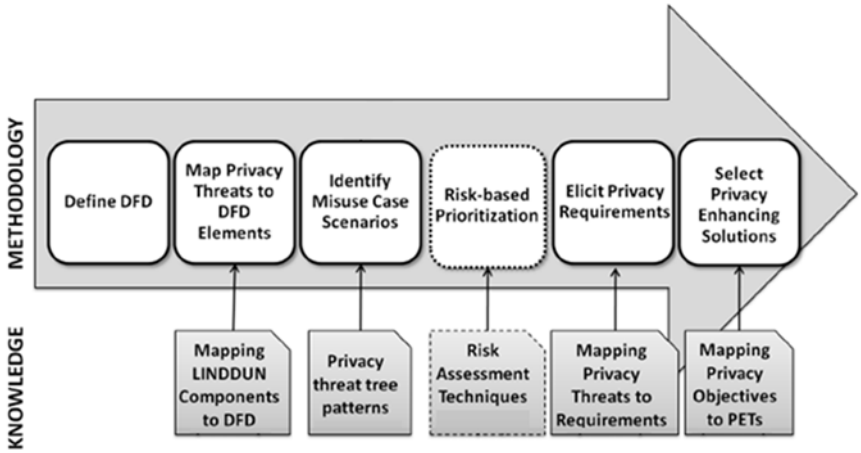


Figure 38 LINDDUN Methodology[101]

5.7.1.1.1 Defined Data Flow Diagrams (DFD)

A high-level system description is used to create one or more DFDs of the system to be analysed. The DFD notation consists of the following elements: ***Entity (E).*** A source of data or a destination of data, e.g. a user; ***Process (P)***: A process or task performed by the system; ***Data Store (DS)***: A place where data is held between processes; ***Data Flow (DF)***: An arrow

that indicates the communication of data; ***Trust boundary***: To indicate the border between trustworthy and untrustworthy elements**.** ***Control flow***: To indicate the control flow between processes.

The last two elements are not part of the DFD standard notation. However, they are included as enhancements to the DFD notation to help analysis. *Trust boundaries* are introduced in LINDDUN while *Control flows* are introduced in this thesis.

5.7.1.1.2 Map Privacy Threats to the DFD elements

LINDDUN provides a taxonomy of privacy properties compliant with the terminology proposed in [104] – recognised by the research community. The privacy properties are categorised as hard and soft, as proposed in [105]. *Hard privacy* is based on the concept of *data minimisation*; that is, the assumption that a data provider (DP) discloses either no information or as little information as possible to third parties in order to reduce the need to trust other entities. On the contrary, s*oft privacy* is based on the concept of trust; that is, information is disclosed to third parties with the consent of the DP and for specific purposes.

LINDDUN defines a list of privacy threat types based on the taxonomy of privacy properties. Each privacy threat type corresponds to *the negation of a privacy property*. This is shown in Table 3. Due to limitation of space only privacy threat types are explained from the perspective of an attacker.

Table 3 Privacy properties and privacy threats[101]

	Privacy properties	Privacy threats
HARD	Unlinkability	**L**inkability
	Anonymity & Pseudonymity	**I**dentifiability
	Plausible deniability	**N**on-repudiation
	Undetectability & Unobservability	**D**etectability
	Confidentiality	**D**isclosure of information
SOFT	Content awareness	content **U**nawareness
	Policy and consent compliance	policy and consent **N**oncompliance

Linkability: An attacker can sufficiently distinguish whether two items of interest (IOI, e.g. messages, actions, etc.) are related or not within the system.

Identifiability: An attacker can sufficiently identify the subject associated to an IOI.

Non-repudiation: An attacker can gather information to prove that a user knows, has done or said something.

Detectability: An attacker can sufficiently distinguish whether an IOI exists or not.

Information Disclosure: An attacker gains access to private information about the user.

Content Unawareness: The user is unaware of information disclosed to the system.

Policy and Consent Non-compliance: The system does not enforce the privacy policy.

According to LINDDUN, each DFD element type (see 5.7.1.1.1) is subject to certain privacy threat types (Table 3). For instance, a *Data Store* element type is subject to *Information Disclosure* threats. In [101], a comprehensive formal analysis on how privacy threats affect DFD element types is provided. Table 4 summarises the result of this analysis and provides a generic mapping to guide system-specific threat analysis systematically.

Table 4Mapping privacy threat types to DFD element types[101]

Threat categories	E	DF	DS	P
Linkability	×	×	×	×
Identifiability	×	×	×	×
Non-repudiation		×	×	×
Detectability		×	×	×
Information Disclosure		×	×	×
content Unawareness	×			
policy/consent Noncompliance		×	×	×

To map privacy threat types for a specific system, first, a DFD model of the system is created (see 5.7.1.1.1) and then Table 4 is used to determine relevant privacy threats types on the components of the DFD model. The intersections marked with ***x*** in Table 4 are potential privacy threats.

5.7.1.1.3 Identify Misuse Case Scenarios

Once the relevant threat types are mapped to the components of the system (see 5.7.1.1.2), a collection of threat scenarios need to be documented. For this, LINDDUN uses misuse cases.

Table 5 Misuse case template

Structure	***Description***
Summary	Brief description of the threat.
Assets	Asset(s) threatened
Stakeholders	Entity affected
Threats	Potential damage if the misuse case succeeds
Primary misactor	The type of actor performing the misuse case
Basic flow	The flow of steps to execute a successful attack
Alternative flow	Alternative flows of steps to execute a successful attack
Trigger	The condition that initiates the misuse case.
Preconditions	Conditions the system must meet for the attack to be feasible

A misuse case is a use case from the perspective of a misactor. A misactor is the entity who (un)intentionally triggers a misuse case. Table 5 shows the structure and description of a generic misuse case template as proposed by [106].

The purpose of documenting misuse cases is to detail the different scenarios and associated threat types that affect the system. Similar to STRIDE's security threat tree patterns, LINDDUN makes use of threat tree patterns that can be used to detail threat types. Threat tree patterns reflect common attack patterns identified and articulated from existing knowledge

and experience. LINDDUN provides a catalogue of privacy threat tree patterns based on state-of-the-art developments[101] to guide analysts in the elaboration of misuse cases.

5.7.1.1.4 Risk Assessment Techniques

The result of the previous step is a set of misuse cases and detailed privacy threats identified that affect the system. These threats need to be prioritised based on the impact of each specific threat on the system. LINDDUN does not propose or advocates for any specific risk assessment technique. The designer or analyst is free to select one.

5.7.1.1.5 Elicit Privacy Requirements

LINDDUN provides the mapping (see Table 6) from types of privacy threats to types of privacy requirements. Privacy threat types are extracted from the different misuse case scenarios and associated preconditions. As described before, misuse case's preconditions are the conditions that must exist in the system for an attack to be feasible. Table 6 is the complementary mapping to Table 3 and extended with detailed mappings for DFD elements

Table 6 Privacy objectives based on LINDDUN threat types[101]

LINDDUN threats	**Elementary privacy objectives**
Linkability of (*E;E*)	Unlinkability of (*E;E*)
Linkability of (*DF;DF*)	Unlinkability of (*DF;DF*)
Linkability of (*DS;DS*)	Unlinkability of (*DS;DS*)
Linkability of (*P;P*)	Unlinkability of (*P;P*)
Identifiability of (*E;E*)	Anonymity / pseudonymity of (*E;E*)
Identifiability of (*E;DF*)	Anonymity / pseudonymity of (*E;DF*)
Identifiability of (*E;DS*)	Anonymity / pseudonymity of (*E;DS*)
Identifiability of (*E;P*)	Anonymity / pseudonymity of (*E;P*)
Non-repudiation of (*E;DF*)	Plausible deniability of (*E;DF*)
Non-repudiation of (*E;DS*)	Plausible deniability of (*E;DS*)
Non-repudiation of (*E;P*)	Plausible deniability of (*E;P*)
Detectability of *DF*	Undetectability of *DF*
Detectability of *DS*	Undetectability of *DS*
Detectability of *P*	Undetectability of *P*
Information Disclosure of *DF*	Confidentiality of *DF*
Information Disclosure of *DS*	Confidentiality of *DS*
Information Disclosure of *P*	Confidentiality of *P*
Content Unawareness of *E*	Content awareness of *E*
Policy and consent Noncompliance of the system	Policy and consent compliance of the system

5.7.1.1.6 Select Privacy-enhancing Solutions

LINDDUN focuses on preventive and reactive privacy-enhancing technologies (PET) as strategy to counter privacy threats, and provides a comprehensive list of state-of-the-art PETs and their corresponding mapping to privacy requirements. This list is meant to improve guidance to designers and analysts over the solution selection process. Please refer to Table 8 in [101].

The following subsection introduces System Privacy Criteria.

5.7.2 The System Privacy Criteria

The proposed Privacy and Security Requirements Enforcement Framework in Internet-Centric services considers two main general privacy requirements:

Req. 1. To enable individuals to express data protection requirements on their personal data according to the sensitivity of the data in a disclosure request.

Req. 2. To ensure that personal data is actually protected and processed according to the intended purpose of use after being disclosed.

These requirements can be encapsulated into a set of system privacy criteria. The privacy criteria cover different privacy principles as defined by the UK Data Protection Act (DPA) [7] on how personal information must be used by organisations, business and the government. The criteria are defined in Table 7.

Table 7 System Privacy Criteria

	System Privacy Criteria	***DPA Principles. Data should be …***
I.	*Consent (Req. 1)*	• Handled according to people's data protection rights. • Used in a way that is adequate, relevant and not excessive.
II.	*Access and usage control (Req. 2)*	• Used for limited, specifically stated purposes.
III.	*Adequate level of security (Req. 1)*	• Kept safe and secure.

Definitions:

I. ***Consent***: Individuals (Data Provider – DP) should be able to express their data protection requirements on their personal data.

II. ***Access and usage control***: When processing personal data, the target system (Data Controller – DC) must enforce the consent of the DP for the specified purpose(s).

III. ***Adequate level of security***: Personal data must be protected according to sensitivity of the data and the context in which is used.

These criteria are used along with the recruitment data disclosure scenario presented in this chapter (see section 5.3.2) as the starting point to perform the LINDDUN threat analysis methodology and show how specific privacy safeguards can be applied using the proposed Privacy and Security Requirements Enforcement Framework in Internet-Centric services and fulfil the above defined criteria.

[7] https://www.gov.uk/data-protection/the-data-protection-act

5.7.3 Applying the LINDDUN Framework

The LINDDUN methodology is a comprehensive threat analysis framework that can be used to cover all the privacy-related threats of any system. However, the (running) recruitment scenario of this chapter is a complex one and covering all the different types of possible threats and privacy aspects would require an extensive detailed analysis. Similarly, the Privacy and Security Requirements Enforcement Framework in Internet-Centric services is a general privacy and security framework applicable to different data disclosure scenarios that if properly designed and implemented can be suitable to fulfil more privacy requirements than those defined in the System Privacy Criteria depending on the specificities of the target system to be analysed.

Therefore, the threat analysis presented in the remaining subsections of this chapter is intended to demonstrate how the Privacy and Security Requirements Enforcement Framework in Internet-Centric services (sufficiently) fulfils the System Privacy Criteria defined for a (non-exhaustive) set of selected components and elements of the target system.

5.7.3.1.1 Description of the Recruitment Scenario and Data Flow Diagrams (DFD)

Figure 39 scopes and describes the target system to be analysed for the recruitment scenario. The diagram shows the Recruitment Scenario DFD system components to be considered.

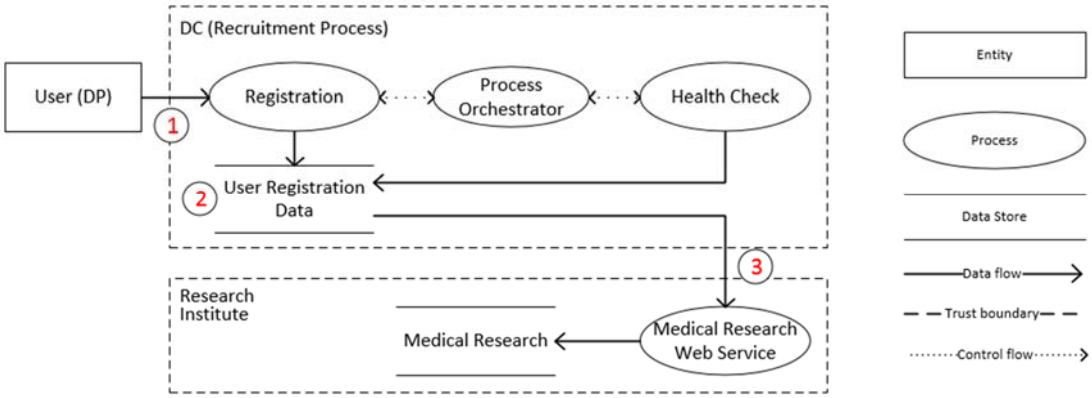


Figure 39 Recruitment Scenario DFD Model

A brief description is as follows. The User (i.e. the Data Provider) discloses a set of personal data to the Recruitment Service (i.e. the Data Controller) after accepting the default privacy policy of the service. The Registration process receives the personal data and store in the User Registration database. The Health Check Service access the database to retrieve information about the user when performing a health check on the user for recruitment purposes. The Health Check service also updates the database with the results of the Health Check. Also, the Medical Research Web service is a third party service that retrieves information about the user

for research purposes. The Medical Research Web Service stores the results in its database (external to the Recruitment process).

5.7.3.1.2 Recruitment Scenario: Mapping Privacy Threats to the DFD elements

Table 8 shows the LINDDUN mapping of privacy threat types to a selected set of elements of Recruitment DFD model. As mentioned before, Table 8 should be applied systematically to all the elements in Figure 39; but, due to lack space, the analysis presented here is scoped. The numbers 1, 2, and 3 (in Table 8), are used to refer to three group of threat types that are considered in this analysis: Group 1 – Content Unawareness and Policy/Consent Non-compliance; Group 2 – Information disclosure; and Group 3 – Identifiability of Data Store, respectively. The same numbers, 1, 2, and 3, correspond to the numbers in Figure 39 to indicate the main element(s) associated to the group of threats considered.

Table 8 Recruitment Scenario: Mapping privacy threat types to DFD element types

Threat categories	E	DF	DS	P
Linkability	×	×	×	×
Identifiability	×	×	**3**	×
Non-repudiation		×	×	×
Detectability		×	×	×
Information Disclosure		×	**2**	×
content Unawareness	**1**			
policy/consent Noncompliance		**1**	**1**	**1**

Privacy Threats identified:

Group 1. *Policy/Consent Non-compliance*: when the DP discloses personal data to the DC, the DP must accept the default privacy policy of the DC. This policy may not be flexible enough to express the real privacy preferences of the DP. Also, it is unknown whether the DC has the infrastructure required to enforce the privacy policy system-wide and beyond its trust boundary for the intended purpose of use. *Content unawareness*: It is no uncommon that the DP (i.e. users) does not read the default privacy policy nor does the DP does know how the sensitivity of the data may change after being processed, for example, after the Health Check service processing.

Group 2. *Information disclosure*: after the Health Check service processing, the Health Check results are stored in the same database as the initial registration data. This database will now contain data with different levels of sensitivity even though it may be accessed by different services for different purposes. It is unknown if the level of security assurance applied to the database is adequate.

Group 3. *Identifiability of Data Store*: The Medical Research Web service access the User Registration database and extract personal data for research purposes. The data extracted may contain sensitive information that may lead to the identification of the user the data is about.

5.7.3.1.3 Recruitment Scenario: Misuse Case Scenarios

In this step, the three privacy threat groups described in the previous section are analysed with respect to relevant existing threat tree patterns provided by the LINDDUN and STRIDE[102] methodologies, in order to produce their corresponding misuse cases and detailed privacy threats.

Figure 40 shows an example of privacy tree patterns. The leaf nodes correspond to the preconditions that must exist for an attack to be feasible. In other words, the leaf nodes are essentially vulnerabilities of the system; that is, errors or weaknesses of the system that an attacker can exploit and lead to a security or privacy failure. A threat is anything that can exploit a vulnerability, e.g. an external attacker. Therefore privacy threat types are detailed by specifying misuse cases where the misactor(s) and vulnerabilities are identified and specified.

5.7.3.1.3.1 Group 1. Content unawareness and Policy / consent noncompliance

Figure 40 shows the privacy tree patterns for the privacy threats *policy/consent non-compliance and content unawareness*.

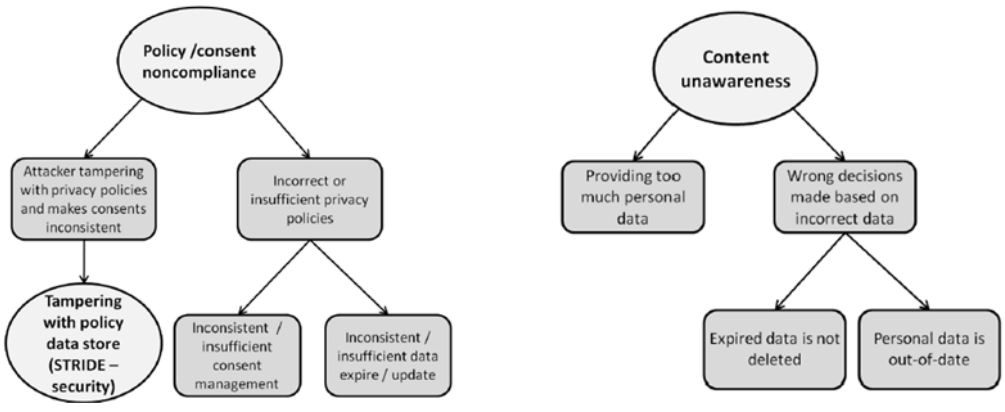


Figure 40 Threat tree patterns: Policy / consent noncompliance (left) and Content unawareness (right) [101]

For the running scenario, the following preconditions are relevant:

Insufficient or inconsistent consent management (Policy / consent noncompliance). This precondition affects: the Registration, Health Check, and Medical Research processes; the User Registration Data Store; and the Data flows between this components. By not having a clearly defined privacy policy it is not possible to control what data flows between process and what data is being processed.

Providing too much personal data (Content unawareness). This precondition affects: the user (i.e. external entity). By not having a clear and usable privacy policy the user is unaware of what data is being disclosed and the specified purpose.

Table 9 shows the corresponding misuse case.

Table 9 Misuse case: Content unawareness, Policy / consent noncompliance

Summary	***Group 1***
Assets	Persona identifiable information (PII)
Stakeholders	The user (DP)
Threats	Content unawareness, Policy / consent noncompliance
Primary misactor	The DC (un)intentionally, the DP unintentionally.
Basic flow	The user discloses data without expressing adequate consent
Alternative flow	*(not considered)
Trigger	Any time the DP discloses information to the DP
Preconditions	Providing too much personal data. Incorrect or insufficient privacy policies.

5.7.3.1.3.2 Group 2. Information disclosure

Figure 41 shows the tree pattern for the *information disclosure* privacy threat. Notice that, differently to the other privacy threats considered, this tree pattern is taken from the STRIDE catalogue since this type of threat is the same for both privacy and security.

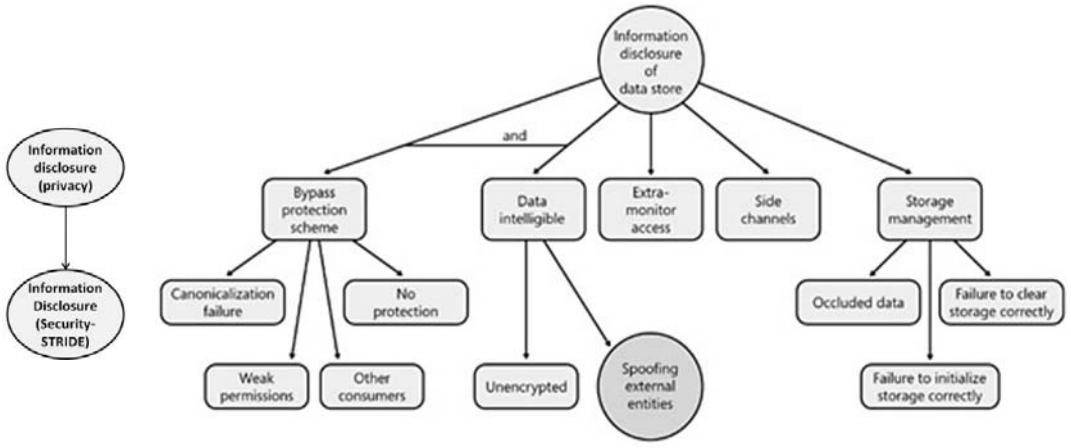


Figure 41 Threat tree pattern: Information disclosure of a data store[102]

For the running scenario, the following preconditions are relevant:

Weak permissions / No protection. This precondition affects: the User Registration Data store. The data store is accessed by the Registration and the Health Check processes (internal to the recruitment process), and by the Medical Research processes (external third party). In addition, the data store contains two types of data categories: health-related and registration-related which may imply a different level of sensitivity.

Unencrypted data. This precondition affects: the User Registration Data store. Considering the data store may contain data with different sensitivity. It is unknown whether highly sensitive date is encrypted. If data is unencrypted, a misactor only would need to bypass the access control mechanism somehow and gain direct access to the data.

Table 10 shows the corresponding misuse case.

Table 10 Misuse case: Information disclosure

Summary	***Group 2***
Assets	Persona identifiable information (PII)
Stakeholders	The user (DP)
Threats	Information disclosure
Primary misactor	Skilled outsider
Basic flow	1. The misactor gains access to the database
Alternative flow	*(not considered)
Trigger	By misactor
Preconditions	Weak permissions / No protection
	Unencrypted data

5.7.3.1.3.3 Group 3. Identifiability of Data Store

Figure 42 shows the privacy tree pattern for the *Identifiability of a data store* privacy threats.

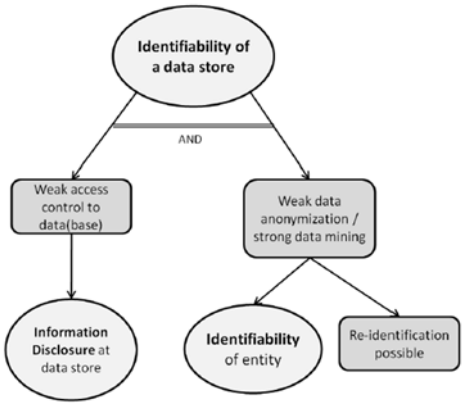


Figure 42 Threat tree pattern: Identifiability of data store[101]

For the running scenario, the following preconditions are relevant:

Weak data anonymisation / strong data mining. The Medical Research Web service fetches health-related data from the User database. If data is not anonymised sufficiently or if the Medical Research Web service perform strong data mining techniques and is able to link the fetched data to data from another sources, the identity of the user can be compromised.

Table 11 shows the corresponding misuse case.

Table 11 Misuse case: Identifiability of Data Store

Summary	***Group 3***
Assets	Persona identifiable information (PII)
Stakeholders	The user (DP)
Threats	Identifiability of Data Store
Primary misactor	Outsider third party DC un(intentionally)
Basic flow	1. The misactor gains access to the database 2. Each data entry is linked to a pseudonym 3. The misactor can link the different pseudonyms together
Alternative flow	*(not considered)
Trigger	By misactor
Preconditions	Weak data anonymisation / strong data mining

5.7.3.1.4 Recruitment Scenario: Risk-based Threats Prioritisation

Threats need to be prioritised based on the impact of each specific threat on the system. Since this analysis only considers a subset of the all the potential privacy threats (i.e. limited to three groups) for a subset of components of the Recruitment scenario system, threats prioritisation is skipped. The three privacy threat groups are analysed.

5.7.3.1.5 Recruitment Scenario: Elicitation of Privacy Requirements

Table 12 lists the Privacy objectives (see Table 6) and Security objectives[102] mapped from the privacy threats analysed (see section 5.7.3.1.2).

Table 12 Privacy objectives (see Table 6) and Security objectives[102]

LINDDUN threats	**Elementary privacy objectives**
Identifiability of (*E;DS*)	Anonymity / pseudonymity of (*E;DS*)
Content Unawareness of *E*	Content awareness of *E*
Policy and consent Noncompliance of the system	Policy and consent compliance of the system
STRIDE threats	**Elementary security objectives**
Information Disclosure of *DS*	Confidentiality of DS
	Access control to DS
	Authentication to DS

Following the LINDDUN methodology, these privacy and security objectives could be used to select from a catalogue of Privacy-enhancing technologies provided by LINDDUN (see section 5.7.1.1.6). However, the purpose of this threat analysis is to demonstrate how the proposed Privacy and Security Requirements Enforcement Framework in Internet-Centric services can be used to fulfil the System Privacy Criteria defined in section 5.7.2.

Instead, in the following and final subsection, an enhanced recruitment DFD model is presented based on the Privacy and Security Requirements Enforcement Framework in Internet-Centric services and it will be described how the enhanced model fulfils the objectives of Table 12 and how these objectives address the System Privacy Criteria.

5.7.3.1.6 Recruitment Scenario: Fulfilment of the System Privacy Criteria

Figure 43 depicts an enhanced recruitment DFD model based on the proposed framework. The components outlined in blue colour correspond to new components added to the initial DFD model.

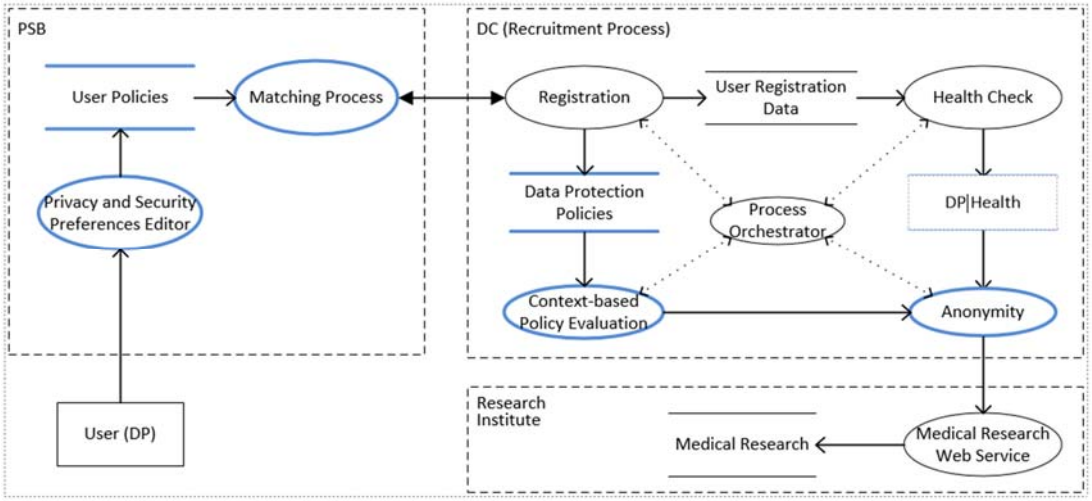


Figure 43 Enhanced recruitment DFD model based on the Privacy and Security Requirements Enforcement Framework in Internet-Centric services

5.7.3.1.6.1 The enhanced DFD model: fulfilment of the System Privacy Criteria and the LINDDUN / STRIDE objectives

Figure 43 shows the introduction of the *Privacy and Security Broker (PSB)* component consisting of 3 DFD elements.

The *Privacy and Security Preferences Editor* allows the DP to define personalised privacy preferences. The *Privacy and Security Preferences Editor* allows the DP to define Data Protection Property Policies (DPPP – see section 5.6). DPPP define context-based security assurance policies. Privacy preferences and DPPP are stored in the *User Policies data store*.

The DP does not need to be present when a data request by the DC is made. The *Matching process* performs a policy matching algorithm and instantiates the DC privacy policy. The DC privacy policy, once instantiated corresponds to a *Business Process Instance* that defines access and usage control rules applicable to the different processes (i.e. activities) within the DC's *trust boundary*.

At the DC side, the *Registration process* receives the *Business Process Instance* and stores it in the *Data Protection Policies* data store. The *Registration process* also receives the *User data* and store it in the *User Registration Data Store*.

The *Health Check process* retrieves user data from the *User Registration data* store, process it, and the result is stored in the *DP Health data store* – a separate data store.

The *Medical Research Web Service* process has access to the *DP Health Data* store through an *Anonymisation process* interface. This interface can be implemented based on the concepts of *forms* (see section 5.5.2) where inputs and outputs for both *DP Health Data store* and *Medical Research Web Service process* are defined by policy.

The *Process Orchestrator* process handles the control flow between the different processes of the system and enforces the evaluation decisions made by the *Context-based Policy Evaluation process*.

The PBS mechanism fulfils the following privacy objective:

I. ***Consent***. Individuals (Data Provider – DP) should be able to express their data protection requirements on their personal data.
   - **Content awareness of *E:*** the PBS remembers the DP's privacy preferences and applies them on the DP's behalf.

II. ***Access and usage control***. When processing personal data, the target system (Data Controller – DC) must enforce the consent of the DP for the specified purpose(s).
   - **Policy and consent compliance of the system**: The *Business Process Instance* contains the DP's preferences (i.e. consent) and is enforced by the Process Orchestrator across the different activities (i.e. DFD processes) in the recruitment scenario.
   - **Anonymity / pseudonymity of (E; DS)**: The anonymisation process ensures the identity of the DP is not revealed when disclosing health-related data.

III. ***Adequate level of security***. Personal data must be protected according to sensitivity of the data and the context in which is used.
   - **Confidentiality, authentication, access control of DS**: Authentication and access control constraints/ requirements are captured by *Business Process Instance*. Confidentiality requirements are captured by the DPPP policies. Moreover, health-related information is now stored in a separate database since it contains data with higher sensitivity than registration data, and security assurance requirements are, therefore, higher.

5.8 Concluding Remarks

This chapter presented a policy-based framework that allows individuals to express data protection requirements in terms privacy and security properties, and to express constraints on how their data propagates once disclosed to the data controller. The proposed policy framework is flexible enough to be used at different levels of abstraction in a business process

to express typical privacy and access control requirements, while providing also a way to express information-flow control constraints.

The concept of a *form* was introduced and it uses aspects of static information-flow analysis to control the data output of activities and to semantically encapsulate it.

Finally, it was taken into account the relationship that exists between category, activity and purpose to characterise personal data and hence define appropriate security measures to be enforced by the data controller.

As further work, one could investigate ways to allow the PSB to calculate data sensitivity in an automated or semi-automated manner.

Chapter 6

Context-Aware Multifactor Authentication Scheme Based On Dynamic Pin

This work proposes an innovative context-aware multi-factor authentication scheme based on a dynamic PIN. The contribution is two-fold. First, an authentication scheme based on graphical passwords where a challenge is dynamically produced based on contextual factors and client device constraints while balancing security assurance and usability. Second, the approach proposed utilises a new methodology for Dynamic PIN-based authentication where not only a new Dynamic PIN is produced in every user authentication attempt, but also the cryptographic transformation used to produce the Dynamic PIN changes dynamically based on the user input, history of authentications, and available authentication factors at the client device.

6.1 Introduction

In the digital space, identity refers to a set of data or identifiers used to describe a human or a digital entity. In the case of humans, managing identity has become an integral and unavoidable part of daily life. Both individuals and organisations routinely use identities for all kind of different purposes, be it social, work, or personal, in order to assert trust and be able to interact. As a result of this situation, identity and authentication-related issues, such as identity theft and fraud, abound and will remain prevalent in the next decades.

User authentication is a means of identifying a user and verifying his identity. There are three main types of methods to achieve user authentication: token-based, biometric-based, and knowledge-based. Token-based methods refer to “what the user has” such as bank cards and

smart cards [107]; biometric-based methods refer to “what the user is” such as iris scans or fingerprints; and knowledge-based methods refer to “what the user knows” such as passwords and PINs. Each type of method has its own characteristics, properties, (dis)advantages, applications, and can be vulnerable to specific types of attacks. Currently, text-based passwords are the most widely used authentication method because of its convenience and usability. However, this type of password is not considered secure enough for certain sensitive or critical transactions and is susceptible to diverse types of attacks including key logging, shoulder-surfing, dictionary attacks, and social engineering, among others.

More recently, graphical-based passwords have been proposed as an alternative to text-based passwords as their characteristics can eliminate or mitigate the abovementioned attacks (if properly designed). Graphical-based passwords require more memory space, and take longer to register and to log-in; but at the same time they are more human-friendly, inexpensive to create, less likely to be written down, have the potential to provide richer symbol space than text passwords[108], and harder to guess. Several studies indicate that graphical-based passwords can be easier to remember [109]. Usability strongly depends on several factors including the expertise and requirements of the target user, the frequency of use, the size and resolution of the screen, and the level of security required[110]. From the security perspective, attacks on graphical-based passwords can be classified as guessing or capture attacks. Guessing attacks include brute-force, dictionary attacks, and attacks on specific graphical password schemes. Capture attacks include malware, phishing, pharming, and social engineering. The same types of attacks apply to text passwords but it has been proved that they are more expensive and require more sophisticated mechanisms to perform them on graphical passwords[110].

One approach to increase the assurance of the authentication process is multi-factor authentication, which consists in combining different authentication methods. For instance, a bank card uses two factors of authentication: a PIN and the card itself used as a token. However, not all transactions require the same level of assurance and choosing the correct authentication factor(s) depends on the nature and criticality of the authentication transaction, its context, and the resources being protected, whether it is the protection of a digital identity itself from abuse or misuse, or controlling access to the usage of restricted information and services, as well as the levels of risk and trust involved. Additionally, as in other areas of security, there are trade-offs among variables such as assurance, performance, and usability that should be balanced for an authentication mechanism to operate optimally.

This work proposes an innovative context-aware multi-factor authentication system based on a dynamic PIN. The contribution of this work is two-fold.

First, traditional password- and PIN-based authentication systems are based on the knowledge of a *fixed* short sequence of digits or characters. At each authentication attempt the user provides always the *same* information. In this way the method is exposed to different attacks which leverage on the *static* feature of the secret. An authentication scheme of the family challenge/response based on graphical-based passwords is presented where a new challenge is dynamically produced based on contextual information (e.g. location), client device constraints, and the risk associated for a given authentication transaction while balancing assurance and usability.

Second, the approach proposed utilises a new methodology for Dynamic PIN-based authentication where not only a new Dynamic PIN is produced in every user authentication attempt but also the cryptographic transformation used to produce the Dynamic PIN changes dynamically, for example, depending on the user input (i.e. response to the challenge), the history of authentication attempts, or authentication factors available on a client device. A new PIN is generated for each authentication attempt without any predictable backward and forward correlation making practically infeasible for a "man-in-the-middle" who manages to intercept content of interactions to predict the next Dynamic PIN given a set of valid Dynamic PIN already used.

The proposed approach leverages on the fact that users commonly use various types of client devices such as smartphones, laptops, tablets, etc., 1) that already incorporate authentication factors (e.g. SIM cards, biometric readers, etc.) that can be integrated in the Dynamic PIN authentication process in order to increase the level of authentication assurance if necessary; 2) that already incorporate sensors and API interfaces that allow obtaining contextual information in order to drive the authentication process; and 3) that provide device-specific information that can be used to optimise the way the user interacts with the device during authentication (e.g. by presenting a customised challenge that takes into account usability and security trade-offs given certain display constraints).

This chapter is organised as follows. Section 6.2 presents the related work. Section 6.3 provides an overview of the proposed authentication mechanism. Sections 6.4, 6.5, and 6.6 present in detail the three main functional phases the authentication system consists of, that is, registration, session key setup, and Dynamic PIN generation, respectively. Section 6.7 describes an innovative use case variant of the proposed authentication scheme. Section 6.8 presents the conclusions.

6.2 Related Work

Graphical-based passwords systems have been extensively studied. Refer to [111] for a detailed survey. However, the proposed system is not concerned with an specific design of graphical-based passwords per se, but instead it considers how the properties of a graphical password, in this case an image-grid challenge, can be adjusted at runtime to balance authentication assurance vs. usability. In [112], a graphical mechanism that handles authentication by means of a numerical PIN that users types on the basis of a secret sequence of objects is presented and the trade-off security vs. usability is analysed. This work is similar to the one proposed here in that it uses a secret sequence of objects as a means to provide variability in the challenge. However, this system does not consider how contextual information can be used to influence the generation of the challenge.

Several frameworks have been proposed that make use of contextual information for user authentication [113, 114]. In [115], the authors introduce the interesting notion of *implicit authentication,* which consists in authenticating users based on behavioural patterns considering time and location. In [116], a context-aware authentication framework is presented as a way of balancing security and usability for authentication by combining a number of passive factors (e.g., a user's location) with appropriate active factors (e.g. tokens) via a probabilistic framework for dynamically selecting an given authentication scheme that satisfies a security requirement. However, the work in [116] is different to the one proposed here, because it does not consider client device constraints at runtime as part of the security vs. usability evaluation process.

Many organisations such as government, military organisations, and banks in particular, use security tokens as a means of two factor authentication. Security tokens can be implemented based on hardware [117] or software [118], and the way they typically work is by displaying a dynamically generated code, also known as one-time-password (OTP) [119], in response to the user entering a PIN number. Then the user uses the OTP to be authenticated (for example to a website). Several security token system have been produced, [120-124], and have as a common feature of a fixed algorithm or function that takes as input some parameter(s), e.g. the PIN entered by the user, and outputs an OTP. The proposed system is effectively a software-based security token that produces an OTP value, i.e. the Dynamic PIN. However, as it will be shown the main difference is in that the cryptographic transformation used to produce the Dynamic PIN changes itself dynamically an in essence this improves the pseudo-randomness of the Dynamic PIN generated. In addition, the cryptographic function changes not only based on the user input in response to the graphical-challenge but also based on

additional available authentication factors, and the history of previous successful authentication attempts.

6.3 Dynamic Pin Overview

The scheme proposed belongs to the family of protocols challenge-response where one party, the authenticator, presents a random challenge and another party, the user, provides a valid response to be authenticated. The challenge consists of a set of objects randomly presented to the user. An object is a *unit* of information with some form of digital representation, it can be a character, a set of characters, an image, etc. In this work image-based challenges are considered.

The system is a client-server distributed application. At the server side, a pseudo-random challenge is generated partly consisting in a pre-shared secret between the server and the user, and based on context and client device constraints; and send it to the client. At the client side, the user responds to the challenge and the response is used as input, along with other additional parameters, to a dynamic pin generation algorithm.

The authentication scheme consists of three functional phases (see Figure 44):

1. **Registration** : the user registers a set of information in order to identify the device and execute the challenge/response protocol
2. **Session key setup** : a lightweight key setup protocol is proposed in order to encrypt the communication channel
3. **Dynamic PIN** : after the environment setup is possible to generate and transmit the new generated Dynamic PIN for the verification

In the following subsections these phases are presented in detail.

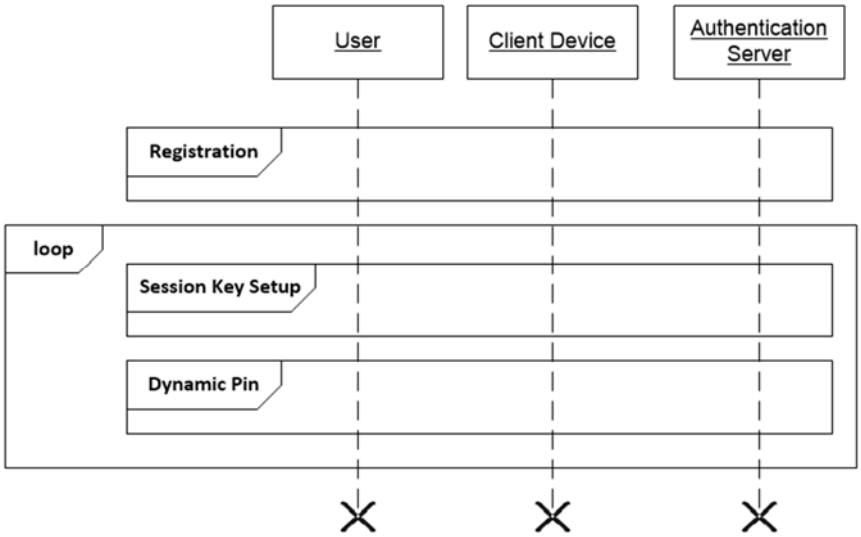


Figure 44 Authentication Scheme Overview

6.4 Registration

During the registration phase the user creates an account at the authentication server and then the server and the user exchange different elements of information. For each device registered, the user specifies *device's authentication factors*, a set of secret images, and *device parameters*.

6.4.1 Registering authentication factors

Authentication factors. The system is primarily based on image-based challenges based on the password(s) the user is going to register (what you know). However, the proposed system also incorporates additional authentication factors that may be available on the client device depending on the level of assurance required during a transaction. For example, the user could register something that the user possesses (what you have) – e.g. a mobile device identified by its International Mobile Station Equipment Identity (IMEI) number, a laptop identified by its CPU-Id number, or a SIM card identified by its International mobile subscriber identity (IMSI) number; or something the user is (what you are) – e.g. a fingerprint.

For each device, the user may register additional authentication factors. For each authentication factor registered, the server creates a record of its name and value, and associates a *secret seed value* generated randomly: ($af_{name_i}, af_{value_i}, af_{seed_i}$), where af_i is an authentication factor. For example, (IMSI, 464989052765867, 4596). Once this is done for all the registered authentication factors, the server then pushes into the device a vector of secret

seed values ($(af_{name_i},\ af_{seed_i}), \ldots, (af_{name_n},\ af_{seed_n})$). The reason for this is to define secure authentication tokens the user/device must reproduce during authentications. The authentication token is computed by,

$$AuthNToken = PBKDF2\Big(concat\big(\big(af_{value_i} + af_{seed_i}\big)\big)\Big) \qquad (6.1)$$

In the proposed system, PBKDF2 [125], a *Key Derivation Function* (part of RSA algorithm), is used to derive a secret key from a partial secret value. PBKDF2 is a key stretching technique used to make the resulting *key* strong against brute force attacks.

And an authentication token vector is defined as

$$AuthNTokenVector = AuthNToken_1, AuthNToken_2, \ldots AuthNToken_m \qquad (6.2)$$

where m is the number of authentication factors registered for a device.

Before transmitting an authentication token during an authentication transaction, the client must recreate the same token calculated at the server side. Notice that at the client side the value af_{value_i} is obtained at runtime; for instance, biometrics data from a thumbprint reader or the IMEI value fetched from a mobile device. It is assumed that the values af_{seed_i} are stored securely on the device.

6.4.2 Registering the image-based password(s)

For each device, the user registers an image-based secret. The user is presented with a selection of image categories, $\mathbb{C} = \{C_1, C_2, C_3, \ldots\}$ and he is asked to select from them. Each category consists of image objects grouped by common characteristics easy to understand, e.g. faces, icons, geometric shapes, etc. Let category C= $(o_1, o_2, o_3, \ldots o_n) \in \mathbb{C}$ be a category set of image objects o_i where $1 < i < n$.

Upon selection of the category, a cryptographic seed is randomly generated for each object o_i. Let $seedsVector = (seed_1, seed_2, seed_3, \ldots seed_n)$ be the vector of seeds and $(o_i, seed_i) \in (C \times seedsVector)$ a relation meaning $seed_i$ corresponds to secret image object o_i. Each $seed_i \in seedsVector$ is of 2 bytes length and represented as 4 hexadecimal digits (0-F).

The user selects a subset of objects in order to form a sequence of images that will constitute his secret password. Let $secretSequence \subset C$ be the sequence of secret image objects o_i with cardinality $|secretSequence|$ selected by the user from C.

For each password registered, the server sends to the client device the $seedsVector$. It is assumed seed vectors are stored securely on the device. Notice that $seedsVector$ contains all the seeds associated to all the image objects for a given category including the subset of seeds associated to the image objects in $secretSequence \subset C$. For $|n| \gg |secretSequence|$ a brute force attack attempting to identify the secret image seeds from the $seedsVector$ can become increasingly difficult as the number of combinations and permutations increases. However, even if an attacker could identify the secret seeds this would not be enough to hack the system as the algorithm depends on additional information elements.

6.4.3 Registering device parameters

Device parameters. This refers to different form factor parameters about the devices the user registers. For instance, type of device – PC, laptop, tablet, smartphone; display size and other form factors; the type of authentication interfaces supported – e.g. biometrics, etc. This type of information is used by the system to determine and specify at runtime an adequate customisation of the image challenge and how it will be displayed, along with any additional authentication factors required for a given authentication transaction. As mentioned before, an optimal authentication should balance trade-off aspects such as security and usability.

6.5 Session Key Setup

The purpose of this phase is to enable the server and the client to establish a secure communication channel. This phase involves a session key generation step and the encryption of the session key with an exchange key. Figure 45 shows the key exchange protocol.

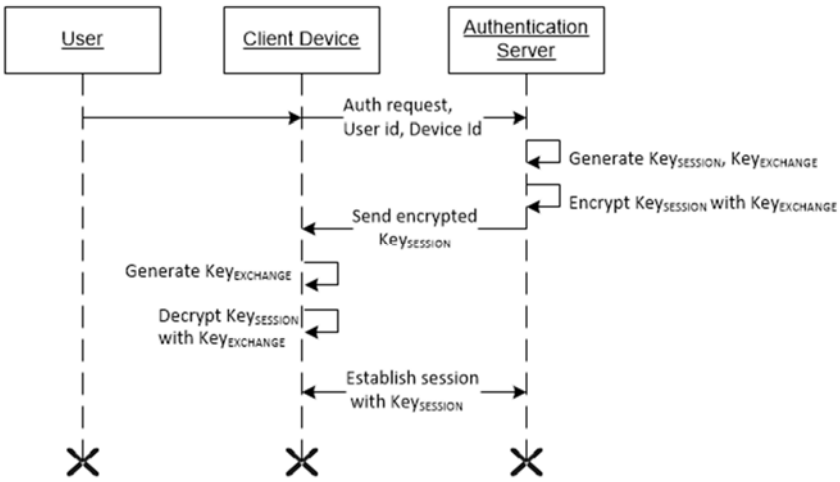


Figure 45 Session Key Exchange Protocol

Session key generation ($Key_{SESSION}$). A session key is a randomly generated symmetric key used for encrypting all messages in one communication session. The session key must be chosen so that it cannot be easily predicted by an attacker, for instance, it can be generated by a cryptographically secure pseudo-random number generator (PRNG). This key needs to be exchanged between server and client in a secure way before it can be used.

Session key exchange ($Key_{EXCHANGE}$). There are several methods for key exchange with different levels of sophistication. For instance, the Diffie-Hellman method allows two parties with no prior knowledge of each other to establish a shared key over an insecure channel. However, in the proposed system since the server and the client already share secret information, the *exchange key* is leveraged on this fact. For an authentication transaction, the *exchange key* is generated in two steps. First, by concatenating all the image seeds in $seedsVector$ and all the authentication tokens $AuthNToken_i$ registered for a given device; and second by applying a key stretching technique on the concatenated string. Here, PBKDF2 is used as an example, but any other key derivation function could be used,

$$Key_{EXCHANGE} = PBKDF2\big(concat(\, seedsVector + AuthNToken_1 + \dots AuthNToken_k)\big) \quad (6.3)$$

Where *k* is the number of authentication tokens.

$Key_{EXCHANGE}$ is generated and used to encrypt and securely transmit the randomly generated $Key_{SESSION}$,

$$Encrypted\ Key_{SESSION} = Encrypt_{Key_{EXCHANGE}}(Key_{SESSION}) \quad (6.4)$$

6.6 Dynamic PIN Generation

Figure 46 describes a high-level view of the steps of the dynamic pin generation process. Once a session key has been exchanged, the Dynamic PIN (DynPIN) is generated involving the following main steps:

1. The server generates a random pin string (RPS) and an image-based challenge. The RPS is used as part of the dynamic pin generation algorithm. The challenge is constructed by combining a subset of image objects taken from the secret sequence and a subset of image objects taken from the image category, and is sent to the client device where the challenge is displayed.
2. The user is asked to recognise and input the combination or sequence of images that represent part of his/her password into the client device.
3. A cryptographic transformation function is computed dynamically based on different variable elements of information that may include the password entered by the user, different authentication factors, and the history of authentication attempts.
4. The cryptographic transformation is then used to generate a one-time dynamic pin.
5. The client device sends the dynamic pin to the server for validation.

One core aspect of the authentication scheme proposed is the fact that the cryptographic transformation used to calculate the one-time dynamic pin changes itself dynamically based on different elements of information.

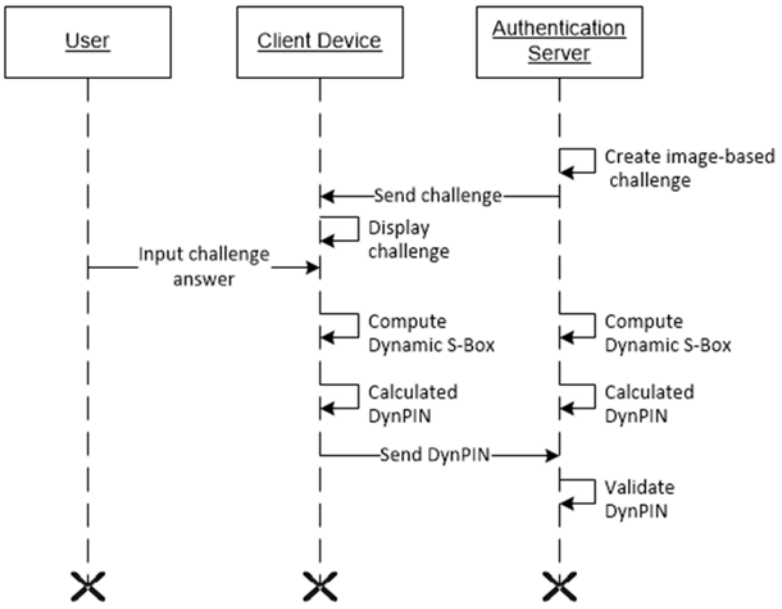


Figure 46 Dynamic PIN generation phase overview

6.6.1 Generation of the Random Pin String and the context-based image-based challenge

In this section, first the random pin string (RPS) and the image-based challenge are defined. Then, it is presented a risk-based rules mechanism used to parameterise and dynamically generate the challenge based on runtime contextual information and client device's constraints.

6.6.1.1 Random pin string (RPS)

The RPS servers as a synchronisation value between the client device and the server during the computation of the dynamic pin. The RPS is created using a pseudo-random number generator (PRNG) function. It is a string of 160 bytes in length.

$$RPS = RPB_1 RPB_2 RPB_3 \dots RPB_{159}, \ \ where\ RPB_i\ is\ a\ byte \tag{6.5}$$

6.6.1.2 Image-based challenge

The image-based challenge is a set of image objects consisting of the union of two subsets randomly generated: 1) $NonSecretImages$ – a subset of images selected from the image category set $C = (o_1, o_2, o_3, .. o_n)$; and 2) $SecretImages$ – a subset of images selected from the $secretSequence \subset C$ that contains the image password objects selected by the user at registration. More formally,

$$SecretImages \subset secretSequence, with\ cardnality\ |SecretImages| = q$$

$$NonSecretImages = C \setminus SecretSequence, with\ cardinality\ |NonSecretIamges| = p$$

Where the relative complement of $secretSequence$ in C is the set of elements in C, but not in $secretSequence$,

$$C \setminus secretSequence = \{o \in C \mid o \notin secretSequence\}$$

Therefore, the image based challenge is the set,

$$ChallengeImages = (SecretImages \cup NonSecretImages), with\ cardinality\ |q + p| \tag{6.6}$$

6.6.1.3 Security strength of the challenge and usability

Figure 47 shows an example of an image challenge represented as a grid of icon images where $q = 5, p = 20,$ and $n = 25$ (the greyed images represent the secret images the user must recognise and select).

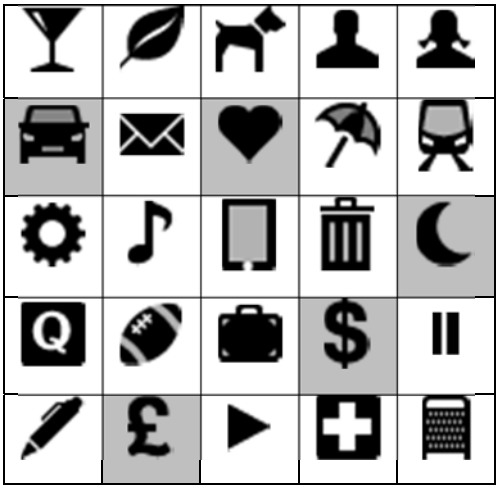


Figure 47 Example of an image challenge. The greyed images represent the secret images.

The security strength of the challenge depends on the values of p and q, non-secret and secret images, respectively; and on the mode in which the user is asked to recognise the secret images. Two challenge modes are defined: ordered and unordered.

Unordered (combination) recognition mode: the user is asked to recognise the set of secret images in any order. The number of possible combinations is given by the equation:

$$\frac{n!}{(n-q)!(q)!}, where\ n = q + p \qquad (6.7)$$

Ordered (permutation) recognition mode: the user is asked to recognise the set of secret images according to the sequence he/she registered initially. In ordered mode the combinatorial equation is:

$$\frac{n!}{(n-q)!}, where\ n = q + p \qquad (6.8)$$

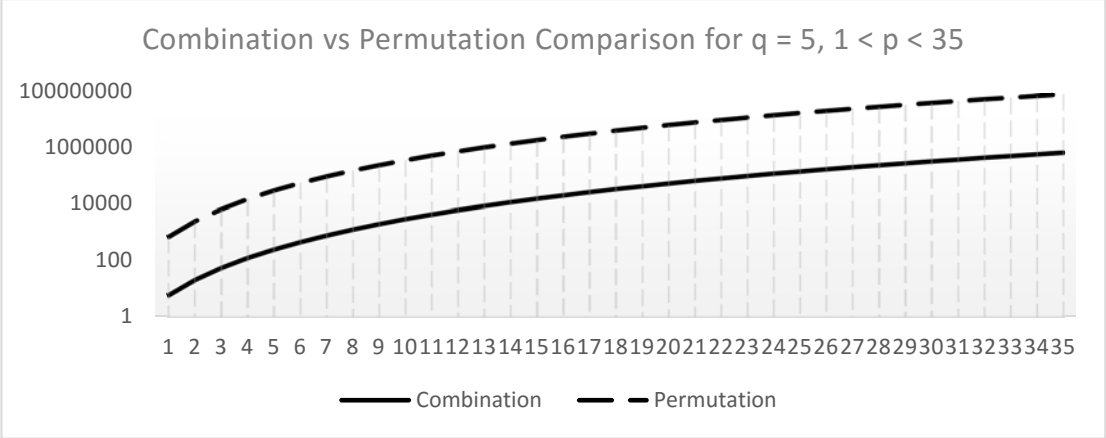


Figure 48 Combination vs. Permutation Functions

Figure 48 illustrates, in logarithmic scale, the speed at which the number of possible combinations (and therefore the security strength) for a challenge with $q = 5$ increases for different values of p ($1 < p < 35$) for the ordered (permutation) and unordered (combination) modes. As illustrated, ordered mode provides higher security over unordered mode for a 5 secret images challenge. However, such increase in security is inversely proportional to the level of usability since it is easier for the user to recognise the 5 secret images in unordered mode without having to recall the exact sequence. Table 13 compares ordered vs. unordered mode for different p and q values.

Table 13 Comparison combination vs. permutation for different p and q

q	***p***	***n***	***Combination mode***	***Permutation mode***
4	20	24	10626	255024
5	20	25	53130	6375600
6	20	26	230230	1.66E+08
4	30	34	46376	1113024
5	30	35	324632	38955840
6	30	36	1947792	1.4E+09
4	40	44	135751	3258024
5	40	45	1221759	1.47E+08
6	40	46	9366819	6.74E+09

6.6.1.4 Context-based challenge generation: usability vs. assurance

The challenge is generated taking into account: client device constraints, contextual factors and risk associated, the level of authentication assurance required, and usability.

Client device constraints. This refers to client device characteristics or properties that constrain some aspect of the authentication process during execution. A challenge constraint

is defined as a tuple $(device\_id, parameter, p)$ where p (see Table 13) is an estimate of the size of the challenge. For instance, a laptop has a larger screen than a smartphone and can display a challenge with a larger number of image objects $n = q + p$. Consider the following two client device constraints:

1. $(laptop_{xyz}, screen_{size} = 14\ inch, 40)$
2. $(smartphone_{abc},\ screen_{size} = 5\ , 20)$

Contextual rules. This refers to contextual factors that carry a level of risk during an authentication transaction. A context rule is defined as a tuple $(context_{parameter_1}, \dots, context_{parameter_x}, risk_{level})$ where $risk_{level}$ is a value between 0 and 1. For example, consider an employee authenticating to a corporate server and the following contextual rules:

3. $(location = WORK, time = 4PM,\ risk_{level} = LOW = 0.2)$
4. $(location = HOME, time = ANY, risk_{level} = MEDIUM = 0.5)$
5. $(location = OTHER, time = ANY, risk_{level} = HIGH = 0.8)$

Assurance level. In the proposed system, the level of assurance depends on the strength of the challenge (see 6.6.1.3) and on the number of additional authentication factors (see 6.4.1) required during an authentication.

Challenge rules. Regarding the strength of the challenge, a challenge rule is defined as a tuple $(q, p, challenge_{mode}, assurance_{level})$. where $assurance_{level}$ is a value between 0 and 1 For example consider the following rules for $p = 20$ (see Table 13):

6. $(4, 20, mode = unordered, assurance_{level} = 0.1)$
7. $(5, 20, mode = unordered, assurance_{level} = 0.3)$
8. $(6, 20, mode = unordered, assurance_{level} = 0.5)$
9. $(4, 20, mode = ordered, assurance_{level} = 0.5)$
10. $(5, 20, mode = ordered, assurance_{level} = 0.7)$
11. $(6, 20, mode = ordered, assurance_{level} = 0.9)$

Notice that rules 8 and 9 are given the same $assurance_{level}$. This is because the number of possible user input combinations for a challenge with the parameter values in these two rules are of similar order of magnitude. As shown in Table 13 rules 8 and 9 would produce 230230 and 255024 possible combinations, respectively.

Generation of the challenge and usability. Based on the applicable client device constraints and applicable contextual rules a level of assurance is determined that mitigates the level of risk present. For instance, if rules (2) and (4) are applicable for a given authentication transaction then any of the challenge rules (8-11) could be used to determine the values q and p to generate the challenge. In such case rule (8) would be the optimal choice since it mitigates the present level of risk and provides the best option in terms of usability, i.e. unordered recognition mode.

Authentication factor rules. Regarding authentication factors, an authentication factor rule is defined as a tuple $\left(number_{authentication_{factors}}, risk_{level}\right)$ and indicates that for a given level of risk a minimum number of authentication factors should be used during authentication. For instance:

12. $\left(authentication_{factors} = 1,\ risk_{level} = LOW = 0.2\right)$
13. $\left(authentication_{factors} = 2,\ risk_{level} = MEDIUM = 0.5\right)$
14. $\left(authentication_{factors} = 3,\ risk_{level} = HIGH = 0.8\right)$

Authentication factor rules are not directly used during the generation of the challenge but they take into account the level of risk associated and are used during the computation of the cryptographic transformation function as it will be shown in section 6.6.4.5.

6.6.2 User response to the challenge

The user responds to the challenge by selecting the secret images from the challenge and entering this information into the client device. The algorithm then retrieves the secret images' seeds and performs an XOR operation over them whose result is a pin of 4 hexadecimal digits (hex_i) in length, hereafter the user pin $UserPIN$.

$$UserPIN = seed_{1'} \oplus seed_{2'} \oplus seed_{3'} \oplus \ldots \oplus seed_{|secretImages|} = hex_1 hex_2 hex_3 hex_4 \quad (6.9)$$

Where each element $seed_{i'}$ corresponds to an element $seed_i \in seedsVector$ (i.e. $seed_{i'} \xrightarrow{f} seed_i$).

The value $UserPIN$ along with the value RPS are taken as input parameters to the cryptographic transformation that calculates the dynamic pin.

6.6.3 Generation of Dynamic PIN ($DynPIN$)

The dynamic pin $DynPIN$ is generated by performing iterative substitutions through the transformation function, $S_{new}(X)$, driven by the values $UserPIN$ and RPS. Recall that:

$UserPIN = hex_1 hex_2 hex_3 hex_4$, where $0 \ll hex_i \ll F$, is 2 bytes long, that is, each hexadecimal digit hex_i is a nibble (half byte), and

$RPS = RPB_1 RPB_2 RPB_3 \ldots RPB_{159}$, is 160 bytes long. Each byte RPB_i is composed by a more significant and a less significant part, $nibble\ {RPB_i}^H\ and\ nibble\ {RPB_i}^L$.

The dynamic pin is defined as a value of 4 byte digits:

$$DynPIN = byte_1 byte_2 byte_3 byte_4 \qquad (6.10)$$

Each $byte_i$ digit is computed as the result of a chain of 7 substitutions between $UserPIN$ (2 bytes long) and 2 bytes of RPS, and 7 iterations through the s-box $S_{new}(X)$. Figure 49 shows the sequence of substitutions between $UserPIN$ and the first two bytes of RPS to produce $byte_1$. Each arrow indicates one iteration through $S_{new}(X)$. During each iteration, $S_{new}(X)$ takes as input one byte consisting of two nibbles: a hexadecimal digit of $UserPIN$ and a nibble of RPB_i; and outputs a new byte, hereafter S_i. The following are the 7 iterations performed to generate $byte_1$:

$S_{new}\left(hex_1, {RPB_0}^H\right) = S_1$

$S_{new}\left({S_1}^H, {RPB_0}^L\right) = S_2$

$S_{new}\left({S_2}^L, hex_2\right) = S_3$

$S_{new}\left({S_3}^H, hex_3\right) = S_4$

$S_{new}\left({S_4}^L, {RPB_1}^H\right) = S_5$

$S_{new}\left({S_5}^H, {RPB_1}^L\right) = S_6$

$S_{new}\left({S_6}^L, hex_4\right) = S_7 = byte_1$

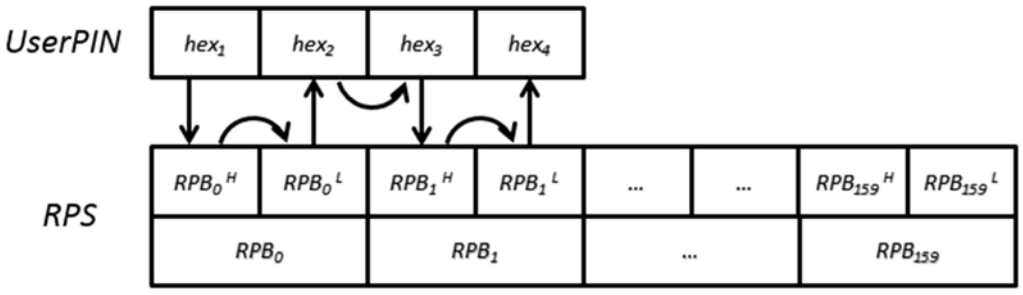


Figure 49 Chain of substitutions and S-Box iterations to generate $byte_1$ of DynPIN

The same process is repeated for the other digits, $byte_2, byte_3, and\ byte_4$, but using a different starting byte of RPS for each digit. To achieve this, the RPS is split into 4 quarters (each one 40 bytes long), and the previous digit ($byte_{i-1}$) of $DynPIN$ viewed as an integer, 0 to 255, and reduced modulo 40, is used to determine a starting byte in RPS. The starting byte in RPS for $byte_i$ is calculated as follows:

$$Z = (40 * j) + Y \qquad (6.11)$$

Where

$$Y = |(byte_{i-1})_{10}|_{40} \qquad (6.12)$$

Z is the starting byte in the RPS used to calculate $byte_2\ (j = 1), byte_3 (j = 2), and\ byte_4 (j = 3)$. Thus the starting byte in the RPS is randomised within each quarter block of the RPS.

Once all the digits are computed, the dynamic pin is sent to the server for validation. The server executes the same algorithm to produce a value that must match the dynamic pin sent by the user for the authentication to be successful.

6.6.4 Computation of the cryptographic transformation function

It is proposed the use of a substitution box, or S-Box, as the transformation function to produce the dynamic pin. A Substitution Box (S-Box) is a component extensively used in cryptosystems to perform substitutions in a way that the relations between the output and the input bits of the S-Box are highly non-linear. This is known as the Shannon's confusion property [126] and ensures a level of protection against linear and differential cryptanalysis.

One of the most well-known S-Boxes specifically designed to be resistant to cryptanalysis attacks is the Rijndael S-Box [127]. It is part of the Advanced Encryption Standard (AES) [128], an industry standard algorithm, selected to replace the Data Encryption Standard (DES) and later Triple DES. The selection decision was made balancing factors including security and computational efficiency.

The security strength of a crypto-algorithm based on substitution boxes can be improved in different ways. For example, by increasing the number of rounds performed by an S-Box, or by changing the S-Box dynamically. In the latter case, Blowfish [129] and Twofish [70] are two well-known examples of this approach and the main advantage is that by dynamically changing the S-Box it becomes more difficult to carry out cryptanalysis attacks since the attacker would not know what S-Box to associate to an S-Box's output for a given session.

In order to increase the pseudo-randomness of the dynamic pin, it is required to use an S-Box that can be obtained dynamically but that at the same time complies with strong security design criteria and crypto-properties. In addition, the S-Box needs to be generated using a deterministic technique, that is, it must be computed based on parameters known to both user's device and the server, and in synchrony.

Barkan et al. [130] show that by replacing the irreducible polynomial and the affine transformation in the Rijndael S-Box it is possible to produce new dual ciphers with the same cryptographic properties of the original S-Box. This result is used to propose an indexing technique that allows selecting a new dual cipher dynamically based on the history of authentication attempts, authentication factors and image object seeds.

In the next subsections the mathematical definitions that support the formulation of the indexing technique are presented along with the proposed indexing function(s).

6.6.4.1 Rijndael S-box[127]

The Rijndael S-box is an algebraic operation that takes in an element of the Galois Field $GF(2^8)$ and outputs another element of $GF(2^8)$, where $GF(2^8)$ is viewed as the finite field $\frac{GF(2)[X]}{(X^8+X^4+X^3+X+1)}$ of polynomials over the finite field $GF(2)$ reduced modulo by the polynomial $X^8+X^4+X^3+X+1$. The operation has 2 steps:

1. Find the multiplicative inverse of the input over $\frac{GF(2)[X]}{(X^8+X^4+X^3+X+1)}$ (0 is sent to 0).

2. Apply the affine transformation $Ax+b$ where x is the result of the first step, (in Rijndael) A is a specific 8×8 matrix with entries in $GF(2)$ and b is a specific vector with 8 entries in $GF(2)$.

The constants were specifically chosen to make it resistant to linear and differential cryptanalysis.

One of the main advantages of the Rijndael S-Box is computational efficiency since elements in the finite field $GF(2^8)$ can be represented as bytes and all transformations can be pre-computed and represented as a lookup matrix. Figure 50 shows the forward Rijndael S-Box as a lookup table of hexadecimal values.

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	xa	xb	xc	xd	xe	xf
0x	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
1x	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
2x	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
3x	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
4x	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
5x	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
6x	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
7x	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
8x	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
9x	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
ax	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
bx	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
cx	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
dx	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
ex	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
fx	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

Figure 50 Forward Rijndael S-Box matrix multiplication

6.6.4.2 Dual Ciphers[130]

Two ciphers E and E' are called Dual Ciphers if they are isomorphic, that is to say there exists three invertible transformations f, g, h such that

$$f\big(E_K(P)\big) = E'_{g(K)}\big(h(P)\big) \quad \forall P, K \qquad (6.13a)$$

where P is the plain text and K is the key. Another formulation of this would be to say:

$$E_K(P) = f\big(E'_{g(K)}\big(h(P)\big)\big) \quad \forall P, K \qquad (6.13b)$$

The benefit of dual ciphers it that a different cipher can be created from an original cipher, but where the new cipher will keep the original's algebraic properties because of the isomorphism.

6.6.4.3 Square Dual Cipher of the Rijndael S-box[130]

If the constants of the Rijndael S-box (denote the Rijndael S-box E) are replaced such that:

- It is replaced A with A^2 where A^2 is not simply the square of the matrix, it is equal to QAQ^{-1} where Q is an 8×8 matrix chosen such that $Qx = x^2$ for all x. As a side result this also means that $QAQ^{-1}x = (Ax)^2$.
- b is replaced with b^2.

Hence it can be shown that these transformations result in a dual cipher (let it be denoted E^2). It can be seen that

$$A^2x + b^2 = QAQ^{-1}x + Qb = Q(A(Q^{-1}x) + b) \qquad (6.14)$$

For an extended discussion on dual ciphers and mathematical proofs of these results refer to [130].

Hence making these transformations (and creating the square dual cipher) is equivalent to applying a pre and post matrix multiplication on the original Rijndael S-box.

This same transformation can be applied to the square dual cipher E^2 to obtain E^4 and similarly for $E^8, E^{16}, E^{32}, E^{64}$ and E^{128} ($E^{256} = E$).

6.6.4.4 Modifying the polynomial of the Rijndael S-box[130]

Recall that the first operation of the Rijndael S-box is to find the multiplicative inverse of the input over $\frac{GF(2)[X]}{(X^8+X^4+X^3+X+1)}$. There are a total of 30 irreducible polynomials of degree 8 over $GF(2^8)$, of which the Rijndael selected $X^8+X^4+X^3+X+1$. As different fields $\frac{GF(2)[X]}{(f(X))}$ for different irreducible polynomials $f(X)$ of the same degree are isomorphic, there exist a linear transformation which can be represented as a binary matrix R such that R takes an element of the Rijndael case and outputs an element of the new case with the changed polynomial. The matrix R is of the form $R = (1,\ a,\ a^2, a^3,\ a^4,\ a^5,\ a^6, a^7)$ where $a^{i}s$ are computed modulo the new irreducible polynomial. Hence R can be used in the same way as Q was used in the previous: applying a pre and post matrix multiplication on the original Rijndael S-box $R(S(R^{-1}x))$.

As there are 30 irreducible polynomials, each of which has the 8 squared ciphers this totals $8 \times 30 = 240$ different dual ciphers. In the book of Rijndaels [131] the 240 dual ciphers of Rijndael are presented including the matrices Rs and the $R^{-1}s$. Here they are used in the following way on the original Rijndael S-box to create a new S-box:

$$S_{new}(X) = R \times SRijndael(R^{-1} \times X) \qquad (6.15)$$

where $SRijndael$ is the original Rijndael S-Box matrix.

6.6.4.5 Indexing the dual ciphers of the Rijndael S-box

In the proposed system, an indexing technique is used for the 240 distinct dual ciphers of Rijndael. The advantages of this are that it is computational efficient and that dual-ciphers have the same crypto-properties of the well-studied AES Rijndael S-box which is widely used and has so far proved to be resistant to cryptanalysis attacks. In this work the number of ciphers is limited to 240 although in [132] the number of possible dual ciphers based on Rijndael has been extended to 9120. As it will be shown in the next section the calculation of the dynamic pin uses the S-box on a random parameter, the RPS, and on the $UserPIN$, to further increases the pseudo-randomness of the proposed approach; hence 240 ciphers are sufficient for the purpose of this work.

More precisely, an indexing function is defined, i.e. $index_{new}$, to determine what dual cipher, out of the 240, to use to generate a new S-Box, i.e. $S_{new}(X)$. The indexing function takes as parameters one or more authentication tokens $AuthNToken_i$, the seeds associated to the set $SecretImages$ of the challenge, the seeds associated to the set $NonSecretImages$ of the challenge, and the index value of the last successful authentication. In addition, the proposed indexing function has two variants depending on whether the user is asked to recognise in order or unordered mode the secret images on the challenge.

Let $DualCipherMatrices = ((R_1, R_1^{-1}), (R_2, R_2^{-1}), \dots, (R_{240}, R_{240}^{-1}))$ be the vector of Dual Ciphers' matrices $(R_{index}, R_{index}^{-1})$ where $0 \leq index < 240$

The two indexing function are defined as follows:

Combination (unordered) recognition mode:

$$index_{new} = (index_{current} + \sum_{i=1}^{l} AuthNToken_i + \sum_{SecretImages} seeds + \sum_{NonSecretImages} seeds) \; mod \; 240 \tag{6.16a}$$

Permutation (ordered) recognition mode:

$$index_{new} = (index_{current} + \sum_{i=1}^{l} AuthNToken_i + \sum_{k=1}^{q=|SecretImages|} k \times seed_k + \sum_{NonSecretImages} seeds)\ mod\ 240 \quad (6.16b)$$

where,

$t_{index_{current}} < t_{index_{new}}, with\ time\ t,$

$SecretImages \subset secretSequence, with\ cardnality\ |SecretImages| = q,$

$NonSecretImages = C \setminus SecretSequence, with\ cardinality\ |NonSecretIamges| = p,$ and

$AuthNToken_i \in AuthNTokenVector =$
$(AuthNToken_1, AuthNToken_2, \ldots AuthNToken_m),\ 1 < l < m.$

Notice that the indexing function in ordered selection mode multiplies each $seed_k$ by the index k forcing the result to depend on the order of the seeds.

Both variants of the indexing function $index_{new}$ output an integer between 0 and 239 that is used to select $(R_{index_{new}}, R^{-1}_{index_{new}}) \in DualCipherMatrices$, and to determine the new S-Box transformation:

$$S_{new}(X) = R_{index_{new}} \times SRijndael(R^{-1}_{index_{new}} \times X) \quad (6.17)$$

$S_{new}(X)$ takes as input a byte and outputs another byte.

6.7 Web-based Dynamic PIN Authentication: Use Case Scenario

The Dynamic PIN Authentication scheme proposed in this chapter is of the family client/server authentication. However, the scheme is flexible enough to be adapted to different use case scenarios. This section describes an innovative use case of online authentication using the proposed scheme.

The Web-based Dynamic PIN Authentication scheme consists of the same three phases described in this chapter: registration, session key setup, and dynamic pin; but with the following architectural and operational variants in the dynamic pin phase design:

- It uses the client device, a smartphone, as a one-time-password token to authenticate to a website.

- During the authentication process the client device and authentication server synchronise via the web browser using a *QR-Code matrix.* Synchronisation is achieved by embedding server and image signatures in a QR-Code matrix and by overlaying the grid of challenge images on top of the QR-Code matrix (see Figure 51).

A QR-Code is a type of machine-readable optical label. The Web-based Dynamic PIN Authentication scheme uses a QR-Code as a means of passing additional hidden information from the Server to the Client Device. The QR-Code operates as a container of information that assists with establishing the authenticity of the challenge and facilitate the dynamic PIN generation process.

Figure 51 Example of QR-Code with overlaid images

The amount of data that can be encoded in a QR-Code depends on the size of the QR-Code (in pixels), the error correction level used during the encoding process – ranges from 7% to 30%. The more error correction the less the amount of data that can be embedded. An image of size 100x100 pixels can be overlaid on a matrix barcode with 30% error correction level and of size 250x250 pixels while allowing the matrix barcode to be read.

The QR-Code technique is detailed in the following section.

6.7.1 Web-based Dynamic PIN Authentication Workflow

The following is the detailed stepwise description of the use case (see Figure 52):

1. The User navigates to the login page of the Website. The web browser sends an authentication request to the Authentication Server.

2. The Server selects a subset of images from the user's password and combines this subset with a subset of images from the larger set that don't correspond to the user's password.

3. The Server creates a grid with these selected images.

4. Server creates a QR code. This QR code has embedded in it a Random PIN String (RPS) of 160 bytes, a signature of the server, and a signature of the images.

5. Server overlays the grid of images on the QR code.
6. Server sends the QR code to the web browser where it is displayed.
7. The user scans the screen with the client device using a QR-Code mobile application. The client device extracts the Random Pin String, the server signature, and the signature of the images. The client device validates the server signature.
8. Based on the signature of images, the Client Device generates a keypad that represent the grid of images displayed on the screen.
9. The User uses the keypad to select the position of the images that are contained in the password.
10. The Client Device computes the Dynamic S-Box and calculates the Dynamic PIN.
11. The User reads the Dynamic PIN and types it on the web page.
12. The Dynamic PIN is sent to the Server for validation.

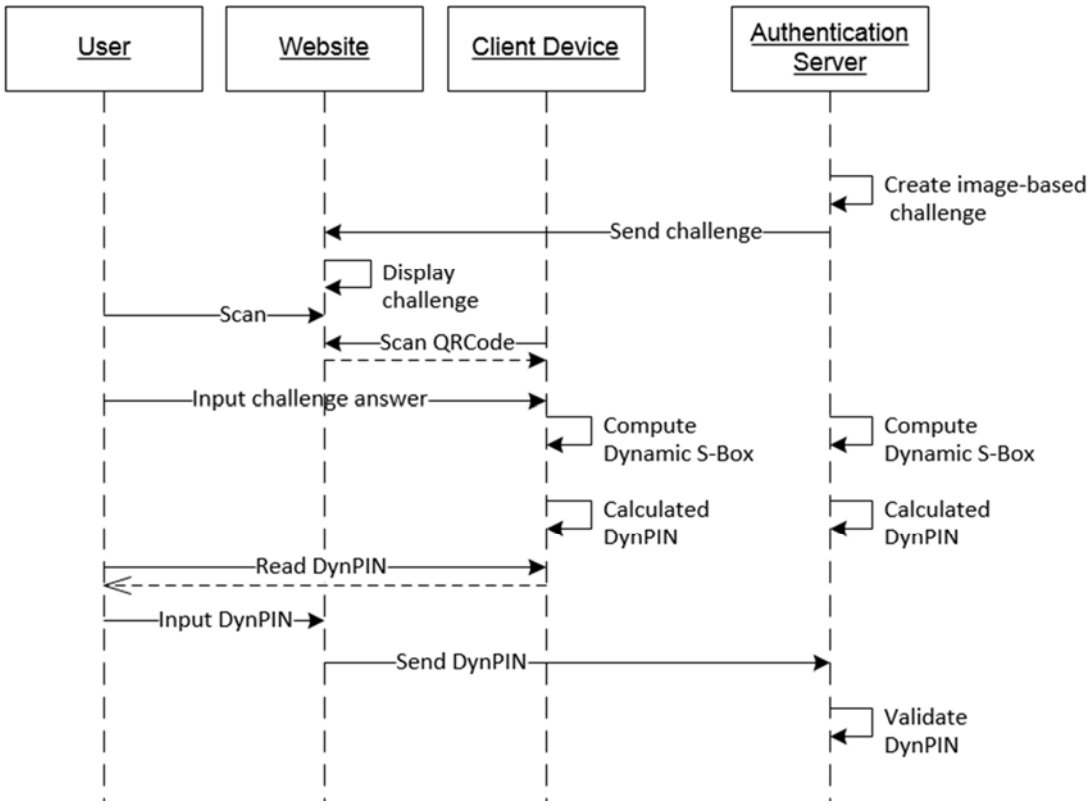


Figure 52 Web-based Dynamic PIN Authentication scheme

6.8 Concluding Remarks

In this chapter it has been presented a context-based multi-factor authentication system based on a dynamic PIN. A novel cryptographic transformation has been proposed that produces a new Dynamic PIN in every user authentication attempt and that changes dynamically based on the user input, history of authentications, and available authentication factors at the client device. The dynamic aspect of the cryptographic transformation increases the pseudo-randomness of the Dynamic PIN and provides strong protection against cryptanalysis attacks.

In addition to this, the authentication scheme is based on graphical passwords where a challenge is dynamically produced based on contextual factors for a specific client device and takes into account the risk associated to the authentication transaction in order to tune an adequate level of assurance while providing the best available usability criteria during the generation of the challenge.

Chapter 7

Policy-Driven Adaptive Protection Systems Methodological Framework

This chapter presents the methodological framework for the realisation of policy-driven adaptive protection systems.

Firstly, the concrete examples of adaptive protection mechanisms developed and presented in Chapter 4, Chapter 5, and Chapter 6 are discussed and analysed (section 7.1) with respect to the concepts on adaptation and policy-driven behaviour presented in the introductory chapters. The works presented in these three chapters served as a foundation and experimental work from which core characteristics, methods, components, and other elements can be analysed in detail towards the investigation of a methodology for the realisation of *policy-driven protection mechanisms* that can be *specialised* and *adapt* to specific *operational environments*.

Secondly, based on the analysis of the common core elements and aspects identified in the adaptive protection mechanisms studied, the discussion is further extended in order to conceptualise, generalise, and describe how these elements and aspects fit in a general methodology (section 7.2). Here, the concept of policy transformation is articulated and derived from this analysis. Also, a general architecture of an adaptation engine that integrates the elements and aspects formulated in the previous sections is presented.

This thesis' proposed framework for the realisation of policy-driven adaptive protection systems is finally presented and incorporated as part of a methodology. The framework consists of two main stages. The development stage (section 7.3) addresses system design and software engineering aspects, and proposes a clearly defined layered architecture. The operational stage (section 7.4) proposes a policy-based management structure of the system, a model of the operational environment, and mechanisms for policy

transformation/refinement. Followed by the presentation of the main steps of the methodology and its general approach (section 7.5).

Section 7.6 presents a final analysis of the system requirements extracted in chapter 3 and how the proposed framework and stepwise guidance for the realisation of policy-driven adaptive protection systems relates and fulfil those requirements.

7.1 Discussion and Analysis of the Protection Mechanisms

This section discusses how protection mechanisms presented in Chapter 4, Chapter 5, and Chapter 6 fit into the investigation on policy-driven adaptive protection mechanisms. For each chapter, an explanatory example is presented followed by an analysis of the specific protection mechanism considering core aspects of this research, that is, translation between abstract and executable policies, execution environment specialisation, policy-driven behaviour, and context-aware adaptation.

7.1.1 Secure Context Execution Enforcement based on Activity Detection: Discussion and Analysis

This section discusses and analyses the secure context execution control framework presented in chapter 4.

7.1.1.1 Explanatory example

Suppose an application PU1 is attempting to access a data file DF1 and consider the following policies:

1. POL_{DF1}: (label=email-apps, write, DF1, {Wi-Fi=on}, allow)
2. POL_{PU1}: (PU1, write, label=contact-list, {location=home}, allow)
3. Host System permissions:
    a. $RES_{PERMISSION}$ (DF1, {contact-list})
    b. $RES_{PERMISSION}$ (PU1, {email-apps, trusted})
4. POL_{HS1}: (PU1, use, Network-Interfaces, {anti-malware=enabled}, allow)

To determine whether to grant PU1 access to DF1 the Policy Manager Module (PMM) (see Figure 22, Chapter 4) executes the following steps:

Policy integration:

Step 1. The PMM retrieves the policies associated to DF1 (1) and PU1 (2) and checks whether it needs to resolve any labels (or categories). In this example, both policies (1) and (2) define labels, that is, email-apps and contact-list, respectively.

Step 2. The PMM retrieves the Host System Permission assignments for DF1 (3a) and PU1 (3b) and tries to resolve for permitted labels in order to parameterise policies (1) and (2). In this case, (1) and (2) become:

5. POL_{DF1}': (PU1, write, DF1, {Wi-Fi=on}, allow)
6. POL_{PU1}': (PU1, write, DF1, {location=home}, allow)

Policy combination:

Step 3. Once all the policies are specified, the PMM retrieves any Host System applicable policies associated to both DF1 and PU1 – in this case policy (4) applies to PU1; and applies the corresponding evaluation rules (see 4.5.5 Table 1) over policies (4), (5) and (6),

eval {(PU1, write, DF1, {Wi-Fi=on}, allow) AND (PU1, write, DF1, {location=home}, allow)

AND (PU1, use, Network-Interfaces, allow)*}*

Step 4. If the result of *eval({})* is false the PMM denies PU1 access to DF1. If the result is true the PMM allows the execution and specifies the following combined policy:

7. *(PU1, access, DF1, {Wi-Fi=on, location=home, anti-malware=enabled}, allow)*

Step 5. The combined policy defines an *active execution context* that evaluates to ALLOW given the conditions defined in it, and that must be enforced by the Policy Enforcement Module (PEM) and monitored by the Secure Context Monitor (SCM) as per Figure 22.

From this example different aspects of the mechanism are discussed in the following subsections.

7.1.1.2 Abstract and executable policies

The policy model distinguishes four types of policies: the individual policies defined on hosted processing units and data files by their administrative entities, i.e. (1) and (2); baseline host system policies, i.e. (3a), (3b), and (4); parameterised policies (5) and (6); and the resulting combined policy (7).

Identifying abstract policies from executable policies depends on how the terms abstract and executable are understood and defined. In the case of the protection mechanism here

discussed, the combined policy (7) is considered as the executable policy because this is the actual policy enforced by the Policy Enforcement Module (PEM); whereas policies (1), (2), (3a), (3b), and (4) are abstract policies as shown in Figure 53. The translation from abstract policies into executable policies is achieved by means of the policy integration (Steps 1 and 2) and policy combination (Steps 3 and 4) mechanisms. Policies (5) and (6) are in between and can be considered as abstract policies but less abstract than (1) and (2).

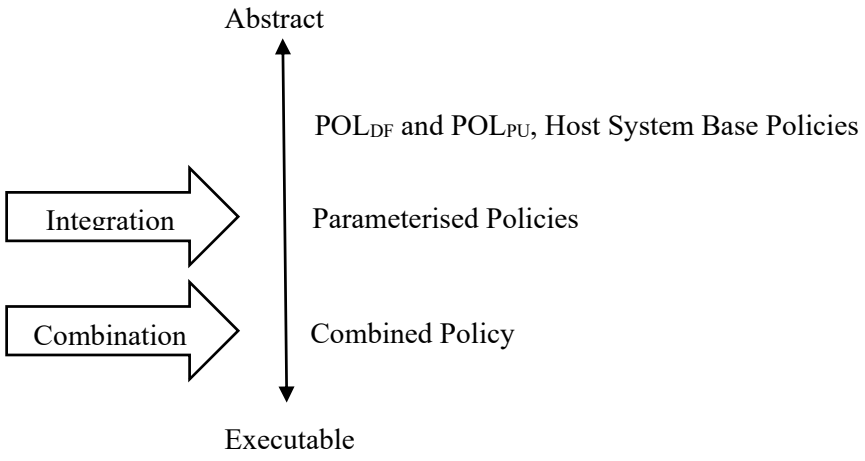


Figure 53 Abstract and executable policies

In fact, defining *abstract* and *executable* is a matter of perspective and strongly depends on how the system is understood. Here, the combined policy (7) is the executable policy as it is the lowest level of abstraction considered in the design of this specific protection mechanism. However, an even lower level of abstraction would consider, for example, the low-level policies enforced at kernel level by the policy enforcement points (PEPs) of the system as the executable policies. In any case, depending on the system to be modelled or designed and the security requirements this must be taken into consideration as policies at different levels provide different levels of control, expressiveness, and granularity.

The running example describes how this system translates from abstract policies into an executable policy taking into account and combining the security requirements of multiple entities. Another aspect to consider is related to how the system delimits the behaviour introduced by processing units and data files' policies.

The host system administrator defines baseline host system policies (4), and *permissions* (3a, 3b) specifying *labels* that characterise and categorise protected resources in order to determine under what conditions users, processing units, and data files can be accessed or used. In other words, baseline host system policies affect the behaviour of the overall system and, in essence, are used as a scoping mechanism to control the allowed behaviour for users, PUs and DFs.

For example, consider again the same policies of the explanatory example, but suppose that at this time only policy (2) is modified and becomes:

8. POL_{PU1}: (PU1, write, label=contact-list, {location=work}, allow)

Notice that the only difference is in that location changed from "home" to "work". This would produce the following combined policy:

9. *(PU1, access, DF1, {Wi-Fi=on, location=**work**, anti-malware=enabled}, allow)*

This is to show that although the policy of PU1 (2) is modified to (8), a new combined policy (9) is generated that is compliant with the baseline policies of the host system, i.e. it is within the allowed scope. In other words, the executable policy changed, thus changing the way the system behaves, while the baseline policies remained unchanged.

7.1.1.3 Protection mechanism and adaptation

Security decisions are made by the proposed architecture, or, more precisely, by the policy combination component (PCC) in the policy management module (see Figure 22) and depend on three elements that change dynamically: host system baseline policies and associated permissions; individual policies defined on hosted processing units and data files; and system state and contextual factors. These three elements may change for different reasons. Any policy issuing entity can add, remove, or modify policies; and also the execution context may change due to activities or events triggered by the entities interacting. As a result, these three elements introduce adaptation to the protection mechanism via the PCC (shown in more detail in Figure 54) that: reacts by generating a new security decision, i.e. the combined policy; requests the PEM to enforce a dynamically generated secure execution context; and requests the Secure Context Monitor (SCM) to monitor that the conditions defined in the combined policy hold true over time.

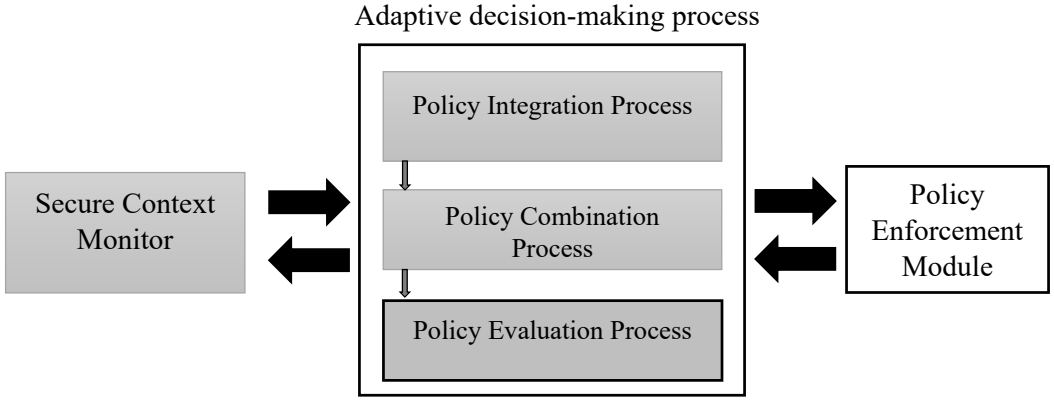


Figure 54 Policy Combination Component and the adaptive decision-making process

It is important to emphasise that the PCC does not have a predetermined way of dealing with detected activities or events. It all depends on the policy integration and combination process, and the interacting parties and protected resources, and their associated policies.

As shown in Figure 54, the PCC can be decomposed into three subcomponents: policy integration process component, policy combination process component, and the policy evaluation process component. Recall from the explanatory example that the policy integration process corresponds to steps 1 and 2, and that the policy combination process corresponds to steps 3 and 4. The final step is the evaluation process, step 5. The decomposition into subcomponents is important to understand how adaptation is introduced into the system.

More specifically, adaptation is introduced at two levels. First, the policy integration and policy combination processes (see Figure 54) have as output a combined policy that is passed to the policy evaluation process to drive the security decisions of the system. This property makes the system adaptive since the combined policy is a runtime dynamically created new policy that did not exist in the system before the involved entities chose to interact. This is equivalent to a policy reconfiguration in the system which creates a totally different behaviour propagating it to the policy enforcement mechanisms. The second level of adaptation is the scoping effect caused by the new policy introduced into the system which is context-based.

Here, the policy combination component (PCC) could be viewed as an adaptive engine consisting of a context-based policy decision point (PDP) extended with two adaptive layers, i.e. policy integration and policy combination. This would be a desirable architectural choice as a way to introduce adaptation into an existing mechanism with a PDP or Policy Engine, such as XACML [76], as one of its components.

7.1.1.4 Specialisation of the protection mechanism

The proposed protection mechanism can be specialised to specific host devices. There are different types of smart devices such as laptops, tablets, smart TVs, game consoles, smartphones, and so on. Each type of device determines a unique execution environment with its own characteristics and properties. For example, a smart device can be characterised by the type of operating system it runs, and the available system resources such as network interfaces, communication protocols, sensor components, kernel, and available APIs. Also, it can be characterised by its form factor, by how it interacts with its user(s) via interfaces, and by important properties such as portability, multi-purpose, multi-function, and multi-user.

Based on these characteristics and properties the secure context execution control framework introduces into the environment appropriate security constraints, in this specific case policy enforcement points, in order to control the execution context and provide isolation for data files and processing units. For instance, a policy enforcement point to block how the user can interact via a display screen is suitable for a smartphone; however, in the case of a game console it would be more appropriate to have an enforcement point at the communication interface between the console and a display connected externally.

More importantly, at the logical level, specialisation is achieved via the policy integration process (see steps 2 and 3 in the running example) since it is during this process that the host system baseline policies resolve labels and permissions and parameterise policies. And in essence, the effect of this process is to logically integrate and delimit what states of execution are allowed or not.

7.1.1.5 Security model: Enhancing adaptation with overrides

Recall from section 4.5.2 that in order to provide flexibility, during policy combination, policy rules were extended with the concept of overrides. The concept of overrides allows processing units to transfer decision authority to other processing units when a certain context-based policy is met. This feature is a form of trust delegation from one policy issuer entity to another. The overrides mechanism extends policy transformation by providing a security model to dynamically support security decisions.

As explained before, the policy combination component takes as input the policies from different administrative entities and produces a combined policy which represents an active secure execution context for the specific execution environment of the host device, and the end result is an overall system whose runtime behaviour is adaptive and dynamic. In essence, the behaviour of the system is based on understanding and incorporating the individual security requirements of all the entities interacting and whose policies reflect their intended actions.

7.1.2 Privacy and Security Requirements Enforcement Framework for Internet-Centric Services: Discussion and Analysis

This section discusses and analyses the Privacy and Security Requirements Enforcement Framework for Internet-Centric Services presented in Chapter 5.

7.1.2.1 Explanatory example

In this section a simplified version of the recruitment process scenario is used as a running example. Figure 55 shows the business process tree where activities are organised hierarchically from abstract to concrete activities. Activity A represents the recruitment process as the whole organisation, activity B represents the administration department of the organisation since it groups all administration-related activities, and so on. Hereafter, "/" is used to represent paths to concrete activities in the tree; for example, "A/C" indicates all concrete activities under "MEDICAL" and "A/C/H" indicates the concrete "RESEARCH" activity.

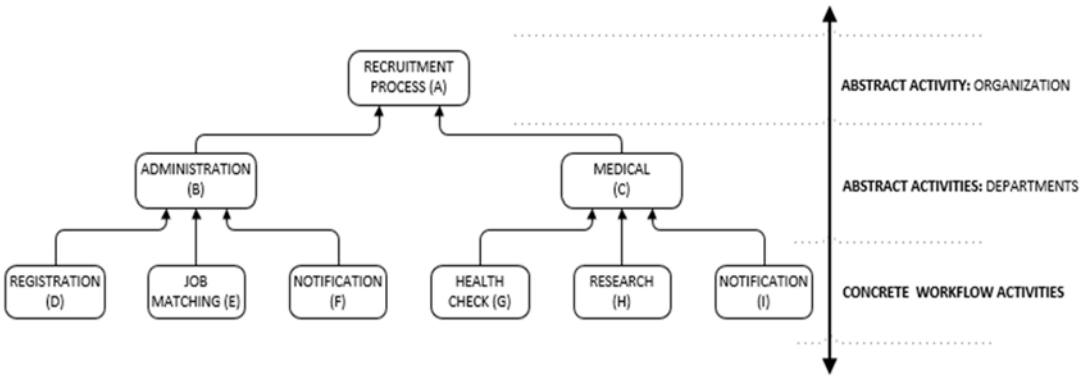


Figure 55 Business Process Tree (simplified recruitment scenario)

Figure 56 shows a graphical representation of the associate business process template (BPT) with control-flows and data-flows (dotted lines). In this example, it represents the intention of the DC to collect the electronic health records of the candidate to perform a health check (A/C/G), produce a medical report, and use it for a job matching activity (A/B/E). The greyed activities indicate optional activities, that is, whether the candidate wants to be notified (A/C/I) about the medical report, and whether he/she agrees for the medical report to be used for medical research (A/C/H).

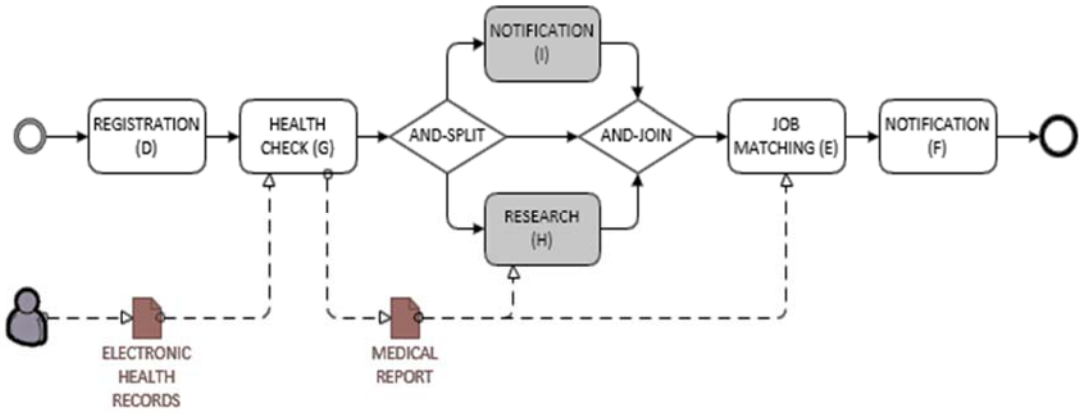


Figure 56 BPMN representation of a business process template (simplified recruitment scenario)

In the system proposed, the Privacy and Security Broker (PSB, see Figure 33) executes two functional phases: 1) the instantiation of business process templates (BPT) and 2) the definition of data protection property policies (DPPP). The following are the steps of phase 1:

Business Template Instantiation Phase

Step 1. The Data Controller (DC) makes a *data disclosure request* to the Data Subject (DS) consisting in the specification of a business process tree and a business process template.

Step 2. The PSB retrieves the *DC policies* from the *data disclosure request*, i.e. rules and policy definitions. Suppose the following are the DC policies for the running example:

1. *Form*: (Medical Report, $\{d_1, d_2\}$)
    a. d_1: (PII, surname, Smith)
    b. d_2: (Medical, Illness, sensitive)
2. Permissions P_i:
    a. P_1: (Medical Report, read)
    b. P_2: (Medical Report, write)
3. Set of *Role-Permissions* $\boldsymbol{RP_{BP}}$:
    a. (Nurse, $\{P_1, P_2\}$, A/C)
    b. (Doctor, $\{P_1, P_2\}$, A/C)
4. The Information-flow Policies $\boldsymbol{Pol_{Flow}}$
    a. (A/C/G, Electronic Health Records, Medical Report)
    b. (A/C/H, Medical Report, -)
    c. (A/B/E, Medical Report, -)

The definition of the Form (1) indicates that the medical report to be produced may contain personal identifiable information (PII) (i.e. surname) and (possibly sensitive) information about illnesses. Permissions (2) are used in role permission assignments (3) and indicate that roles "nurse" and "doctor" are requesting permission to read and write the medical report when executing any medical activity (A/C). The Information-flow policies (4) mean that activity HEALTH CHECK will take as input the electronic health records and will output the medical report (4a); activities RESEARCH and JOB MATCHING will take as input the medical report, i.e. (4b) and (4c), respectively; this is graphically represented in Figure 56.

Step 3. Based on the data disclosure request made the DC, the PSB retrieves applicable DS policies (i.e. user preferences) and executes the instantiation process which consists in a policy matching mechanism between DS policies and DC policies. The result is a *BPT Instance*, i.e. the business process template instantiated with the DS constraints. Suppose the following are the constraints added in the running example:

5. Set of *Role-Permissions* $\boldsymbol{RP_{BP}}'$
    a. (Nurse, {P_1}, A/C/G)
6. The Information-flow Policies $\boldsymbol{Pol_{Flow}}'$
    a. (A/C/H, Medical Report/d_2, -)
7. OPT-OUT activity (A/C/I).

Role-Permission (3a) now becomes (5a) and indicates role "nurse" can only "read" the medical report and only when executing activity HEALTH CHECK. The Information-flow Policies (4b) now becomes (6a) and indicates that activity RESEARCH can have as input the medical report as long as no personal identifiable information is disclosed. Constraint (7) opts out the execution of activity NOTIFICATION (A/C/I). The rest of DC policies remain unaffected.

Step 4. The PSB sends the BPT Instance to the DC to be executed and its constraints enforced by the security infrastructure of the DC. Figure 57 shows the resulting *BPT Instance*.

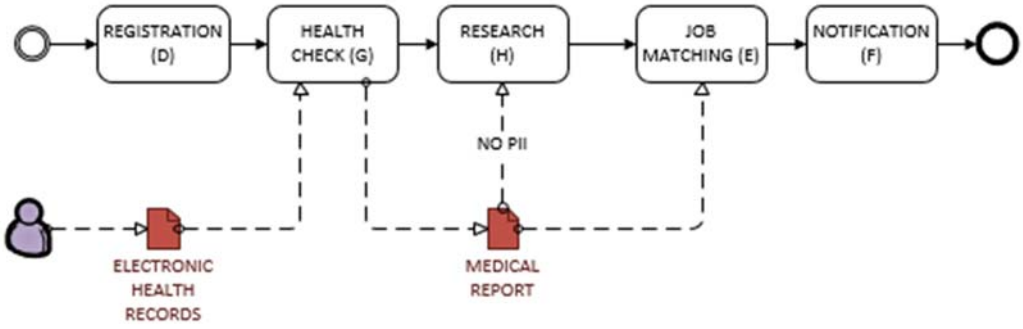


Figure 57 BPT Instance

DPPP Phase

During this phase the PSB determines the level of protection the data to be disclosed requires. More precisely, the PSB uses a labelling mechanism to characterise forms (1) (or its data elements) and associate a level of protection required to be enforced by the activity accessing them as detailed in section 5.6.1. In the running example the BPT Instance allows activity JOB MATCHING to access the medical report. Consider the following associated labels:

Data Label DL_1: (data-type: medical, activity-type: admin, sensitivity-level: high)
Activity Label AL_1: ({security-property: confidentiality, assurance-required: high})

If the PSB assigns DL_1 to the medical report then AL_1 determines the security level required from activity JOB MATCHING if the latter is of type "admin", which happens to be the case (i.e. A/B/E). Therefore, the following DPPP policy is defined:

8. DPPP: (JOB MATHCING, ({security-property: confidentiality, assurance-required: high}))

This policy is to be enforced while executing the BPT Instance. The following subsections elaborate on this running example.

7.1.2.2 Abstract and executable policies

The proposed framework distinguishes between three different types of policies that drive the enforcement behaviour: data subject policies (5-8), data controller policies (1-4), and the BPT Instance.

The business process template (1-4) represents the privacy-aware access control policies of the data controller embedded into the template itself as the template expresses the declaration(s) of the data controller on how it attempts to access and use the information being requested. This is defined via *forms, request permissions and information-flow constraints* attached to the template (see section 5.5.3), and via the *business process tree* which provides the semantics of the business process necessary to understand the context of the *purpose of use* (section 5.4).

The data subject policies consist of two sub-policies: first, the applicable disclosure policies on the data resource(s) requested (5-7), including information-flow and control-flow constraints to be applied on the business process template. And second, data protection property policies (DPPPs) (8) that capture the privacy and security properties required from the data controller for the protection of the requested data.

The BPT Instance is essentially the result of the matching process (steps 1-3) between the data subject's and the data controller's requirements on how data can be used. Moreover, the BPT Instance acts as a scoping mechanism on what can or cannot be done and under what conditions. In other words, the BPT Instance constraints the execution environment.

Policies (1-7) can be considered as abstract policies. The BTP Instance along with the DPPP policy are the executable policies. The BPT instance is the actual business process specification to be executed by an orchestrator at the DC's infrastructure. The DPPP policy is to be incorporated and enforced during execution of the BPT instance.

7.1.2.3 Protection mechanism and adaptation

In the proposed architecture, the Privacy and Security Broker (PSB) executes two functional phases: 1) the instantiation of business process templates and 2) the mediation of disclosure of personal information (via authorisation tokens) to data controllers requesting it. It is important to emphasise the fact these two phases may take place at different points in time. As explained before (see section 5.3.1), during the first phase, constraints are defined and the business process is instantiated but data may not be disclosed yet.

Later in time, during the second phase (execution phase), a labelling mechanism is used to characterise data and associate a level of security assurance required from the activity accessing it. In fact, data is formally characterised by modelling *context* according to three elements: the level of sensitivity associated to data as an intrinsic property, the context in which data is accessed (i.e. the activity and its intended purpose of use), and the category of data (see DPPP phase in the running example and section 5.6). Any of these three elements can dynamically change over time because of several factors (the sensitivity of data depends on several factors: the nature of the data itself, the value given to data by a human, contextual factors, and the purpose of use and its further use). As a result the data protection property policies (DPPP) also change dynamically forcing the context of execution of the business process and the enforcement of protection mechanisms to dynamically adapt in order to provide adequate level of security. In essence, the DPPP policies are context-aware policies.

Figure 58 shows an architectural view of the protection mechanism here discussed. The greyed components function as an adaptation engine that introduces new policies into the infrastructure of the DC. The Policy Matching Process component introduces behaviour in the instantiation phase while the Context-based DPPP component introduces behaviour in the second phase via DPPP policies. As shown in Figure 58, the Data Controller infrastructure behaves or functions as a Policy Engine (i.e. the orchestrator) that consumes and evaluates new incoming policies and as a Policy Enforcement component that enforces them. Here, the resulting business process instantiation combined with the DPPPs drive the behaviour of this protection mechanism.

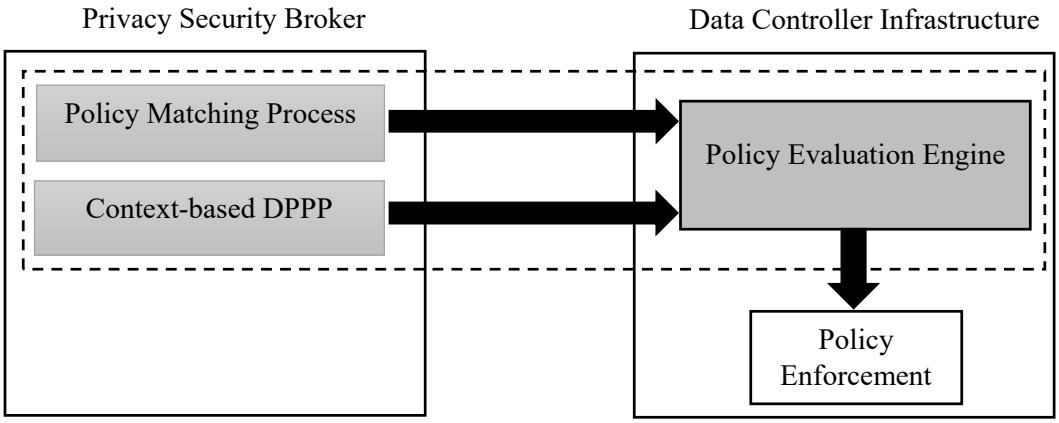


Figure 58 Adaptive decision-making process and PSB

In addition, the sensitivity of data is not only taken into account at disclosure time but throughout the execution. Recall that a BPT Instance is a template with constraints whose execution is performed by an orchestrator. As the execution proceeds an orchestration engine invokes different activities and propagates any applicable execution constraints. As described in section 5.5.2, the input and output of each activity are semantically encapsulated via *forms*. However, the actual information contained in a *form* is not known until runtime and it depends on how the information is processed, whether it is only read, or modified, or aggregated and so on. For example, the form defined in the running example (1) indicates that the medical report may contain information about illnesses; however, whether this type of information will exist in the medical report is not known until the activity HEALTH CHECK produces its report. Therefore, as information flows between activities the sensitivity associated to *forms* may increase/decrease and the system will evaluate DPPPs differently thus adapting and reflecting the appropriate security requirements.

The protection mechanism presented here is a clear example of the separation between the "adaptation component" and the "enforcement component", that together constitute and adaptive enforcement mechanism.

7.1.2.4 Specialisation of the protection mechanism

At implementation level, the BPT instance is input into an orchestrator for execution. Web services are the actual entities that are invoked and implement the activities of the business process and consume and process information. Thus, the execution environment is defined at runtime when the corresponding web services are selected.

Specialisation of the protection mechanism is achieved via the policy matching process which produces the BPT instance. The BPT instance is the result of the data subject and data controller's policies that combined constrain and refine the specification of the business process template and its enforcement (see Figure 57). In other words, the BPT template is a generic model from which the BPT instance is derived, thus generating a complete new workflow model within the system.

7.1.3 Context-Aware Multifactor Authentication Scheme Based on Dynamic PIN: Discussion and Analysis

This section discusses and analyses the Context-Aware Multifactor Authentication Scheme Based on Dynamic PIN presented in Chapter 6.

7.1.3.1 Explanatory example

The example follows the risk-based rules mechanism described in section 6.6.1. The mechanism is used to parameterise and dynamically generate an image-based challenge based on runtime contextual information and client device's constraints.

Consider an authentication transaction where a user chooses to authenticate using a smartphone already registered with the authentication server and consider the following policies:

1. Client device constraint rule:

$$(smartphone_{abc}, \quad screen_{size} = 5\ inch\ , p = 20)$$

2. Contextual rule:

$$(location = HOME, time = ANY, risk_{level} = MEDIUM = 0.5)$$

Rule 1 adds a constraint to the size of the challenge ($p = 20$) given the size of the smartphone's screen; while rule 2 is contextual and associates a given time and location to a level of risk. These rules become applicable within the context of a given transaction.

3. Challenge rules:
    a. $(6, 20, mode = unordered, assurance_{level} = 0.5)$
    b. $(4, 20, mode = ordered, assurance_{level} = 0.5)$
    c. $(5, 20, mode = ordered, assurance_{level} = 0.7)$
4. Authentication factor rule:

a. $\left(authentication_{factors} = 2,\ risk_{level} = MEDIUM = 0.5\right)$

As explained before, the level of assurance depends on the strength of the challenge (see 6.6.1.3) and on the number of additional authentication factors (see 6.4.1) required during an authentication. This corresponds to rules 3 (a, b and c) and 4, respectively. The challenge rules above associate the size of the challenge (i.e. p and q) and mode (i.e. ordered or unordered) to a given level of assurance. For the three rules above the assurance level value was calculated based on the analysis presented in section 6.6.1.3 and the number of combinations and permutations of Table 13 (see the case for rules 3a and 3b which provide same order of magnitude in the number of possible permutations; therefore, they were assigned the same level of assurance 0.5 in this example). The authentication factor rule associates a number of authentication factors (e.g. a pin and a token would be two factors) to a level of risk. Intuitively, number of factors and level of risk an inversely proportional.

The authentication process consists of the following main steps:

Assurance Policy Configuration Process

Step 1. The authentication sever verifies that policies (1) and (2) hold true and searches for applicable challenge rules (3) and authentication factor rules (4). In this case, policies (3a), (3b), and (3c) are applicable since p=20, the same as in policy (1), and the assurance level value is equal to or higher than the level of risk in policy (2). Policy (4) is also applicable since $risk_{level}$ is equal or less in value to that of policy (2).

Step 2. Since there are three applicable challenge rules, the authentication server selects the best one in terms of usability. In this case policy (3a) has the best usability as it defines unordered mode.

Step3. The authentication server defines the *assurance level policy* for the authentication transaction consisting of policies (3a) and (4), or:

5. $(6, 20, mode = unordered, assurance_{level} = 0.5) \wedge \left(authentication_{factors} = 2,\ risk_{level} = MEDIUM = 0.5\right)$

The authentication server generates the challenge based on policy (3a) and send it to the client device along with the RPS, and the IDs of the 2 authentication factors required by policy (4). In this case, suppose one authentication factor is the challenge itself and the other it is the IMEI and IMSI numbers of the smartphone.

Cryptographic Transformation Policy Configuration Process

Step 4. The client device retrieves the authentication token associated to the IMSI/IMEI of the device and retrieves information about the last successful authentication attempt (i.e. $index_{current}$); and uses this information to parameterise the unordered mode *indexing policy*:

6. $index_{new} = (index_{current} + \sum_{i=1}^{l} AuthNToken_i + \sum_{SecretImages} seeds + \sum_{NonSecretImages} seeds)\ mod\ 240$

Policy Evaluation Process

Step 5. The user responds to the challenge, in this case an image-grid, by recognising the secret images. The client device retrieves the required seeds including those associated to the user input and evaluates the indexing policy.

Step 6. Once evaluated, the indexing policy (6) is used to configure the cryptographic transformation function to be used in the system which is given by

$$S_{new}(X) = R_{index_{new}} \times SRijndael(R^{-1}_{index_{new}} \times X)$$

Dynamic PIN generation and validation Process

Step 7. The user input in response to the challenge is transformed into the $UserPIN$ that is used as input along with the RPS value to the cryptographic transformation function $S_{new}(X)$ in order to produce the Dynamic PIN.

Step 8. The Dynamic PIN is sent to the authentication server for validation.

7.1.3.2 Abstract and executable policies

The proposed authentication system distinguishes the following policies: client device constraints rules (1), contextual rules (2), challenge rules (3a, 3b, and 3c), authentication factor rules (4), assurance level policy (5), and the indexing policy (6).

Policies (1), (2), (3a), (3b), (3c), (4), and (5) are abstract policies while policy (6) is executable.

Client device constraint rules (1) and contextual rules (2) are used to select applicable challenge rules (3) and authentication factor rules (4). Client device constraint rules provide parameters about the device that are used to determine the existing constraints that affect the characteristics of the challenge to be constructed and also the type of authentication factors

that may be available. Contextual rules (2) provide runtime information about the context of execution of the authentication transaction, in the running example location and time, and the associated level of risk.

Challenge rules (3) and authentication factor rules (4) take into account the level of risk and reflect the level of authentication assurance required for the given transaction while considering the trade-off usability vs. security. As described in the running example, first, the level of assurance required is determined and applicable policies are selected; and then based on the applicable policies a priority selection approach is used to determine the best rules in terms of usability. Here the result is the selection of policies (3a) and (4).

The assurance level policy ultimately defines how the challenge will be constructed and the authentication factors required to be presented by the client device as part of the authentication process.

The indexing policy (6) is the executable policy since it determines what cryptographic transformation function to use and how to configure it to produce the Dynamic PIN.

7.1.3.3 Protection mechanism and adaptation

In the proposed protection mechanism, security decisions are driven by the different policies in the system. The authentication system consists of four main functional processes that introduce dynamic adaptation: Assurance Policy Configuration Process (steps 1-3), Cryptographic Transformation Policy Configuration Process (step 4), Policy Evaluation (steps 5 and 6), and Dynamic PIN Generation and Validation Process (steps 7-8).

Figure 59 shows how these three processes fit into the overall system. The greyed components indicate the components/processes that induce adaptation into the system.

The Assurance Policy Configuration Process occurs at the authentication server and consists of two sub-processes: Policy Selection Process and Policy Combination Process. Policy Selection process takes as input client device constraint rules (1) and contextual rules (2) and via a matching mechanism determines applicable challenge rules (3) and authentication factor rules (4). As explained before, policy selection takes into account contextual information of the execution and device constraints. The policy combination takes as input the applicable policies and based on risk level and usability combines them to produce a new executable policy, i.e. the assurance policy (5), that reflects the level of assurance required during authentication. The assurance policy introduces adaptation into the system as it defines the characteristics of the challenge produced at runtime. The challenge influences how the user

interacts with the system. It defines how many secret objects the user needs to recognise and the recognition mode, whether it is unordered or ordered. The authentication factors required also influence how the user interacts with the client device or how the client executes actions. For example, the user could be asked to provide a fingerprint as a biometric factor. In the running example, the IMEI/IMSI values force behaviour by the client device in order to provide the adequate token. The cognitive step between user and the client device is not predefined in any way and it entirely depends on the assurance policy generated dynamically at runtime that takes into account the abovementioned factors.

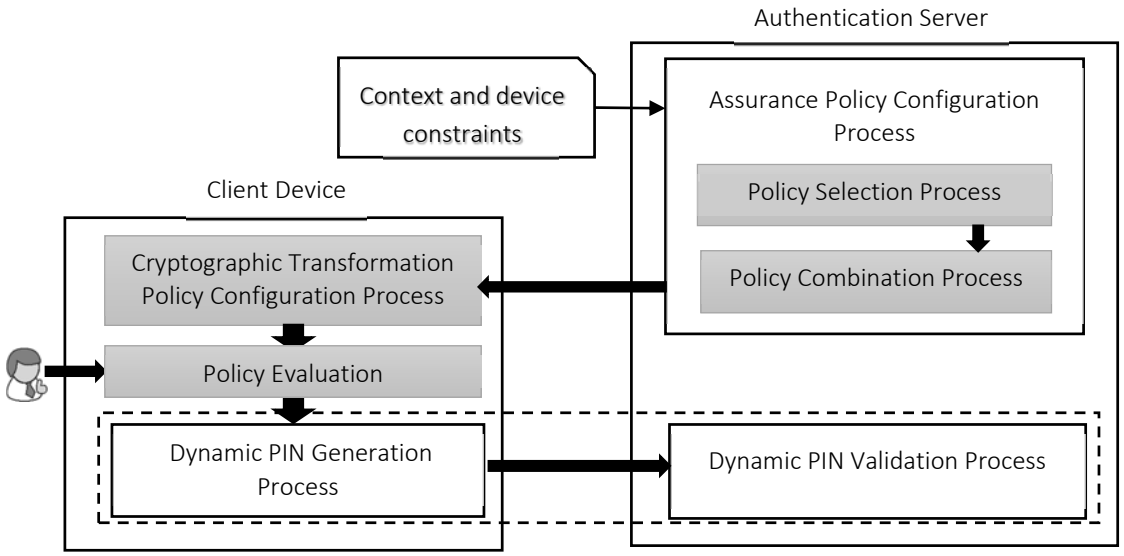


Figure 59 Dynamic PIN Authentication and adaptive decision-making process

The Cryptographic Transformation Policy Configuration Process takes place at the client device (see Figure 59) and consists of two sub-processes: the generation of required authentication tokens and fetching of execution state information such as the last successful authentication attempts, and the parameterisation of the indexing policy (6).

The parameterised indexing policy is the executable policy that is taken as input into the Policy Evaluation module (steps 5 and 6).

During policy evaluation, adaptation is introduced into the system via the indexing policy primarily due to the fact that this function depends on the user input and how he/she interacts with the challenge and the device. More precisely, depending on the user input the indexing policy will define a specific cryptographic transformation to produce the Dynamic PIN. In other words, the user input determines the cryptographic transformation that will reconfigure the system dynamically at runtime. The system will not know what cryptographic transformation to use except until the user provides its input and a parameterised indexing policy is defined.

As shown in Figure 59, the Dynamic PIN Generation and Validation Process is the last functional process and it occurs at both sides: client device and authentication server, that is, Dynamic PIN Generation Process and Dynamic PIN Validation Process, respectively. These processes correspond to the enforcement aspect of the system, which is essentially the enforcement of the indexing policy (6) that if correctly parameterised will cause a valid Dynamic PIN to be produced and verified.

7.1.3.4 Specialisation of the protection mechanism

Specialisation of the authentication system takes place at both sides: authentication server and client device. The challenge is produced based on a specific set of images associated to seeds stored on the client device, the client device constraints, and available authentication factors. All these elements are considered in the usability vs. assurance trade-off aspect and as mentioned before they scope the new behaviour induced into the system. Recall that the challenge is derived from a generic model characterised by the values p and q. Specialisation is achieved by making the values p and q dependent on the client device that the user chooses to use in a given transaction and on the device's unique characteristics. There are parameters specific to a given authentication transaction and device that scope the execution such RPS and $index_{current}$, which are here considered as execution state elements.

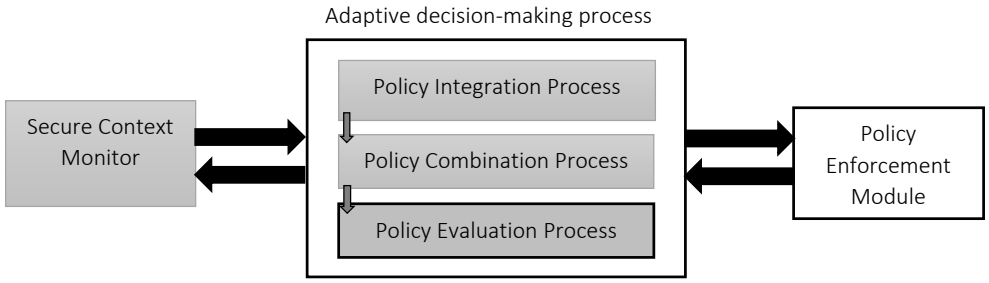
7.1.3.5 Security model: Enhancing adaptation using the indexing policy

Challenges are generated dynamically in order to suit a particular user, a particular device, and contextual information such as, for example, location and time. The combination of all these factors is taken into account in order to produce a challenge that provides the right level of authentication assurance given the implicit levels of risk and trust involved in the particular context of execution.

The indexing policy (6) used to configure the cryptographic transformation function that generates the dynamic PIN is in essence a security model that determines what crypto-function to use (out of the 240 possible options). The indexing policy is a modular component within the policy transformation process that ultimately determines what S-Box to apply. However, a different indexing policy (representing another model) could be used instead. Therefore, the indexing policy is ultimately a model that acts as an extension and supports security decisions within the policy transformation process. Recall that the purpose of the indexing policy is to increase the pseudo-randomness of the authentication scheme.

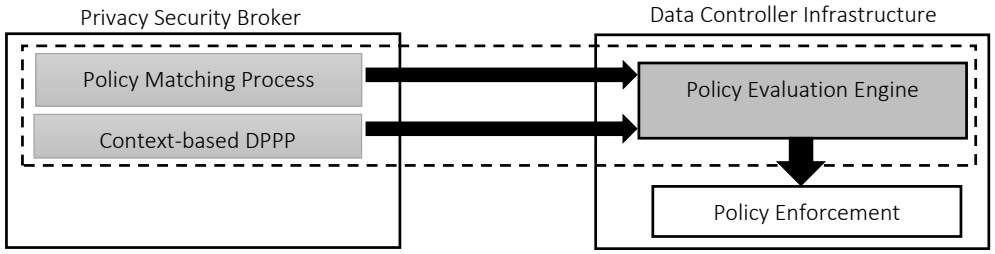
7.2 Towards a general methodology for policy-driven adaptive protection mechanisms

Based on the previous discussion common core elements are highlighted and now the discussion is focused on how to generalise these elements to fit in a general methodology.



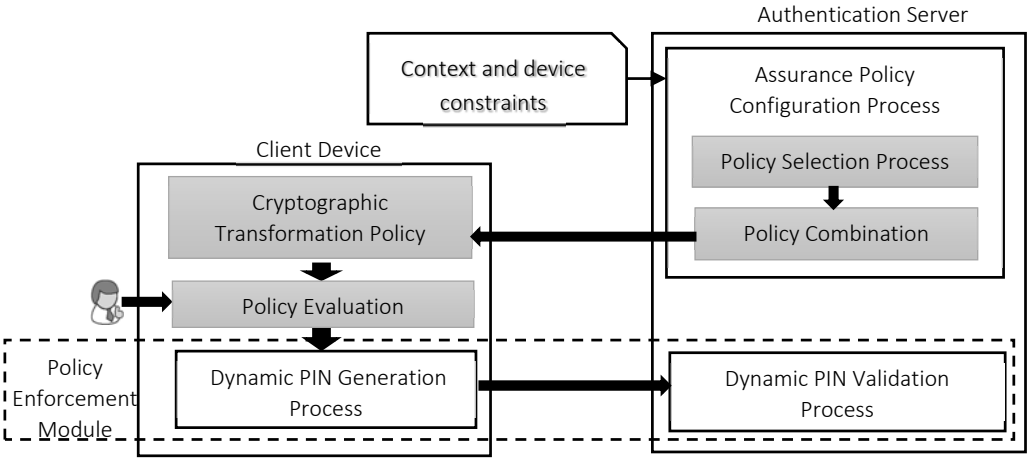
Policy Integration process: via the *Attribute Resolver (AR), Policy Combination Component (PCC)*
Policy Combination process: via the *Policy Resolver (PR)*
Policy Evaluation process: via the *Policy Manager Module (PMM)*
Security Context Monitor: The SCM monitors such execution context and notifies when the conditions change triggering policy re-evaluation (at the PMM)
Policy Enforcement Points: via the Policy Enforcement Module (PEM) performs activity detection and policy enforcement functions

(a) Refer to Figure 22.



Policy Matching process: via *Privacy Security Broker (PSB)* policies are instantiated
Context-based DPPP: via *Data Controller (DC) Workflow Orchestrator* policies are integrated
Policy Evaluation Engine: *via Data Controller (DC) Workflow Orchestrator* are evaluated
Policy Enforcement: via *Data Host (DH)* policies are enforced

(b) Refer to Figure 33



Policy Selection process: via *Authentication Server (AS)* context-based policies are selected based on priority rules
Policy Combination process: via *Authentication Server (AS)* policies are combined into a single assurance policy.
Cryptographic Transformation Policy: via *Client Device (CD)* crypto-transformation is parameterised.
Policy Evaluation process: via *Client Device (CD)* crypto-transformation policy is evaluated.
Dynamic PIN Generation process: via *Client Device (CD)* a dynamic PIN is generated.
Dynamic PIN Validation process: via *Authentication Server (AS)* the dynamic PIN is validated.

(c) Refer to Figure 44

Figure 60 Adaptive decision-making process summary: (a) Secure Context Execution Control Framework, (b) Privacy and Security Requirements Enforcement Framework for Internet-Centric Services, and (c) Context-Aware Multifactor Authentication Scheme Based on Dynamic PIN

Figure 60 provides a summary of the adaptive-decision making process diagrams presented in the previous section for the three protection mechanisms developed: (a) Secure Context Execution Control Framework, (b) Privacy and Security Requirements Enforcement Framework for Internet-Centric Services, and (c) Context-Aware Multifactor Authentication Scheme Based on Dynamic PIN.

The discussion primarily concentrates on the concepts of policy transformation, adaptive behaviour, policy evaluation, and monitoring and detection processes. Monitoring and detection are important aspects that have not been treated so far but they are now briefly introduced.

7.2.1 Policy Transformation, Policy Evaluation, and Adaptive Behaviour

One core aspect of the methodology is to enable translating from abstract policies to executable policies. The first commonality between the protection mechanisms of Chapter 4, Chapter 5, and Chapter 6, hereafter refer to as (a), (b), and (c) – see Figure 60, is that abstract

policies are transformed into executable policies via a process, here, referred to as *Policy Transformation*.

As discussed before, in an adaptive system there are several types of policies or rules with different levels of abstraction that go through a transformation process in order to become executable policies. An executable policy is one that is fully specified and is ready to be deployed into the target system for evaluation.

The Policy Transformation process involves not only different types of policies, but also the use of different types of information in the sense of how information is interpreted and used during this process.

In an adaptive protection mechanism, policies introduce two types of adaptive behaviour: *specialisation* and *context-based adaptation*. *Specialisation refers to how new executable policies are created and then introduced into the system as a result of the behaviour captured (in the operational environment) by the system, with Policy Transformation being the process that creates them*. More specifically, the policy transformation process takes as input a set of abstract policies and dynamically transforms them into fully specified executable policies that are specialised to a specific operational environment and external entities' requirements. *Context-based adaptation refers to how contextual information influences how policies are transformed or evaluated in order to produce an evaluation decision as in the case of executable policies, or in order to influence the Policy Transformation process*.

The protection mechanisms examined in detail in previous chapters share several common aspects respect to *specialisation* and *context-based adaptation* in their policy transformation and policy evaluation phases.

Now, during the policy transformation phase, executable policies are generated by means of two main different sub-processes, here classified as *instantiation processes* and *integration processes*. Instantiation process refers to a process in which constraints defined in an abstract policy are further refined by resolving constraints that need to be parameterised from another policy or set of policies. Integration process refers to a process in which constraints defined in different abstract policies are mixed together, e.g. by means of logical operators, in order to produce a policy with a new set of constraints. In policy transformation, each one of these two types of sub-processes may appear exclusively or complementing to each other as sequential refinement steps. In any case, the constraints instantiation and integration processes ultimately introduce constraints into the execution environment.

Furthermore, how constraints are combined during policy transformation may solely depend on specialisation (i.e. constraints introduced due to behaviour in the operational environment

are specified and refined and they scope behaviour in the system); or may additionally depend on context-based adaptation.

After the policy transformation process, once executable policies are created and passed to the policy evaluation phase, depending on the definition of the executable policies there can be context-based adaptation or not. If the executable policy includes contextual parameters that need to be resolved for evaluation then there is context-based adaptation, otherwise it is said that the evaluation process is fully specified (at that particular runtime point).

These concepts are analysed for the three protection mechanisms.

In Figure 60 (a), in the first step data files and processing unit policies are parameterised with respect to the baseline policies of the system (instantiation process). In a second step the parameterised policies are mixed together in order to produce the combined policy (integration process) based on the combination rules specified in Table 1. In both steps policies are transformed dynamically based on a predetermined set of policies or rules, that is, baseline policies and combination rules. Therefore the policy transformation process introduces only specialisation as the resulting combined policy (i.e. executable policy) specialises the behaviour in the system based on a set of predetermined baseline rules. This combined policy is then passed to the policy evaluation process. Here, the evaluation process introduces context-based adaptation since the combined policy contains contextual parameters that need to be resolved and evaluated dynamically. Notice that abstract policies of data files and processing units are also context-based policies; however, during policy transformation contextual parameters of the policies are not used directly to determine the strategy as to how to transform these policies.

In Figure 60 (b), data subject (DS) and data controller (DC) policies are first instantiated by the Privacy Security Broker (PSB). First, the DC creates a description of a business process that represents constraints on the execution environment at DC's side. Additionally, the business process description is annotated with context-based constraints of the DC (recall that here context is modelled based on the type of activity, data sensitivity, and data category) in order to produce a business process template. These constraints represent the DC policy. Then at the PSB side, DS policies are added via a matching process that instantiates and further constraints the business process template and produces a business process instance (instantiation process). Notice that PSB understands the DS policies as contextual constraints of the data request and therefore the matching process depends on this type of information (context-based adaptation). Later on during the execution phase at the DC's side, a further set of context-based constraints, i.e. DPPP, are mixed with the business process instance (integration process – by a simple logical AND rule) before policy evaluation by a workflow

orchestrator engine. Here, the instantiation process is much more complex than that of (a) – i.e. it depends on the user's preferences known to the PSB as opposed to a set of predetermined rules specified in Table 1 for the protection mechanism (a); whereas the integration process is very basic – i.e. in Chapter 5, it was implicitly assumed an AND operation since it was said that the orchestrator takes as input the business process instance and adds the DPPP policies during execution.

In Figure 60 (c), during the first phase, client device rules and contextual rules are taken as input into a policy selection process that takes into account device parameters (i.e. execution constraints that introduce specialisation) and the level of risk (i.e. contextual constraints that introduce context-based adaptation) in an authentication transaction and maps to a set of applicable rules, i.e. challenge rules and authentication factor rules. Then, the selection process is further refined via a priority policy selection procedure that measures the trade-off security vs. usability and determines the optimal challenge rule and authentication factor rule that are then combined to generate the assurance policy of the system. The policy selection process is effectively an integration process that combines execution and contextual constraints. During the second phase the assurance policy is used to select an indexing policy (either ordered or unordered mode policy) and to parameterise it according the appropriate available authentication factors (instantiation process) and history of successful authentications. The resulting parameterised indexing policy is then introduced into the policy evaluation module. The parameterised indexing policy is context-based since during evaluation it depends on the user input to the challenge (context-based adaptation).

Table 14 summarises the above analysis on types of adaptive behaviour in the Policy Transformation and Policy Evaluation phases for the protection mechanisms (a), (b), and (c).

Table 14 Policy Transformation and Policy Evaluation adaptation types

		Type Of Policy Transformation or Evaluation	**Based On**	**Type Of Adaptive Behaviour**
Secure Context Execution Control Framework	***Policy transformation phase***	1) Constraints instantiation	Baseline policies	Specialisation
		2) Constraints integration	Combination rules	Specialisation
	Policy evaluation phase	Context-based.	Combined policy	Context-based adaptation
Privacy And Security Requirements Enforcement Framework For Internet-Centric Services	***Policy transformation phase***	1) Constraints instantiation	Business process template	Context-based adaptation and Specialisation
		2) Constraints integration	Simple-AND rule	Specialisation
	Policy evaluation phase	Context-based	Business process instance + DPPP	Context-based adaptation

Context-Aware Multifactor Authentication Scheme Based On Dynamic Pin	***Policy transformation phase***	1) Constraints integration	Different rules	Specialisation and context-based adaptation
		2) Constraints instantiation	Authentication factors, $index_{CURRENT}$	Specialisation
	Policy evaluation phase	Policy fully specified (at this point)	Indexing policy	Context-based adaptation

7.2.2 Monitoring Process

This process involves monitoring relevant information and events internal to the execution environment (self-awareness) as well as external information and events sourced from the operational environment (context-awareness). External information and events relate to context-related aspects and behaviour captured, but out of the execution control of the system. Internal information and events relate to the execution state and other context-related aspects within its control of execution. The execution state changes according to the behaviour of different entities interacting and performing actions in the operational environment. Context-related aspects change over time and influence the policy transformation and policy evaluation processes, that is, how policies are refined or evaluated. Information or events related to these two aspects, execution state and context-related, can manifest exclusive to each other, but also in some situations, the execution state can be used as context, i.e. they overlap. In other words, execution environment and operational environment may overlap.

Consider the following example of protection mechanism (a). Suppose there are three entities on the host device: two processing units PU_1 and PU_2, and a data file DF_1. PU_1 is accessing data file DF_1 and PU_1 has a policy that says "if PU_1 is accessing DF_1 then PU_2 should not run". Now suppose the user attempts to launch PU_2.

In this example, first the monitoring process captures when PU_1 executes the access action on DF_1. This is a form of execution state monitoring. Notice that this is not triggered by any contextual policy. Then, when the user attempts to launch PU_2 the monitoring process captures this event and triggers policy re-evaluation since the policy of PU_1 is dependent on PU_2's execution state. This is an example where execution state information becomes contextual information used for the evaluation process. An example of only context-based information could be a role attribute value queried to the monitoring process in order to evaluate an access control policy. Similar examples can easily be described for protection mechanisms (b) and (c).

The previous simple example also illustrates another important aspect of the monitoring process that is concerned with how it is used in the system by other processes. It is used as a

query-based component to resolve parameters and also as an event-based component to triggers signals to the policy transformation and evaluation process.

Finally, the distinction between state of execution and context in the monitoring process is important as it highlights how information or events are understood and used to drive specialisation and context-based adaptation in protection mechanisms. In addition, this distinction allows reasoning about how to model context and execution state and consider the possible implications whether the two aspects are isolated or co-exist.

7.2.3 Detection Process

This process has to do with determining what useful and relevant information can be extracted, mined, or inferred from the monitoring process that can influence the behaviour of the system. The detection process is used to detect from the context and the execution state when a change is required. More specifically, the detection process adds another element of adaptation and provides feedback to the policy transformation process consisting of (possibly newly discovered) information or events on the system.

As an implementation example consider an intrusion detection system (IDS) that detects abnormal behaviour and triggers certain actions as a countermeasure to eliminate or mitigate risk in the system. Another example is to consider the risk-based contextual rules such as those of the protection mechanism (c) – see section 7.1.3.1, where the level of risk could be dynamically changed not only based on external contextual parameters but also, for example, based on other detected behaviour or a more sophisticated risk model. This would directly impact on how the image-based challenge would be created.

In the protection mechanism (a) the detection process is explicitly considered and it is based on the "activity detector" and "policy enforcement points" components in the proposed architecture (see Figure 22, 4.6).

7.2.4 Architectural Overview of the Policy-driven Adaptive Protection Mechanism

Figure 61 provides an architectural view of the adaptation engine and links between the different components discussed in the previous (sub) sections.

The execution environment is represented as entities that perform or intend to perform actions on other entities. It is represented in such a way that interactions can be encapsulated in the form of policies, e.g. {subject, action, and object}. The adaptation engine consists in a monitoring module, detection module, policy transformation module, and policy evaluation

module. The adaptation engine clearly highlights the separation of concerns between the policy transformation phase and the policy evaluation phase. As depicted, the policy transformation phase is supported by both the monitoring and the detection modules since the specialisation processes requires, first, to potentially resolve context-based queries or events via the monitoring module, and second, to determine how to execute policy transformation via the detection module. The detection module processes events (i.e. behaviour in the system) and triggers the applicable transformation strategy towards the policy transformation module. For example, for the protection mechanism (a) such strategy is based on the policy combination rules of Table 1.

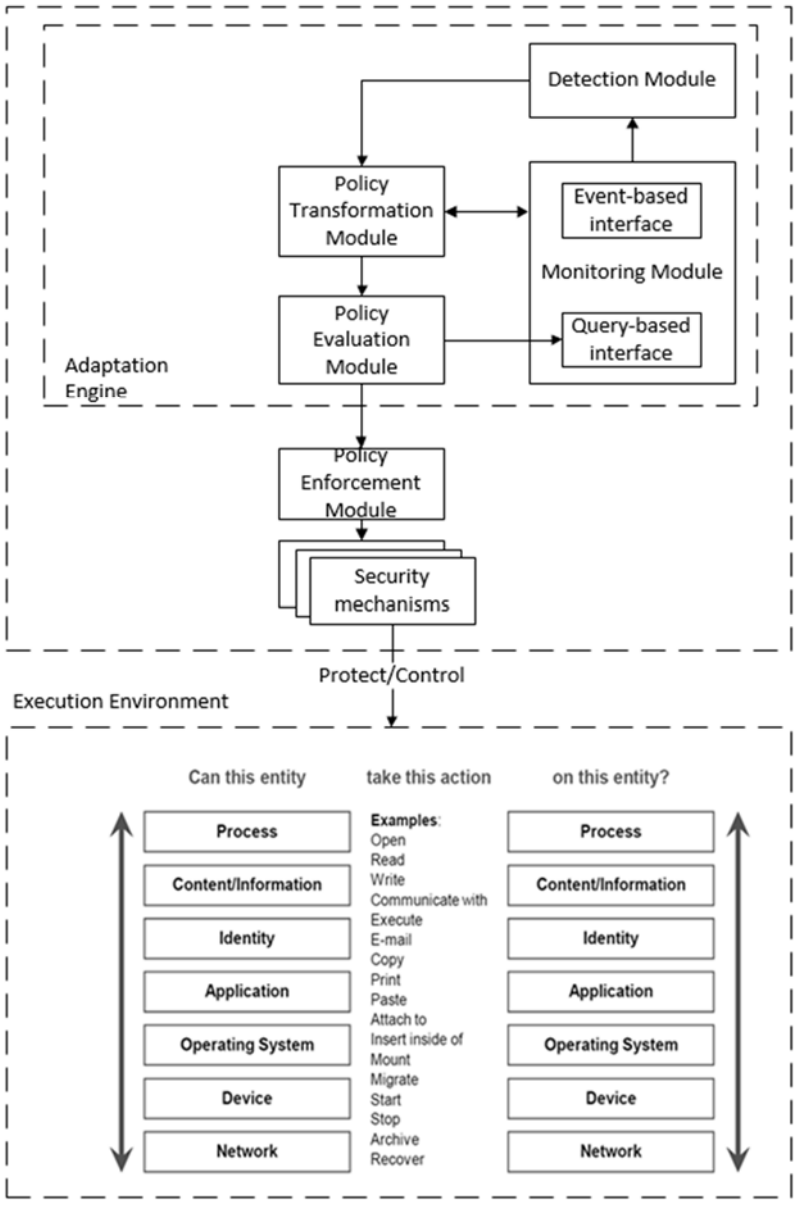


Figure 61 Architectural Overview of the Policy-driven Adaptive Protection Mechanism with a policy transformation module

This is the proposed general architecture. It conceptualises and describes the elements and aspects covered in the previous sections, and provides the functional components required to enable specialisation. The next two sections describe how the general architecture can be realised and how it fit into a general methodology. The proposed methodological framework for realising policy-based adaptive systems consisting of two stages: development and operational.

7.3 Protection System Development Stage

The development stage provides the required abstractions, mappings, and all other required elements in a layered architecture that allows specifying the system. The operational stage is concerned with how to manage such elements.

7.3.1 Software Engineering Perspective

During the development of the protection system three phases are considered: software security requirements, design and specification, and concrete-level software implementation. See Figure 62.

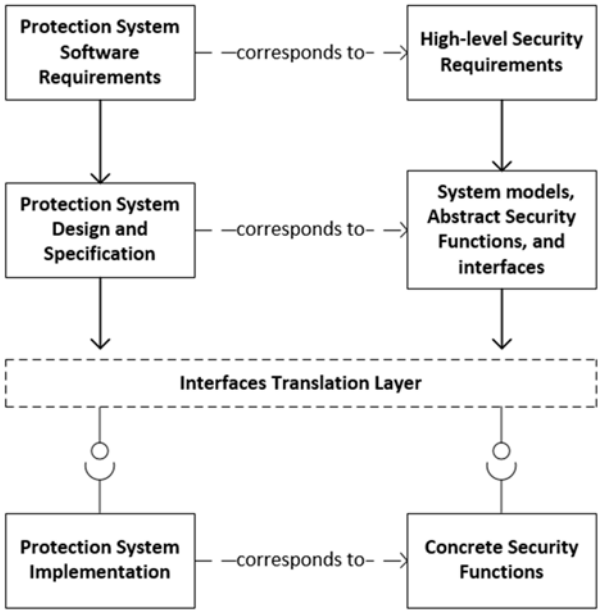


Figure 62 Development phase

7.3.1.1 System Security Requirements

In the first phase, high-level security goals are identified and decomposed into sub-goals. Security requirements are elucidated, analysed, and specified from stakeholders' high-level security goals.

7.3.1.2 System Design and specification

The methodology follows a model-driven approach. Based on the high-level security requirements the system is designed. Normally, the design consists of different system models specifying different functions and sub-functions that address the security requirement. The following elements are modelled:

- *Abstract Security Functions.* These functions are referred to as abstract security functions because they are not implemented at a technology-specific concrete level but, instead, they are corresponding modelled abstractions of high-level security requirements. For example, an abstract security function could be seen as an abstract class in object-oriented programming.
- *System models* capture the structural and behavioural aspects of *abstract security functions*. The idea is to exploit available associations and dependencies among functions to reason about them. For instance, associations may include composition, inheritance, aggregation, and so on, or structural dependencies such as concatenation, parallelism, selection, and repetition.
- Abstract Security Functions' *Interfaces.* Interfaces are of two types, provided and required. *Provided interfaces* specify the type of output offered by abstract security functions to other components at the concrete level (i.e. control aspects). *Required interfaces* specify the type of input needed by abstract security functions to execute its operations with the input coming from components at the concrete level (i.e. monitoring/detection aspects).

Abstract security functions and their corresponding interfaces should be designed using (ideally) standardised modelling languages, e.g. UML or Model-Driven Architecture (MDA). As shown in Figure 62, there is an *interfaces translation layer* that is responsible for the conversion and mapping of the interfaces defined by the abstract security functions in terms of a modelling language into implementation-specific (concrete-level) interfaces for different technologies, here referred to as *concrete interfaces*.

7.3.1.3 (Concrete-level) Software Implementation

During this phase, the concrete-level functions of the system are either implemented from scratch or integrated in the case of legacy systems, based on the design and specification

models provided by the previous phase. The implementation can be done in more than one way. Abstract security functions and their associations, dependencies, and interfaces are implemented as low-level concrete software components. At the concrete level, functions and interfaces are referred to as *concrete security functions* and *concrete interfaces*. From the implementation phase perspective, concrete interfaces are technology-specific *interfaces*.

The inclusion of the *interfaces translation layer* in the methodology is a design choice that allows flexibility in the use of implementation technologies, supports integration with legacy implementations, the separation of concerns between modelling and implementation tasks, and extensibility properties.

7.3.2 Layered Architectural Perspective

The software engineering perspective (previous section) provides two clear mappings. As shown in Figure 62, there is a mapping from high-level security requirement to an abstract security function (or a set of them, e.g. a composition), and there is a second mapping from abstract security functions to concrete security functions that fulfil the same corresponding high-level security requirement(s).

In addition, the phases described in the software engineering perspective (security requirements, design and specification, and concrete-level implementation) and their corresponding primary outcomes (high-level security requirements, Abstract Security Functions and Concrete Security Functions), respectively; allow to establish logical and architectural separation of concerns between phases. These concerns are encapsulated into three different layers of abstraction as shown in Figure 63.

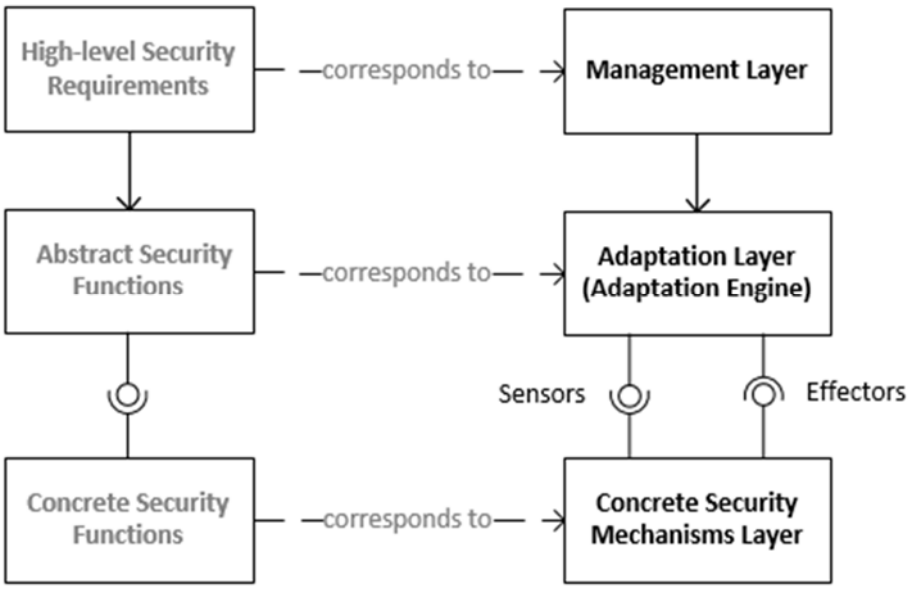


Figure 63 Architectural Design Perspective

7.3.2.1 Management Layer

At the highest level, software requirements engineering is concerned with the elucidation of high-level security requirements. The management layer corresponds to the highest level of abstraction in the system and its purpose is to provide adequate abstractions for the definition and more importantly the management of high-level security requirements. It also provides the components required (such as e.g. policy editors) to enable the administration of the system.

7.3.2.2 Adaptation Layer

The adaptation layer corresponds to the Abstract Security Functions (and associated models) of the system. It is called adaptation layer because it encapsulates the functions of an adaptation engine. Recall that Abstract Security Functions and associated abstract elements are a representation of the model from which Concrete Security Functions can be implemented in multiple ways. In addition, the interfaces defined between abstract and concrete security functions provide the logical mapping between functions at two levels of abstraction (adaptation layer and concrete security mechanism layer), and correspond to the sensors and effectors elements (see Figure 63) of self-adaptive systems and autonomic computing systems' architectures (see 2.2.1).

In the terms of autonomic computing and self-adaptive systems, and for the purpose of this methodology, this layer has a similar function to the autonomic manager and to the adaptation engine, respectively.

7.3.2.3 Concrete Security Mechanisms Layer

This layer corresponds to the Concrete Security Functions of the system and their associated interfaces. In the terms of autonomic computing and self-adaptive systems, this layer is equivalent to *managed elements* and *application logic*, respectively.

For an example of separation of concerns based on the above described layered architecture, consider the following scenario. Organisations AcmeA and AcmeB require to collaborate on different projects. At the Management Layer, AcmeA defines a high-level (management) policy that grants access to their resources to AcmeB's employees on a project type and role type basis. AcmeA also defines a high-level (management) security policy that requires high-level of security assurance due to the nature of the project. At the Concrete Security Mechanisms Layer, AcmeA deploys configurable access control and authentication applications on the interfaces of their IT resources. At the Adaptation Layer, AcmeA defines a context-based access control model parameterised per external company, project, user and role. AcmeA also defines context-based risk and trust models parameterised per level of security assurance required (in this example high-level). The Adaptation Layer configures the low-level concrete security applications based on the current state of the context-based models. This scenario highlights how the management, adaptation, and concrete implementation concerns are encapsulated and modularised. Notice that in this example the concrete implementation layer does not know about risk or trust models.

The development stage covers engineering, architectural, and structural aspects of the system. The following section covers management and operational aspects, and the adaptation layer in more detail.

7.4 Protection System Operational Stage

The development stage (section 7.3) provides the required abstractions, mappings, and all other required elements in a layered architecture that allows specifying the system. The operational stage is concerned with how to manage such elements.

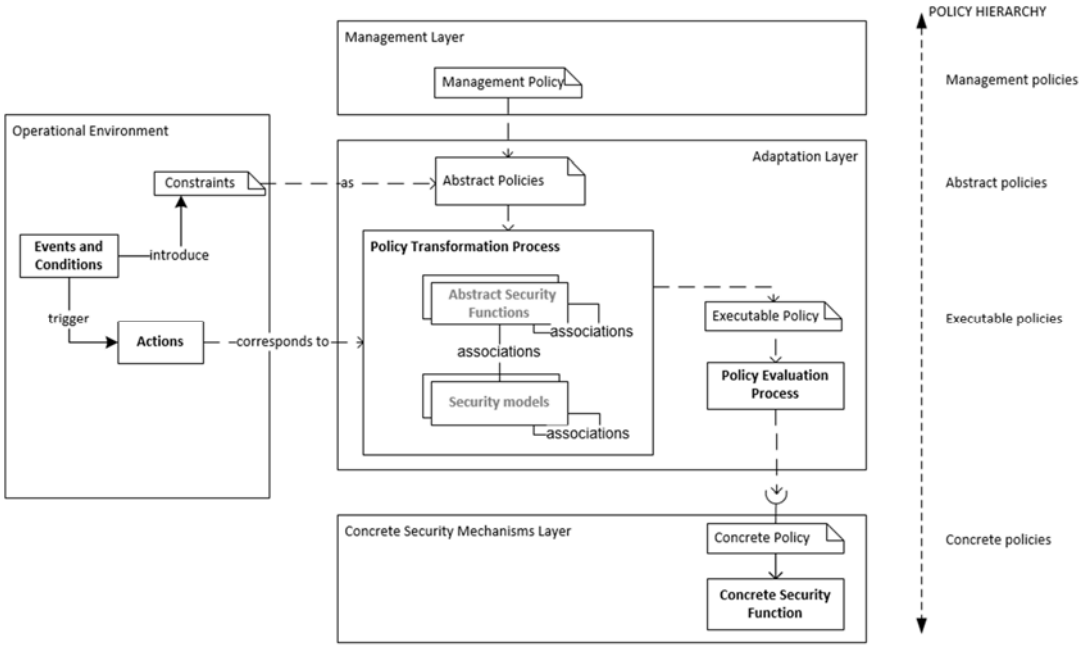


Figure 64 System Operational Stage Overview

In this section, first, a policy hierarchy for the management and operation of the system is defined, then, a model of the operational environment is described, and finally, the policy transformation process (already introduced and discussed) is incorporated into the proposed framework.

7.4.1 Policy hierarchy

The methodology organises policies in a policy hierarchy consisting of four types of policies each one corresponding to one of the three abstraction layers of the system. The policy hierarchy is shown on Figure 64 (right-hand side). Management policies corresponds to the management layer, abstract and executable policies to the adaptation layer, and concrete policies to the concrete security mechanisms layer.

7.4.1.1 Management policies

Management policies are defined by the system administrator(s) or power user(s), and reflect the decisions made about how the *high-level security requirements* must be enforced. Notice that high-level security requirements are not the same as management policies. For instance, a security requirement of the system could be “network traffic protection must be provided” or “firewalls must be provided at the perimeter of the network”, while a management policy could be “deny incoming traffic through ports above 5000 for non-premium clients”. At the management layer, management policies can be defined at different levels of abstraction, for example, using natural language via user-friendly policy editors or using more sophisticated languages via command control consoles. As shown in Figure 64, management policies are

mapped to corresponding abstract policies in the adaptation layer. Management policies are essentially baseline policies that define overall management decisions and scope the decisions made at the adaptation layer.

7.4.1.2 Abstract policies

At the adaptation layer, abstract policies are input into the transformation process (see Figure 64) from two sources. From the management layer via management policies, and from the operational environment via constraints (see section 7.4.2 for a detailed explanation). Notice that abstract policies are not the same as abstract security functions. Abstract policies determine how *abstract security functions* are configured or parameterised in the policy transformation process.

7.4.1.3 Executable policies

Executable policies also correspond to the adaptation layer and are the output from the policy transformation process. As mentioned in previous sections, after the transformation process, executable policies are ready to be evaluated.

7.4.1.4 Concrete policies

The policy evaluation process evaluates executable policies and outputs the concrete policies to be deployed on *concrete security functions* at the concrete security mechanisms layer. Concrete policies configure the concrete-level functions implemented by the protection mechanism. Technology-specific policies targeting security functions are achieved via the *interfaces translation layer* (see section 7.3.1).

7.4.2 Operational environment

Behaviour and contextual changes monitored or detected in the operational environment are modelled as events, conditions, actions, and constraints (see Figure 64). Given certain *event(s)* under certain *condition(s)* triggers an *action* that corresponds to and, in turn, triggers a specific *policy transformation process instance* to be initiated. At the same time, the same events, conditions, and actions introduce constraints into the transformation process. Events, conditions, actions, and constraints can be expressed in the form of policies. Consider as an example a snippet of the operational environment model for the authentication mechanism described in Chapter 6. This is shown in UML notation in Figure 65.

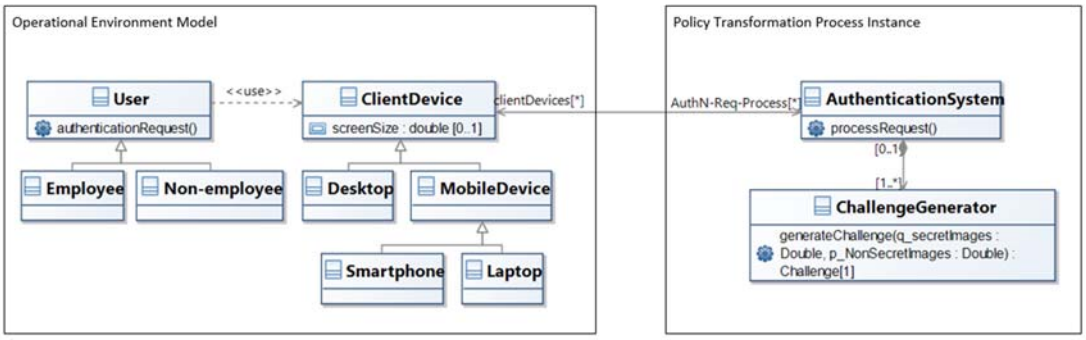


Figure 65 Operational Environment Model – Authentication Mechanism Example

Policies are specified using information from the model(s). In this example, an event could be "user A requests authentication to Authentication Server X", a condition could be "user A is an employee", and associated constraints could be information about "the type of client device used, time, location, etc." The action "requests authentication" is mapped to and triggers the policy transformation process instance. Formally, policies are expressed using an adequate policy language. Notice that in this example the information in the constraints consists of parameter values only. However, it is also possible that the constraints contains actions. For example, in the protection mechanism described in Chapter 4, the policy "Processing unit PU access Data file DF at location XY" is the *event* that triggers a transformation process instance (i.e. policy combination in the example) and it is also this policy itself that is introduced as the *constraint* into the transformation process (for the purposes of policy combination).

7.4.3 The policy transformation process extended with security models

As mentioned before, abstract security functions at the adaptation layer are mapped to concrete security functions at the concrete security mechanisms layer. However, abstract security functions are encapsulated in policy transformation processes as logical adaptation units. In other words, the policy transformation process enables to introduce abstract security functions and security models that incorporate adaptation logic and strategies – see Figure 64. The adaptation layer is an abstract representation of the system and can be extended without breaking the mapping between abstract and concrete functions as long as the defined interfaces are maintained and respected. This characteristic of the system allows for the incorporation of information (security) models at the adaptation layer that are not supported at the layer below.

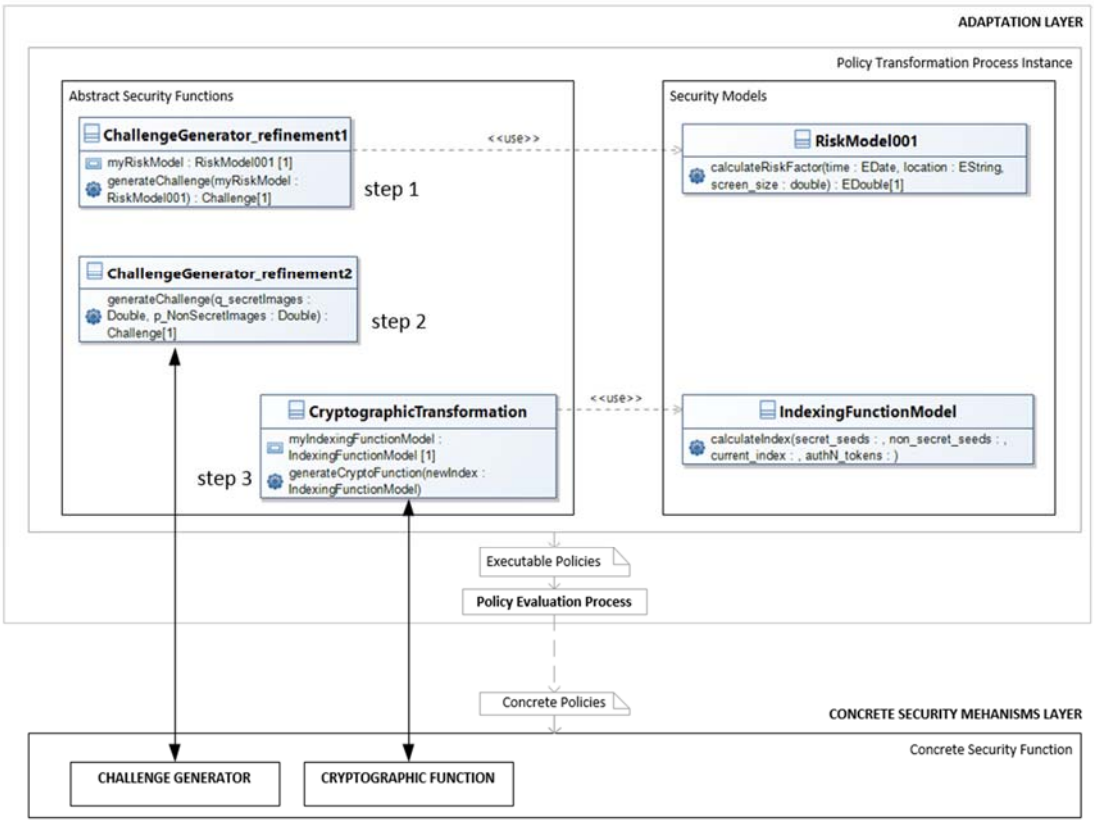


Figure 66 Adaptation Layer Model – Authentication Mechanism Example

Consider a partial representation of the adaptation layer model for the authentication mechanism described in Chapter 6. This is shown in Figure 66. In this example, the abstract security functions *ChallengeGenerator_refinement2* and *CryptographicTransformation* are mapped to the concrete security functions *Challenge Generator* and *Cryptographic Function.* However, two adaptation strategies are introduced in the policy transformation process.

First, the functions *ChallengeGenerator_refinement1* and *ChallengeGenerator_refinement2* correspond to a two-step refinement of the function *Challenge Generator* at two levels of abstraction. The first refinement is a function that takes as arguments values from a risk model. The second refinement is a function that takes as arguments values related to the characteristics of a grid of images (secret and non-secret images). Each refinement introduces configurations (i.e. constraints) about orthogonal aspects. Another example of refinement could be a function first preconfigured (i.e. specialised) in terms of business constraints (e.g. service level agreements) and then further refined in terms of technical aspects.

Second, the functions *ChallengeGenerator_refinement1*and *CryptographicTransformation* are enhanced with security models, in this case *RiskModel001* and *IndexingFunctionModel*, respectively. Without these two security models, the security functionalities of the system would correspond to conventional functionality of the functions *Challenge Generator* and *Cryptographic Function* in a traditional authentication mechanism. However, by

incorporating the security models abovementioned the function *Challenge Generator* is extended with the capability of generating a challenge by reasoning about a risk model while the function *Cryptographic Function* is turned into a function that is dynamically reconfigured at runtime based of an indexing model. The introduction of security models allows introducing adaptation logic and variability, and extending the security capabilities of the protection mechanisms.

The way abstract security functions and security models are structured within the transformation process (e.g. sequentially as shown by the labels step 1, 2, and 3 in Figure 66, or in parallel, etc.) depends entirely on the requirements of the system and how it is modelled. However, the proposed policy transformation process differentiates and separates abstract security functions from security models in order to achieve modularity and reusability. For example, *RiskModel001* and *IndexingFunctionModel* could be easily replaced by other security models. Additionally, it could be possible to have a library of security models at the adaptation layer readily available for the composition of adaptation logic for diverse security mechanisms.

7.5 Methodology stepwise guidance

In this section a stepwise approach to describing the blueprint of the methodology from a security perspective is presented.

1. *Define the high-level security requirements or security capabilities of the protection mechanism and identify protected assets, possible threats, vulnerabilities, and attacks.* This is important in order to reason as to what and where security controls and security functions should be placed within the IT infrastructure and how these controls and functions map to the security requirements of the system.

2. *Identify relevant constraints to be introduced into the system.* Types of constraints include: (a) those that are determined by the physical or behavioural constraints of the operational/execution environment itself such as e.g. the screen size of a device or the possible control-flows and data-flows defined in a business process description. This type of constraints requires understanding the operational/execution environment and modelling entities and interactions among them; (b) information about the different states of the operational/execution environment. States may be defined at different levels such as transaction or system level; (c) constraints related to QoS aspects such as security, performance, and usability; and (d) contextual parameters, among others.

3. *Identify relationships among constraints and security concerns*. This is a very important step in the methodology and involves modelling relationships. Establishing relationships among constraints allows reasoning about causal relations among constraints and security concerns such as risk, trust, level of assurance, threats, vulnerabilities, and attacks; as well as considerations regarding trade-off between security and other non-functional requirements.

4. *Define abstract policies based on relationships among constraints.* Abstract policies include different types of constraints possibly defined by different security entities or actors involved in a given execution.

5. *Determine the type of adaptive behaviour to be induced during policy transformation*. This will depend on the level of sophistication required for the different abstract policies defined previously and the type of causal relationships among them. There are two types of adaptive behaviour that can be introduced into the system and that may co-exist in a complimentary or combined way at the same time: (a) *context-based adaptation*: If constraints are used to drive the policy transformation process then they are considered to be context-based adaptive constraints. Typically, context-based adaptive constraints are defined in the form of rules or policies that specify under what circumstances other policies can be combined; and (b) *specialisation*: If constraints are introduced due to behaviour and are combined during the policy transformation process then they are considered as specialising constraints.

   Typically, policy transformation with context-based adaptation is considered to be more complex since it is essentially a context-dependent policy transformation whereas simple specialisation (without requiring to resolve context-based parameters) occurs as a result of behaviour in the execution environment.

6. *Define appropriate policy transformation techniques*. Again, this depends on the type of abstract policies defined and the level of complexity in their causal relationships. Policy transformation techniques have been classified in two categories: (a) *instantiation process*: it based on the parameterisation of constraints on a given abstract policy; and (b) *integration process*: it is based on the mixing of constraints from different abstract policies in order to generate a new abstract or executable policy with the resulting constraints.

   In the case of instantiation process, typically the abstract policy to be parameterised defines an execution scope that can be further refined. In terms of scoping it is easier or

less complex to verify correctness of the resulting policy in the sense that the base policy already scopes the execution in the first place. In the case of integration process, the resulting scope of execution after applying integration can be reduced but also can be extended in terms of constraints. Therefore it provides more flexibility, but it is more complex to determine how an integrated policy will affect behaviour on the overall system. Similar to the instantiation process technique, the integration process technique can be scoped by having a set of transformation rules as baseline policies defining constraints on how the integration process occurs, for example, as in the case of the protection mechanism of Chapter 4. In addition, in both techniques there can be only simple specialisation, or specialisation plus context-based adaptation depending on the logic behind the corresponding policy transformation process.

An additional aspect to consider is the order in which policy transformation techniques can be applied. Depending on the level of sophistication of the protection system and on the nature of the abstract policies defined it can be necessary to apply several steps of policy transformation. As shown in Figure 60 a fairly complex protection mechanism requires at least two steps: one for instantiation and one for integration of abstract policies. Intuitively, the greater the decomposition/number of policy transformation stages the easier it is to understand and define causal relationships among abstract policies.

7. *Define the appropriate policy evaluation process*. This process takes as input executable policies. If the executable policies are context-based then contextual information needs to be dynamically resolved and the policy evaluated (context-based adaptation). Therefore, it is important to understand how the protection mechanism implements the evaluation process. In the case of legacy systems, the evaluation module can be too basic or limited on the amount of contextual information that can resolve in which case one option would be to consider externalising the policy evaluation module. Another alternative would be to apply context-based adaptation in the Policy Transformation phase thus influencing the executable policy introduced in the system. In the case of a sophisticated protection mechanism with an existing context-based evaluation module, such as e.g. XACML, it might be more sensible to induce the context-based adaptive behaviour internally.
   An additional potential scenario is the case in which contextual information needs to be resolved internally as well as externally, or a case in which the evaluation module cannot evaluate a policy. In such cases a protocol is required to pass control back to the policy transformation phase.
8. *Define the appropriate monitoring process*. This depends on what information or events are or become relevant, and should be monitored. Moreover, as mentioned before the monitoring process can be event-based or query-based depending on whether the process

signals events to other processes, i.e. transformation and evaluation processes, or whether the latter require to resolve contextual parameters, respectively.

9. *Define the appropriate detection process*. This process overlaps with the monitoring process in that it is event-based and communicates with the policy transformation process. However, the main difference is in that it can correlate information and events incoming from the monitoring process, infer non-expected but relevant execution state or context information, and to adaptation strategies to be fed to the policy transformation process. This aspect is mentioned for completeness but is out of scope in this research.

The above described stepwise guidance directly relates to the fundamental research questions and system requirements extracted in chapter 3 and also the research aims of the introductory chapter. The next section presents a discussion on the fulfilment of the system requirements by the proposed framework and stepwise guidance.

7.6 Compliance with System Requirements

Table 15 summarises the system requirements for policy-driven adaptive protection systems described in chapter 3.

Table 15 Policy-driven Adaptive Security System Requirements

	Requirements
Policy Hierarchy and translation / mapping of policies	
R1	To provide a policy hierarchy that facilitates system management, provides different policy abstraction levels, and enables different semantics for expressing concerns separately, i.e. implementation-specific, adaptation and management.
R2	To provide consistent transformations and mappings between the policies of the policy hierarchy.
Three-Layer Architecture: management, adaptation, and implementation	
R3	The separation of concerns across the following levels: management, adaptation, and application; and the externalisation of the adaptation logic.
R4	The management and control (i.e. management level) of internal behaviour of the application logic (i.e. implementation level) via policy scoping while providing flexibility in the definition of executable policies (i.e. adaptation level)..
Modelling the operational environment	
R5	To provide a model the operational environment in a way that allows reasoning about its entities (including their characteristics) and behaviour.
The property of specialisation	
R6	To dynamically capture constraints and requirements in the operational environment and to use them to specialise the adaptation process.
Enhancing adaptation with security models	
R7	To incorporate and make use of security models to enhance the adaptation logic to dynamically support security decisions.

7.6.1 Policy Hierarchy and translation / mapping of policies

7.6.1.1 Requirement R1

The proposed framework provides a policy hierarchy that separates management, adaptation, and implementation-level concerns. At the management level, high-level business goals or policies can be expressed in terms of high-level security requirements independently reducing the system management complexity. Similarly, implementation-level requirements are tied to specific concrete-level policies of the target system. At the adaptation-level different types of abstract policies sourced from the management-level and the operational environment are incorporated and processed according to different adaptation strategies and models.

7.6.1.2 Requirement R2

Consistent transformations and mappings between policies in the hierarchy is achieved by structuring the system in a 3-layered architecture. High-level security requirements are translated into abstract-level policies which are tied to abstract functions of the adaptation layer. Similarly, the executable policy resulting from the policy transformation process is mapped to low-level policies tied to concrete-level security functions via the interfaces translation layer (see Figure 62)

7.6.2 Three-Layer Architecture: management, adaptation, and implementation

7.6.2.1 Requirement R3

The proposed framework separates the adaptation logic from the application logic (i.e. the logic implemented by the target system). This decoupling allows modelling and defining security functions and models out of the scope of the application logic. Recall the example of a firewall component (application logic) configured based on rules dynamically obtained from a risk model defined at the adaptation-layer (adaptation logic). Additionally, management-level requirements are also treated separately as input to the adaptation logic.

7.6.2.2 Requirement R4

Management-level policies introduce constraints as scoping mechanism and drive the overall behaviour of the system by enforcing management goals while at the same time constraining the behaviour of external entities. This is reflected by the executable policies resulting from the policy transformation process. Moreover, the transformation process allows the flexibility to introduce external entities' constraints into the system (within the scope defined by the management baseline policies of the system). Recall the example of section 7.1.1.1 where the

external entity PU1's policy changed the parameter "location" from *home* to *work* resulting in a new *applicable executable policy* without any modification of the baseline policies of the system.

7.6.3 Modelling the operational environment

7.6.3.1 Requirement R5

Events, conditions, actions and associated constraints captured in the operational environment are explicitly associated to policy transformation process instances. This is an advantage because it allows: reasoning about the operational environment's entities and their behaviour and linking them to the adaptation process. Additionally, by having an explicit model of the operational environment it is possible to readily define and operationalise policies based on entities' security requirements and constraints.

7.6.4 The property of Specialisation

7.6.4.1 Requirement R6

The specialisation property is achieved by means of the policy transformation process at the adaptation level. External entities' requirements and management baseline policies are processed by a combination of security functions, security models, and security strategies.

7.6.5 Enhancing adaptation with security models

7.6.5.1 Requirement R7

By having an adaptation layer it is possible to externalise the adaptation process and extended with security models to support abstract security functions aiming at improving adaptation decisions. As mentioned before, security models such as risk, trust, policy overrides, and so on, can be incorporated into the adaptation logic even if such models or concepts do not exist at the application level (i.e. the target system).

7.7 Concluding Remarks

The proposed framework and stepwise guidance is an integral and comprehensive approach for the realisation of policy-based adaptive protection systems. The three very different protection mechanisms of Chapter 4, Chapter 5, and Chapter 6 have been analysed in detail in order to extract common core components and articulate the property of specialisation.

From the analysis, two types of adaptive behaviour has been extracted, articulated, generalised and defined: *specialisation* and *context-based adaptation*. The policy transformation process

has been articulated, decomposed, and generalised into two sub-processes: *integration* and *instantiation*. The proposed model of the operational environment has been conceptualised as entities that perform or intend to perform actions on other entities, be it external entities or internal entities (i.e. components of the policy-based adaptive system itself). The *development stage* proposed follows a software engineering approach and provides the required abstractions, mappings, and all other required elements in a 3-layered architecture that allows specifying the system. The *operational stage* is concerned with the runtime operation and control over the elements and constructs of the 3-layered architecture achieved by the development stage. The capability of the framework to accommodate diverse types of security models at the adaptation level allows for the design and specification of complex security adaptation strategies. The *stepwise blueprint guidance* proposed provides a generic systematic way for reasoning and addressing the different architectural and behavioural aspects of the system to be designed to achieve specialisation via policy transformation.

The following chapter presents the conclusions and future work.

Chapter 8

Conclusions and Future Work

This chapter expands on the abovementioned aspects, presents their novelties and contributions. This is followed by a list of implementation aspects that need to be addressed when applying the proposed methodological framework. The chapter ends with the research challenges identified as future work.

8.1 Conclusions

This thesis proposes a stepwise blueprint and a general methodological framework for the realisation of *policy-driven protection mechanisms* that can be *specialised* to specific *operational environments* that are *context-aware* and *adaptive*. It has been introduced a general architecture aimed to capture the building components for the realisation of adaptive protection mechanisms including core design aspects of an adaptation engine.

Three very distinctive context-aware policy-driven adaptive protection mechanisms have been proposed in this thesis, with their own novelties in their own fields. They have been analysed and common core elements, components, methods, and processes were extracted and generalised in order to incorporate them in a general methodological framework.

The major novelties of the proposed protection mechanisms are:

A secure execution context enforcement framework [7]. The framework acknowledges different types of policies issued by different administrative entities. The framework defines an active secure execution context that integrates, combines, and enforces the applicable policies based on system state, contextual information, and baseline policies. It provides a

novel policy integration and combination mechanism that allows the *dynamic runtime configuration of new dynamically generated security profiles*, and also proposes an *enforcement architecture*. The dynamic generation characteristic of this proposal is novel and different from the related work that only selects from a set of predefined security profiles. The resulting active execution context is generated depending on the system state and contextual information, and when the interacting entities become known. This is an important and desirable characteristic in highly dynamic, platform-open, and multi-purpose systems, such as smart devices, where predefined security profiles would be inflexible or difficult to define.

A framework for the enforcement of privacy and security requirements in internet-centric services [8] that enables the data providers (DP) the definition of data protection policies and enforces them on the infrastructure of the data consumer (DC). It proposes a mechanism that adds constraints on the DC's execution workflows based on contextual information and user preferences (i.e. DP's external constraints). The main novelty of the framework is that it *allows the DP to specify not only privacy and access control-related constraints before any data disclosure* but also *to specify additional security and assurance-related constraints to be enforced after private data is disclosed*. The proposed policy framework is flexible enough to be used at different levels of abstraction in a business process to express typical privacy and access control requirements, while providing also a way to express information-flow control constraints.

A multi-factor dynamic pin authentication mechanism [9] that generates a pseudo-random dynamic pin based on the user input, different authentication factors, and past successful authentication attempts. The main novelty is the *cryptographic transformation function that generates the dynamic pin*. For each authentication attempt the crypto-function changes itself dynamically based on external contextual constraints provided by the user and the operational environment. The crypto-function provides pseudo-randomness to the authentication process since the crypto-algorithms it uses changes dynamically making it more difficult to perform cryptanalysis attacks. Another novelty of the authentication mechanism is that *it takes into account the risk associated to the authentication transaction in order to tune an adequate level of assurance while providing the best available usability criteria during the generation of the challenge*. This is achieved by two policy transformation steps: policy selection and policy combination.

From the analysis the three protection mechanisms, two types of adaptive behaviour has been extracted, articulated, generalised and defined: *specialisation and context-based adaptation*. Specialisation is a property achieved by means of the policy transformation process and enables the incorporation of external entities' requirements and their harmonisation with

management-level requirements. Context-based adaptation has been articulated as a complementary aspect that drives behaviour in the system and may be present in both the policy transformation phase and the policy evaluation phase.

From the same analysis, the policy transformation process has been articulated, decomposed, and generalised into two sub-processes: *integration and instantiation*. In the instantiation process constraints defined in an abstract policy are further refined by resolving constraints that need to be parameterised from another policy or set of policies. In the integration process constraints defined in different abstract policies are mixed together, e.g. by means of logical operators, in order to produce a policy with a new set of constraints. Depending on the system to be designed, these two sub-processes are generic mechanisms that can be applied systematically in modular steps to enable policy transformation for diverse type of protection mechanisms. This includes policy combination, policy matching, policy refinement, policy selection, and policy integration, as demonstrated for the three protection mechanisms analysed. Although complementary, this is different to simply a top-down policy refinement as described in most of the works of Chapter 2.

An adaptive protection mechanism must consider its operational environment. The proposed model of the operational environment has been conceptualised as entities that perform or intend to perform actions on other entities, be it external entities or internal entities (i.e. components of the policy-based adaptive system itself). This has two advantages. Firstly, diverse types of system behavioural/architectural patterns can be modelled such as *request/response* (Chapter 5, and Chapter 6) and *execution control* (see Chapter 4). Secondly, behaviour and events can be encapsulated in the form of policies (e.g. *{subject, object, and action}*) and incorporated into the adaptation process. In addition, from the same representation of the operational environment any type of information can be used as contextual parameters to be monitored. In other words, the proposed framework distinguishes between behaviour manifested as events (event-based) as well as parameter values required to be resolved during policy transformation and policy evaluation (query-based).

The development stage proposed follows a software engineering approach and provides the required abstractions, mappings, and all other required elements in a 3-layered architecture that allows specifying the system. The software engineering approach described in the proposed framework consists of three phases: software security requirements, system design and specification, and concrete-level software implementation. In the development stage, *high-level security goals* have a correspondence with the software security requirements phase; *abstract security functions and extended security models* have a correspondence with the system design and specification phase, and *concrete security functions* have a

correspondence with the concrete-level software implementation phase. Moreover, *abstract security functions and extended security models* are proposed to be designed and structured following a model-driven engineering approach which allows exploiting abstracts domain models (in this case the security domain at the adaptation level) decoupled from lower levels of abstraction (in this case concrete-level security software implementations).

The operational stage is concerned with the runtime operation and control over the elements and constructs of the 3-layered architecture achieved by the development stage. The operational stage proposed defines a policy hierarchy where management policies corresponds to the management layer, abstract and executable policies to the adaptation layer, and concrete policies to the concrete security mechanisms layer. This policy hierarchy has a function similar to the policy-based management systems studied in Chapter 2 but with extended support to enable specialisation via policy transformation as described in this chapter.

Differently to the approaches discussed in Chapter 2, the proposed framework distinguishes and separates abstract policies and abstract security functions. This allows for the introduction of different types of constraints as input to abstract security functions and as a result a reconfiguration effect is achieved at the adaptation level. Other works treat abstract policies and abstract security functions indistinctively.

The capability of the framework to accommodate diverse types of security models at the adaptation level allows for the design and specification of complex security adaptation strategies that otherwise would not be possible to introduce at the level of concrete implementation. This is particular important in concrete implementations where the codebase and low-level policy model cannot be modified/recompiled to influence behaviour. Instead, by using executable policies that result from the externalised adaptation logic it is possible to influence the behaviour of such implementations.

The stepwise blueprint guidance proposed provides a generic systematic way for reasoning and addressing the different architectural and behavioural aspects of the system to be designed to achieve specialisation via policy transformation. This includes d*efining the high-level security requirements or security capabilities of the protection mechanism and identify protected assets, possible threats, vulnerabilities, and attacks*; i*dentifying relevant constraints to be introduced into the system* (both management-level and external); i*dentifying relationships among constraints and security concerns* (for the mapping of behaviour in the operational environment to abstract and concrete security functions)*; defining abstract policies based on relationships among constraints* (tied to the policy hierarchy)*; determining the type of adaptive behaviour to be induced during policy transformation* (specialisation and context-based adaptation)*; defining appropriate policy transformation techniques* (based on

the instantiation and integration sub-processes)*; defining the appropriate policy evaluation process* (that evaluates the resulting executable policies)*; defining the appropriate monitoring process* (i.e. event-based and query-based)*; and defining the appropriate detection process* (i.e. when and what adaptation strategies to feed to the adaptation logic).

The transformation process is determined based on the desired high level security requirements and adaptation requirements. The system security requirements provide guidance and they are associated to abstract security functions. Management actions are derived from high level security requirements. The desired adaptation requirements are used to create the transformation model. Adaptation requirements help elucidate important aspects such as what information models to consider, e.g. abstract entities that relate to concrete entities of the implementation-level, or abstract entities with no direct mapping to concrete elements but models that extend the concepts of the implementation. For example, a contextual model. In any case the transformation model is extensible and it is effectively a set of models.

The proposed methodological framework considers the external entities interactions as a way to specialise the system via the policy transformation process. External entities introduce constraints into the system (explicitly or implicitly) which impose restrictions on the system model, this is called here specialisation. Baseline policies impose restrictions in external entities' behaviour.

This is different to having a set of policies that are activated at a given time. In the proposed methodology the system introduces or generates a complete new policy that has not been predefined. This occurs because the policy to be generated depends on the external entities' input at runtime.

8.2 Implementation challenges for following the methodology

The following are implementation challenges that anyone will face when following the methodology blueprint.

Requirements Analysis Challenges

Requirements analysis for a protection mechanism involves analysing both functional and non-functional requirements since both are interrelated. Security requirements includes determining security high-level goals, identifying the *protected assets, possible threats, vulnerabilities, and attacks*. Two issues need to be considered. First, how to map security requirements to security controls; and second, during the operational phase how security

requirements affect non-functional requirements (e.g. performance, usability, etc.) and how to capture this in a model.

Identify Relationships among Constraints and Security Concerns Challenge

Establishing relationships among constraints allows reasoning about causal relations among constraints and security concerns such as risk, trust, level of assurance, threats, vulnerabilities, and attacks; as well as considerations regarding trade-offs between security and other non-functional requirements. Issues include how to model these different types of constraints, how to relate them to contextual and execution state parameters and to what degree, what relationships among them are useful for adaptability.

Policy Model Challenges

One core issue to address is what policy model is appropriate for the adaptation engine that is able to capture different types of constraints and express relationships among them.

Policy Transformation Challenges

Translating between abstract policies and executable policies requires explicit representation of policies and translation algorithms in order for the policy transformation process to be effective and efficient in particular in large-scale protection mechanisms. One challenge that needs to be addressed is what policy model and what policy transformation algorithms suit best a particular type of protection mechanism.

Recall that the result of the policy transformation process is an executable policy. The executable policy is executable in the sense that has been fully refined. However, it is evident that an adaptation engine is expected to interface to several security controls or functions with implementation-specific policy models. The issue is how the adaptation engine should interface to diverse security controls that expect specific executable policies. This is an interoperability issue.

Design Challenges

In the approach taken in this investigation, an adaptive protection mechanism is decomposed into the adaptive element, i.e. the adaptive engine, and the protection mechanism by itself. As shown in Figure 61, the policy evaluation module and the monitoring module are part of the adaptation engine. However, sometimes these modules are already part of the protection mechanism itself (e.g. XACML engine) and full decomposition is not possible. In such cases, the issue is how to design the remaining modules of the adaptation engine with an adequate architectural interfaces so that can connect to the policy evaluation and monitoring modules

of the protection mechanism. More generally, the issue is how building an adaptive protection mechanism from legacy systems affect the overall design and architecture.

Another issue relates to *design for adaptability* [6] and has to do with what sensors exists for monitoring information and events and what effectors, i.e. security functions or controls, exist for variability.

Monitoring Challenges

Recall that the monitoring module monitors context and the state of execution. The issues here are what information or events to monitor and when it becomes relevant to monitor them, and how to capture them. To properly address these issues it is required to explicitly model the execution environment and the context. So an additional issue is what the most appropriate models are.

8.3 Research Challenges and Future Work

The work described in this thesis has been concerned with the realisation of policy-driven (self-) adaptive protection systems. In order to progress in this field, one of the most important challenges that remains to be solved by the research community is the evaluation of (self-) adaptive systems.

8.3.1 Research Challenge: Adaptive Security and Evaluation

Some works have partially addressed the performance evaluation of factors such as security and safety. However, what evaluation criteria and metrics can be used to measure the *quality of security adaptation* and how to apply them is still an open issue. Although there might be some research ideas in areas such as QoS and control theory, comprehensive work in adaptive security and closer areas such as autonomic and self-adaptive systems is minimal.

8.3.2 Future work: A Quality of Security Adaptation Framework

The validation and verification of (self -) adaptive software systems is necessary to ensure that adaptation mechanisms function properly and can be trusted. In order to evaluate such systems it is required to identify their adaptation properties and associated metrics.

In [133], the authors propose a framework for evaluating quality-driven self-adaptive software systems. The framework provides a set of dimensions to classify adaptive systems; a list of adaptation properties observable in the adaptation loop (consisting of two components: the managed system and the controller); a mapping from adaptation properties to software quality attributes; and, different quality metrics to evaluate quality attributes and adaptation

properties. The following adaptation properties are considered: accuracy, settling-time, small-overshoot, robustness, termination, consistency, scalability, and security. *Security* is classified based on two dimensions: *where the property is observed* and *the proposed evaluation mechanism*. Regarding the former, security is an adaptation property observable in both: the managed system and the controller. Regarding the latter, security requires *dynamic evaluation* – data gathered by directly monitoring the running system in order to expose the actual behaviour[134]. Furthermore, the authors establish a mapping between security and the following *software quality attributes*: confidentiality, integrity, and availability. They highlight the fact that security must be evaluated independently on the controller and on the managed system since ensuring security in one does not guarantee security on the other. Moreover, the software quality attributes should be measured on the managed system since they are usually not visible on the controller. In an extensive survey on representative self-adaptive systems, the authors also analyse adaptation metrics to measure performance, dependability, and safety. However, security is not addressed.

As part of the future work, it is proposed the development of a framework for the evaluation of security adaptation properties within the context of policy-driven adaptive protection mechanisms. The quality of adaptation framework to be proposed would leverage on the fundamental link that exists between security as adaptation property and security in terms of software quality attributes, as highlighted by [6, 133].

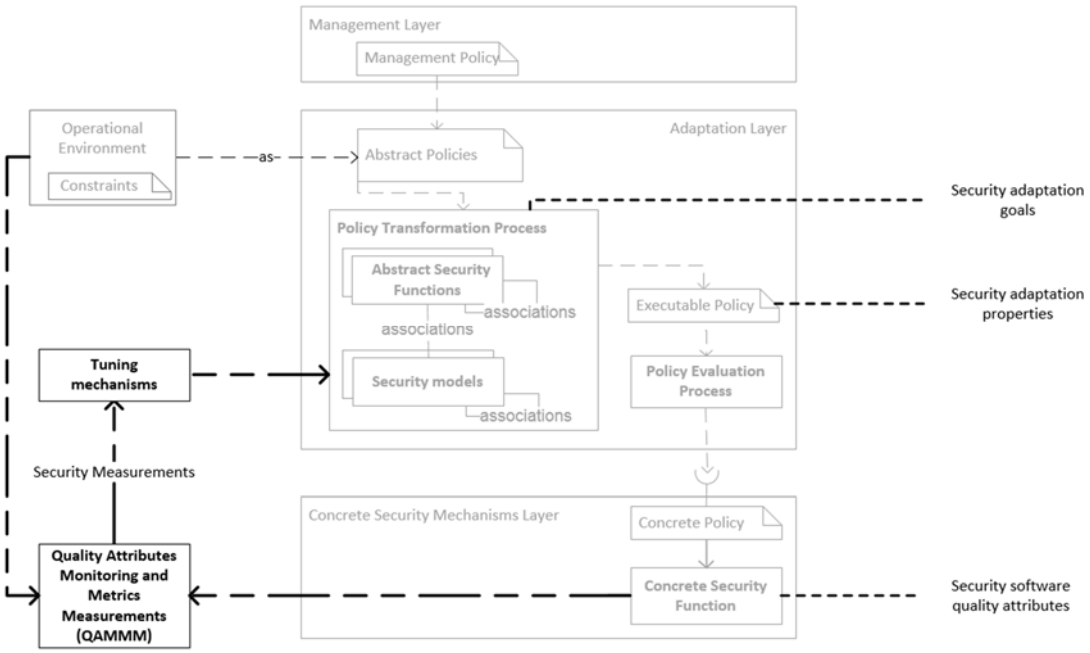


Figure 67 Quality of Adaptation for the proposed framework

Figure 67 shows the 3-layered architecture of the framework for policy-driven adaptive protection mechanisms integrated with an initial set of conceptual components and mappings for enabling the measurement of the quality of adaptation.

As depicted in Figure 67, the policy transformation process consists of a set of abstract security functions and security-enhancing models. During the design phase, the policy transformation process is modelled based on the *security adaptation goals* desired for the system. For example, if one goal is to provide *adaptive authorisation* based on the level of a perceived risk, the policy transformation is modelled in a way that the *security-enhancing models* incorporate risk aspects, while *abstract functions* incorporate parameters related to authorisation such as authentication of entities and access rights. Such parameters can be used to characterise the *executable policy* (resulting from the policy transformation process) in terms of *security adaptation properties*. That is, the goal of adaptive authorisation can be achieved via two security sub-properties, *authentication* and *access control*; and such sub-properties are mapped to the executable policy.

If the executable policy contains elements of the (sub-) properties, a weighted sum expression could be constructed to capture both the individual and the combined importance of the (sub) properties involved [135]. For example,

$$f(f_1, f_2) = w_1 authentication + w_2 accessControl \qquad (8.1)$$

How to model the policy transformation process and the executable policy, what security (sub) properties, how to map the properties to the executable policy, and what criteria to use, are important questions that are part of the future work.

As mentioned before, security adaptation (sub) properties can be linked to security software quality attributes. Once evaluated, the executable policy is translated into a concrete function which, in turn, is used as input to the concrete security function(s). In order to associate the adaptation (sub) properties (carried in the executable policy) to the security quality attributes, it is required to understand the way the concrete security functions are configurable, to characterise the implementation of these functions, and to identify the relevant quality attributes.

Important questions are how to model the concrete security functions in terms of quality attributes, and what taxonomy of quality attributes can be used to enable the mapping to the adaptation (sub) properties. For the future work investigation, [136] provides a security taxonomy for quality attributes that could be used as the starting point towards a more general taxonomy. The taxonomy is depicted in Figure 68 and includes the following aspects: *concerns* – i.e. the quality attributes to be measured; and *factors* – i.e. properties of the

concrete level implementation and their execution environment that have an impact on the concerns.

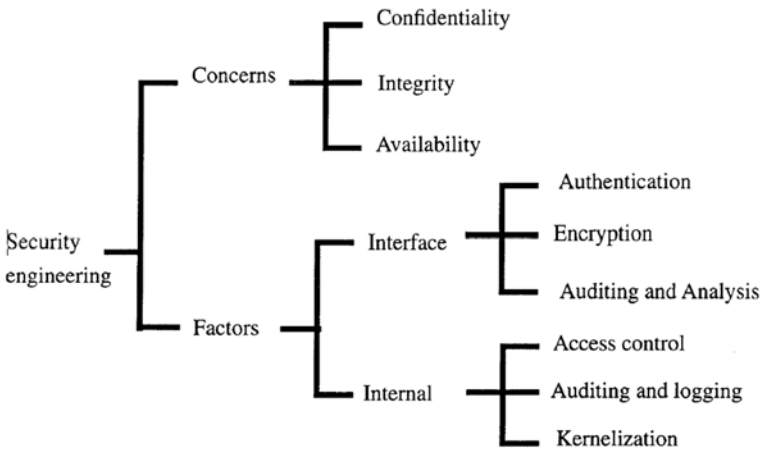


Figure 68 SEI security taxonomy

The evaluation of adaptive security systems requires to be dynamic. The behavior of the managed system is driven by and depends on the following: the environmental and contextual information perceived by the system, the security factors (see Figure 68), and the concrete security policy (i.e. the managed system's configurable input). As part of the future work, it is necessary to investigate what type of security metrics can be identified and how to use them to measure the security software quality attributes (i.e. concerns). Figure 67 depicts the conceptual component that would encapsulate such functionality, i.e. *Quality Attributes Monitoring and Metrics Measurements (QAMMM).*

The output of the *QAMMM* are *security measurements* expressed as quality attribute values and obtained by applying *measurement functions* that take as input the metrics being monitored on the system. For the future work, it is envisioned that such *security measurements* have the potential of being used in two (complimentary or alternative) ways: evaluation by comparative analysis, and evaluation by feedback analysis.

8.3.2.1 Evaluation by comparative analysis

The framework for policy-driven adaptive protection mechanisms is generic enough for modelling and implementing different types of adaptive security systems under the same design and operational criteria. Moreover, the proposed methodology also enables the modelling of different policy transformation processes for a single adaptive system. Therefore, *security measurements* can be used to compare adaptive security properties between different systems / policy transformation models.

8.3.2.2 Evaluation by feedback analysis

The *security measurements* can be used as feedback information to the adaptive layer via the *tuning mechanisms* component shown in Figure 67. Recall that the executable policy can be expressed as a weighted sum expression of the form $f(f_1, f_2) = w_1 authentication + w_2 accessControl$, and that the security adaptive (sub) properties of the expression are mapped to software security quality attributes. In a similar way, the *security measurements* can be expressed using the same type of expression, $f(f_1', f_2') = w_1' authentication + w_2' accessControl$. If the weights w_i correspond to the initial values (i.e. importance) given to the (sub) properties, and the weights w_i' correspond to the values calculated by the *QAMMM* component in real time and conditions, then tuning functions could be used to update the weights w_i with respect to the weights w_i'. This would allow evaluating the security adaptive properties of the policy transformation process by comparing initial or previous security property weights with respect to real-time *security measurements*; and potentially, tuning the weights w_i, accordingly.

Section 8.3 introduced the conceptual components and mappings of a quality of security adaptation framework for the evaluation of (self-) adaptive systems. The framework focuses on *dynamic evaluation* for the measurement of software quality attributes during runtime, and proposes two types of mechanisms for evaluation: comparative analysis and feedback analysis. Section 8.3 presents the core ideas as the future research path for *quality of adaptation* within the context of policy-driven (self-) adaptive protection systems and specialisation.

REFERENCES

[1] B. Hashii, S. Malabarba, R. Pandey, and M. Bishop, "Supporting reconfigurable security policies for mobile programs," *Comput. Netw.,* vol. 33, pp. 77-93, 2000.

[2] A. Elkhodary and J. Whittle, "A Survey of Approaches to Adaptive Application Security," presented at the Proceedings of the 2007 International Workshop on Software Engineering for Adaptive and Self-Managing Systems, 2007.

[3] N. MacDonald, "The future of information security is context aware and adaptive," Technical report, Gartner RAS Core Research, 2010.

[4] B. H. Cheng, R. De Lemos, H. Giese, P. Inverardi, J. Magee, J. Andersson*, et al.*, "Software engineering for self-adaptive systems: A research roadmap," in *Software engineering for self-adaptive systems*, ed: Springer, 2009, pp. 1-26.

[5] R. De Lemos, H. Giese, H. A. Müller, M. Shaw, J. Andersson, M. Litoiu*, et al.*, "Software engineering for self-adaptive systems: A second research roadmap," in *Software Engineering for Self-Adaptive Systems II*, ed: Springer, 2013, pp. 1-32.

[6] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," *ACM Trans. Auton. Adapt. Syst.,* vol. 4, pp. 1-42, 2009.

[7] Y. Diaz-Tellez, E. L. Bodanese, F. El-Moussa, and T. Dimitrakos, "Secure Execution Context Enforcement Framework Based on Activity Detection on Data and Applications Hosted on Smart Devices," in *Social Computing (SocialCom), 2013 International Conference on*, 2013, pp. 630-636.

[8] D.-T. Yair, "An Architecture for the Enforcement of Privacy and Security Requirements in Internet-Centric Services," 2012, pp. 1024-1031.

[9] Y. Diaz-Tellez, E. Bodanese, T. Dimitrakos, and M. Turner, "Context-Aware Multifactor Authentication Based on Dynamic Pin," in *ICT Systems Security and Privacy Protection*. vol. 428, N. Cuppens-Boulahia, F. Cuppens, S. Jajodia, A. Abou El Kalam, and T. Sans, Eds., ed: Springer Berlin Heidelberg, 2014, pp. 330-338.

[10] M. T. Ibrahim, R. J. Anthony, T. Eymann, A. Taleb-Bendiab, and L. Gruenwald, "Exploring adaptation & self-adaptation in autonomic computing systems," in *Database and Expert Systems Applications, 2006. DEXA'06. 17th International Workshop on*, 2006, pp. 129-138.

[11] P. Norvig and D. Cohn, "Adaptive software, 1998," *URL http://norvig. com/adapaper-pcai. html*.

[12] P. Oreizy, M. M. Gorlick, R. N. Taylor, D. Heimhigner, G. Johnson, N. Medvidovic*, et al.*, "An architecture-based approach to self-adaptive software," *Intelligent Systems and their Applications, IEEE,* vol. 14, pp. 54-62, 1999.

[13] A. Evesti and E. Ovaska, "Comparison of Adaptive Information Security Approaches," *ISRN Artificial Intelligence,* vol. 2013, 2013.

[14] M. C. Huebscher and J. A. McCann, "A survey of autonomic computing—degrees, models, and applications," *ACM Computing Surveys (CSUR),* vol. 40, p. 7, 2008.

[15] R. Bruni, A. Corradini, F. Gadducci, A. L. Lafuente, and A. Vandin, "A conceptual framework for adaptation," in *Fundamental Approaches to Software Engineering*, ed: Springer, 2012, pp. 240-254.

[16] B. Jacob, R. Lanyon-Hogg, D. K. Nadgir, and A. F. Yassin, "A practical guide to the IBM autonomic computing toolkit," ed: IBM, International Technical Support Organisation, 2004.

[17] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer,* vol. 36, pp. 41-50, 2003.

[18] J. Kephart, J. Kephart, D. Chess, C. Boutilier, R. Das, J. O. Kephart*, et al.*, "An architectural blueprint for autonomic computing," *IEEE internet computing,* vol. 18, 2007.

[19] M. M. Kokar, K. Baclawski, and Y. A. Eracar, "Control theory-based foundations of self-controlling software," *Intelligent Systems and their Applications, IEEE,* vol. 14, pp. 37-45, 1999.
[20] R. Boutaba and I. Aib, "Policy-based management: A historical perspective," *Journal of Network and Systems Management,* vol. 15, pp. 447-480, 2007.
[21] G. S. Graham and P. J. Denning, "Protection: principles and practice," in *Proceedings of the May 16-18, 1972, spring joint computer conference*, 1972, pp. 417-429.
[22] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," DTIC Document1973.
[23] K. J. Biba, "Integrity considerations for secure computer systems," DTIC Document1977.
[24] D. D. Clark and D. R. WilsonH, "A Comparison of Commercial and Military Computer Se¢ uritY POli¢ ies," 1987.
[25] D. F. Brewer and M. J. Nash, "The chinese wall security policy," in *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*, 1989, pp. 206-214.
[26] J. O. Kephart and W. E. Walsh, "An artificial intelligence perspective on autonomic computing policies," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, 2004, pp. 3-12.
[27] D. Estrin, "Inter-organisation networks: implications of access control: requirements for interconnection protocol," in *ACM SIGCOMM Computer Communication Review*, 1986, pp. 254-264.
[28] D. Robinson and M. Sloman, "Domains: a new approach to distributed system management," in *Distributed Computing Systems in the 1990s, 1988. Proceedings., Workshop on the Future Trends of*, 1988, pp. 154-163.
[29] D. Robinson and M. Sloman, "Domain-based access control for distributed computing systems," *Software Engineering Journal,* vol. 3, pp. 161-170, 1988.
[30] B. Moore, E. Ellesson, J. Strassner, and A. Westerinen, "RFC 3060: Policy core information model–version 1 specification," *IETF, February,* 2001.
[31] J. Strassner, E. Ellesson, and B. Moore, "Policy framework core information model," *IETF Policy WG, Internet Draft,* 1999.
[32] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," in *Policies for Distributed Systems and Networks*, ed: Springer, 2001, pp. 18-38.
[33] L. Kagal, "Rei," 2002.
[34] Y. Zou, T. Finin, and H. Chen, "F-OWL: An Inference Engine for Semantic Web," in *Formal Approaches to Agent-Based Systems*. vol. 3228, M. Hinchey, J. Rash, W. Truszkowski, and C. Rouff, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 238-248.
[35] A. Uszok, J. Bradshaw, R. Jeffers, N. Suri, P. Hayes, M. Breedy, *et al.*, "KAoS policy and domain services: Toward a description-logic approach to policy representation, deconfliction, and enforcement," in *Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003. IEEE 4th International Workshop on*, 2003, pp. 93-96.
[36] A. Uszok, J. M. Bradshaw, M. Johnson, R. Jeffers, A. Tate, J. Dalton, *et al.*, "KAoS policy management for semantic web services," *Intelligent Systems, IEEE,* vol. 19, pp. 32-41, 2004.
[37] T. Phan, J. Han, J.-G. Schneider, T. Ebringer, and T. Rogers, "A survey of policy-based management approaches for service oriented systems," in *Software Engineering, 2008. ASWEC 2008. 19th Australian Conference on*, 2008, pp. 392-401.
[38] Y. Zhang, Y. Zhang, and W. Wang, "Policy Engineering for Security Management of Organisation Information Systems," in *LANOMS*, 2005, pp. 289-294.
[39] A. Westerinen, J. Schnizlein, J. Strassner, M. Scherling, B. Quinn, S. Herzog, *et al.*, "Terminology for policy-based management," RFC 3198, November2001.
[40] J. D. Moffett and M. S. Sloman, "Policy hierarchies for distributed systems management," *Selected Areas in Communications, IEEE Journal on,* vol. 11, pp. 1404-1414, 1993.
[41] M. S. Beigi, S. Calo, and D. Verma, "Policy transformation techniques in policy-based systems management," in *Policies for Distributed Systems and Networks, 2004. POLICY 2004. Proceedings. Fifth IEEE International Workshop on*, 2004, pp. 13-22.
[42] S. Linying, D. W. Chadwick, A. Basden, and J. A. Cunningham, "Automated decomposition of access control policies," in *Policies for Distributed Systems and Networks, 2005. Sixth IEEE International Workshop on*, 2005, pp. 3-13.
[43] K. Barrett, J. Strassner, S. van der Meer, W. Donnelly, B. Jennings, and S. Davy, "A policy representation format domain ontology for policy transformation," in *2nd IEEE International Workshop on Modelling Autonomic Communications Environments (MACE2007), San Jose, CA, USA*, 2007, p. 34.

[44] Z. Bubnicki, *Modern control theory*: Springer, 2005.
[45] S. Haykin, *Neural networks: a comprehensive foundation*: Prentice Hall PTR, 1994.
[46] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, and G. Pavlou, "A methodological approach toward the refinement problem in policy-based management systems," *Communications Magazine, IEEE,* vol. 44, pp. 60-68, 2006.
[47] K. Barrett, S. Davy, J. Strassner, B. Jennings, S. van der Meer, and W. Donnelly, "A model based approach for policy tool generation and policy analysis," in *Global Information Infrastructure Symposium, 2007. GIIS 2007. First International*, 2007, pp. 99-105.
[48] R. Craven, J. Lobo, E. Lupu, A. Russo, and M. Sloman, "Decomposition techniques for policy refinement," in *Network and Service Management (CNSM), 2010 International Conference on*, 2010, pp. 72-79.
[49] R. Kowalski and M. Sergot, "A logic-based calculus of events," in *Foundations of knowledge base management*, ed: Springer, 1989, pp. 23-55.
[50] P. Kumari and A. Pretschner, "Model-based usage control policy derivation," in *Engineering Secure Software and Systems*, ed: Springer, 2013, pp. 58-74.
[51] P. Kumari and A. Pretschner, "Automated Translation of End User Policies for Usage Control Enforcement," in *Data and Applications Security and Privacy XXIX*. vol. 9149, P. Samarati, Ed., ed: Springer International Publishing, 2015, pp. 250-258.
[52] M. Hilty, A. Pretschner, D. Basin, C. Schaefer, and T. Walter, "A Policy Language for Distributed Usage Control," in *Computer Security – ESORICS 2007*. vol. 4734, J. Biskup and J. López, Eds., ed: Springer Berlin Heidelberg, 2007, pp. 531-546.
[53] A. Pretschner, E. Lovat, and M. Büchler, "Representation-Independent Data Usage Control," in *Data Privacy Management and Autonomous Spontaneus Security*. vol. 7122, J. Garcia-Alfaro, G. Navarro-Arribas, N. Cuppens-Boulahia, and S. de Capitani di Vimercati, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 122-140.
[54] L. Lymberopoulos, E. Lupu, and M. Sloman, "An adaptive policy-based framework for network services management," *Journal of Network and systems Management,* vol. 11, pp. 277-303, 2003.
[55] N. Samaan and A. Karmouch, "An automated policy-based management framework for differentiated communication systems," *Selected Areas in Communications, IEEE Journal on,* vol. 23, pp. 2236-2247, 2005.
[56] C. Frenzel, S. Lohmuller, and L. C. Schmelz, "Dynamic, context-specific SON management driven by operator objectives," in *Network Operations and Management Symposium (NOMS), 2014 IEEE*, 2014, pp. 1-8.
[57] A. Computing, "An architectural blueprint for autonomic computing," *IBM White Paper,* 2006.
[58] F. B. Schneider, "Enforceable security policies," *ACM Trans. Inf. Syst. Secur.,* vol. 3, pp. 30-50, 2000.
[59] A. A. Hassan and W. M. Bahgat, "A framework for translating a high level security policy into low level security mechanisms," in *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, 2009, pp. 504-511.
[60] "An Artificial Intelligence Perspective on Autonomic Computing Policies," presented at the Proceedings of the Fifth IEEE International Workshop on Policies for Distributed Systems and Networks, 2004.
[61] L. Hua, M. Parashar, and S. Hariri, "A component-based programming model for autonomic applications," in *Autonomic Computing, 2004. Proceedings. International Conference on*, 2004, pp. 10-17.
[62] J. Loyall, D. Bakken, R. Schantz, J. Zinky, D. Karr, R. Vanegas, *et al.*, "QoS Aspect Languages and Their Runtime Integration," in *Languages, Compilers, and Run-Time Systems for Scalable Computers*. vol. 1511, D. O'Hallaron, Ed., ed: Springer Berlin Heidelberg, 1998, pp. 303-318.
[63] J. Floch, S. Hallsteinsen, E. Stav, F. Eliassen, K. Lund, and E. Gjorven, "Using architecture models for runtime adaptability," *Software, IEEE,* vol. 23, pp. 62-70, 2006.
[64] J. P. Bigus, D. A. Schlosnagle, J. R. Pilgrim, W. N. Mills Iii, and Y. Diao, "ABLE: A toolkit for building multiagent autonomic systems," *IBM Systems Journal,* vol. 41, pp. 350-371, 2002.
[65] A. Jøsang and S. Presti, "Analysing the Relationship between Risk and Trust," in *Trust Management*. vol. 2995, C. Jensen, S. Poslad, and T. Dimitrakos, Eds., ed: Springer Berlin Heidelberg, 2004, pp. 135-145.
[66] M. Becher, F. C. Freiling, J. Hoffmann, T. Holz, S. Uellenbeck, and C. Wolf, "Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices," in *Security and Privacy (SP), 2011 IEEE Symposium on*, 2011, pp. 96-111.

[67] L. Qing and G. Clark, "Mobile Security: A Look Ahead," *Security & Privacy, IEEE,* vol. 11, pp. 78-81, 2013.
[68] M. Conti, V. Nguyen, and B. Crispo, "CRePE: Context-Related Policy Enforcement for Android," in *Information Security*. vol. 6531, M. Burmester, G. Tsudik, S. Magliveras, and I. Ilić, Eds., ed: Springer Berlin Heidelberg, 2011, pp. 331-345.
[69] G. Russello, M. Conti, B. Crispo, and E. Fernandes, "MOSES: supporting operation modes on smartphones," presented at the Proceedings of the 17th ACM symposium on Access Control Models and Technologies, Newark, New Jersey, USA, 2012.
[70] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, *The Twofish encryption algorithm: a 128-bit block cipher*: John Wiley & Sons, Inc., 1999.
[71] T. Vidas, D. Votipka, and N. Christin, "All your droid are belong to us: a survey of current android attacks," presented at the Proceedings of the 5th USENIX conference on Offensive technologies, San Francisco, CA, 2011.
[72] M. Nauman, S. Khan, and X. Zhang, "Apex: extending Android permission model and enforcement with user-defined runtime constraints," presented at the Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, Beijing, China, 2010.
[73] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically rich application-centric security in Android," *Security and Communication Networks,* vol. 5, pp. 658-673, 2012.
[74] V. Rao and T. Jaeger, "Dynamic mandatory access control for multiple stakeholders," presented at the Proceedings of the 14th ACM symposium on Access control models and technologies, Stresa, Italy, 2009.
[75] A. Goode, "Managing mobile security: How are we doing?," *Network Security,* vol. 2010, pp. 12-15, 2010.
[76] OASIS, " eXtensible Access Control Markup Language (XACML) TC," ed. http://docs.oasis-open.org/xacml/3.0/xacml-3.0-core-spec-os-en.html, 2013.
[77] "Directive 95/46/EC of the European Parliament and of the Council of 24 of October 1995," *Official Journal of European Communities,* 1995.
[78] R. Turn, "Classification of personal information for privacy protection purposes," presented at the Proceedings of the June 7-10, 1976, national computer conference and exposition, New York, New York, 1976.
[79] *Data Protection Act of 1998, The Stationery Office Ltd*. London, 1998.
[80] K. McCullagh, *Data Sensitivity: Proposals for Resolving the Conundrum*, 2009.
[81] S. Al-Fedaghi, "How sensitive is your personal information?," presented at the Proceedings of the 2007 ACM symposium on Applied computing, Seoul, Korea, 2007.
[82] S. Al-Fedaghi, "Crossing Privacy, Information, and Ethics," presented at the 17th International Conference Information Resources Management Association (IRMA 2006), Washington, DC, USA, 2006.
[83] L. Klüver, R. Berloznik, W. Peissl, T. Tennøe, D. Cope, and S. Bellucci, "ICT and Privacy in Europe: Experiences from technology assessment of ICT and privacy in seven different European countries. Final report," 2006.
[84] M. Jafari, R. Safavi-Naini, and N. P. Sheppard, "Enforcing purpose of use via workflows," presented at the Proceedings of the 8th ACM workshop on Privacy in the electronic society, Chicago, Illinois, USA, 2009.
[85] M. Jafari, P. W. L. Fong, R. Safavi-Naini, K. Barker, and N. P. Sheppard, "Towards defining semantic foundations for purpose-based privacy policies," presented at the Proceedings of the first ACM conference on Data and application security and privacy, San Antonio, TX, USA, 2011.
[86] M. Petković, D. Prandi, and N. Zannone, "Purpose Control: Did You Process the Data for the Intended Purpose? Secure Data Management." vol. 6933, W. Jonker and M. Petkovic, Eds., ed: Springer Berlin / Heidelberg, 2011, pp. 145-168.
[87] K. McCullagh, "Data sensitivity: proposals for resolving the conundrum," *Journal of International Commercial Law and Technology,* vol. 2 (4), pp. 190-201, 2007.
[88] S. Simitis, "cited in Bygrave, L. (2002) Data Protection Law: Approaching its Rationale, Logic and Limits, Kluwer, 132," 1973.
[89] Y. Park, S. C. Gates, W. Teiken, and P.-C. Cheng, "An experimental study on the measurement of data sensitivity," presented at the Proceedings of the First Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, Salzburg, Austria, 2011.

[90] S. S. Al-Fedaghi, "Beyond purpose-based privacy access control," presented at the Proceedings of the eighteenth conference on Australasian database - Volume 63, Ballarat, Victoria, Australia, 2007.
[91] G. Karjoth, M. Schunter, and M. Waidner, "Platform for enterprise privacy practices: privacy-enabled management of customer data," presented at the Proceedings of the 2nd international conference on Privacy enhancing technologies, San Francisco, CA, USA, 2003.
[92] (2011, 20/01/2011). *ProjectVRM*. Available: http://cyber.law.harvard.edu/projectvrm/Main_Page
[93] (2010). *User-Managed Access Work Group (UMA WG)*. Available: http://kantarainitiative.org/confluence/display/uma/Home;jsessionid=E0073E9BC37584C8DA6D4BA9AB8B58E6
[94] (2010, 20/01). *The Case for Personal Information Empowerment: The rise of the personal data store*. Available: http://mydex.org/wp-content/uploads/2010/09/The-Case-for-Personal-Information-Empowerment-The-rise-of-the-personal-data-store-A-Mydex-White-paper-September-2010-Final-web.pdf
[95] OMG. *Business Process Modeling Notation (BPMN) Specification, Object Management Group, January 2009*. Available: http://www.omg.org/spec/BPMN/1.2/PDF
[96] OASIS, "Web Services Business Process Execution LanguageVersion 2.0 – Committe Specification. Technical report " Jan 2007.
[97] B. Kiepuszewski, A. H. M. t. Hofstede, and C. Bussler, "On Structured Workflow Modelling," presented at the Proceedings of the 12th International Conference on Advanced Information Systems Engineering, 2000.
[98] (06/08). *OECD Guidelines on the Protection of Privacy and Transborder Flows of Personal Data*. Available: http://www.oecd.org/document/18/0,3746,en_2649_34255_1815186_1_1_1_1,00&&en-USS_01DBC.html
[99] A. Sabelfeld and A. C. Myers, "Language-based information-flow security," *Selected Areas in Communications, IEEE Journal on,* vol. 21, pp. 5-19, 2003.
[100] P. Dourish, "What we talk about when we talk about context," *Personal Ubiquitous Comput.,* vol. 8, pp. 19-30, 2004.
[101] M. Deng, K. Wuyts, R. Scandariato, B. Preneel, and W. Joosen, "A privacy threat analysis framework: supporting the elicitation and fulfillment of privacy requirements," *Requirements Engineering,* vol. 16, pp. 3-32, 2011.
[102] M. Howard and D. LeBlanc, "The STRIDE Threat Model. From the Book'Writing Secure Code'," ed: Microsoft Press, 2002.
[103] M. Howard and S. Lipner, *The security development lifecycle*: O'Reilly Media, Incorporated, 2009.
[104] D. J. Solove, "A taxonomy of privacy," *University of Pennsylvania law review,* pp. 477-564, 2006.
[105] A. Pfitzmann and M. Hansen, "A terminology for talking about privacy by data minimisation: Anonymity, unlinkability, undetectability, unobservability, pseudonymity, and identity management," ed, 2010.
[106] G. Sindre and A. L. Opdahl, "Templates for misuse case description," in *Proceedings of the 7th International Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ'2001), Switzerland*, 2001.
[107] K. Srivastava, A. Awasthi, and R. C. Mittal, "A Review on Remote User Authentication Schemes Using Smart Cards," in *Quality, Reliability, Security and Robustness in Heterogeneous Networks*. vol. 115, K. Singh and A. Awasthi, Eds., ed: Springer Berlin Heidelberg, 2013, pp. 729-749.
[108] B. Hoanca and K. Mock, "Secure graphical password system for high traffic public areas," presented at the Proceedings of the 2006 symposium on Eye tracking research & applications, San Diego, California, 2006.
[109] A. Adams and M. A. Sasse, "Users are not the enemy," *Commun. ACM,* vol. 42, pp. 40-46, 1999.
[110] R. Biddle, S. Chiasson, and P. C. V. Oorschot, "Graphical passwords: Learning from the first twelve years," *ACM Comput. Surv.,* vol. 44, pp. 1-41, 2012.
[111] S. Xiaoyuan, Z. Ying, and G. S. Owen, "Graphical passwords: a survey," in *Computer Security Applications Conference, 21st Annual*, 2005, pp. 10 pp.-472.
[112] L. Catuogno and C. Galdi, "A Graphical PIN Authentication Mechanism with Applications to Smart Cards and Low-Cost Devices," in *Information Security Theory and Practices. Smart*

Devices, Convergence and Next Generation Networks. vol. 5019, J. Onieva, D. Sauveron, S. Chaumette, D. Gollmann, and K. Markantonakis, Eds., ed: Springer Berlin Heidelberg, 2008, pp. 16-35.

[113] J. Bardram, R. Kjær, and M. Pedersen, "Context-Aware User Authentication – Supporting Proximity-Based Login in Pervasive Computing," in *UbiComp 2003: Ubiquitous Computing*. vol. 2864, A. Dey, A. Schmidt, and J. McCarthy, Eds., ed: Springer Berlin Heidelberg, 2003, pp. 107-123.

[114] M. D. Corner and B. D. Noble, "Protecting applications with transient authentication," presented at the Proceedings of the 1st international conference on Mobile systems, applications and services, San Francisco, California, 2003.

[115] M. Jakobsson, E. Shi, P. Golle, and R. Chow, "Implicit authentication for mobile devices," presented at the Proceedings of the 4th USENIX conference on Hot topics in security, Montreal, Canada, 2009.

[116] E. Hayashi, S. Das, S. Amini, J. Hong, and I. Oakley, "CASA: context-aware scalable authentication," presented at the Proceedings of the Ninth Symposium on Usable Privacy and Security, Newcastle, United Kingdom, 2013.

[117] (17/08). *GOLD Challenge Response*. Available: http://www.safenet-inc.com/products/data-protection/two-factor-authentication/gold-challenge-response/

[118] F. Aloul, S. Zahidi, and W. El-Hajj, "Two factor authentication using mobile phones," in *Computer Systems and Applications, 2009. AICCSA 2009. IEEE/ACS International Conference on*, 2009, pp. 641-644.

[119] L. Lamport, "Password authentication with insecure communication," *Commun. ACM,* vol. 24, pp. 770-772, 1981.

[120] B. Dodson, D. Sengupta, D. Boneh, and M. Lam, "Secure, Consumer-Friendly Web Authentication and Payments with a Phone," in *Mobile Computing, Applications, and Services*. vol. 76, M. Gris and G. Yang, Eds., ed: Springer Berlin Heidelberg, 2012, pp. 17-38.

[121] M. Gianluigi, D. Pirro, and R. Sarrecchia, "A mobile based approach to strong authentication on Web," in *Computing in the Global Information Technology, 2006. ICCGI '06. International Multi-Conference on*, 2006, pp. 67-67.

[122] H. Wen-Bin and L. Jenq-Shiou, "Design of a time and location based One-Time Password authentication scheme," in *Wireless Communications and Mobile Computing Conference (IWCMC), 2011 7th International*, 2011, pp. 201-206.

[123] C. A. Soare, *Internet Banking Two-Factor Authentication using Smartphones* vol. 4, 2012.

[124] M. H. Eldefrawy, M. K. Khan, K. Alghathbar, T.-H. Kim, and H. Elkamchouchi, "Mobile one-time passwords: two-factor authentication using mobile phones," *Security and Communication Networks,* vol. 5, pp. 508-516, 2012.

[125] IETF, "PKCS #5: Password-Based Cryptography Specification Version 2.0," ed, 2000.

[126] C. E. Shannon, "Communication theory of secrecy systems," *Bell Syst. Tech. J., vol. 28, pp.656 -715 1949,* 1949.

[127] J. Daemen and V. Rijmen, "AES Proposal: Rijndael," *submitted to the Advanced Encryption Standard (AES) contest,* 1998.

[128] NIST, "ADVANCED ENCRYPTION STANDARD (AES)," ed: FIPS PUBS, 2001.

[129] B. Schneier, "Description of a new variable-length key, 64-bit block cipher (Blowfish)," in *Fast Software Encryption*. vol. 809, R. Anderson, Ed., ed: Springer Berlin Heidelberg, 1994, pp. 191-204.

[130] E. Barkan and E. Biham, "In How Many Ways Can You Write Rijndael?," in *Advances in Cryptology — ASIACRYPT 2002*. vol. 2501, Y. Zheng, Ed., ed: Springer Berlin Heidelberg, 2002, pp. 160-175.

[131] E. Barkan and E. Biham, "The book of Rijndaels," *Cryptology ePrint Archive, Report 2002/158 (2002), http://eprint.iacr.org/2002/158*.

[132] H. Raddum, "More Dual Rijndaels," in *Advanced Encryption Standard – AES*. vol. 3373, H. Dobbertin, V. Rijmen, and A. Sowa, Eds., ed: Springer Berlin Heidelberg, 2005, pp. 142-147.

[133] N. M. Villegas, H. A. Müller, G. Tamura, L. Duchien, and R. Casallas, "A framework for evaluating quality-driven self-adaptive software systems," in *Proceedings of the 6th international symposium on Software engineering for adaptive and self-managing systems*, 2011, pp. 80-89.

[134] G. Tamura, N. M. Villegas, H. A. Müller, J. P. Sousa, B. Becker, G. Karsai, *et al.*, "Towards practical runtime verification and validation of self-adaptive software systems," in *Software Engineering for Self-Adaptive Systems II*, ed: Springer, 2013, pp. 108-132.

[135] C. Wang and W. A. Wulf, "Towards a framework for security measurement," in *20th National Information Systems Security Conference, Baltimore, MD*, 1997, pp. 522-533.
[136] M. Barbacci, M. H. Klein, T. A. Longstaff, and C. B. Weinstock, "Quality Attributes," DTIC Document1995.