

Imperial College London
Department of Computing

A Data Protection Architecture for Derived Data Control in Partially Disconnected Networks

Enrico Scalavino

2011

Supervised by Dr. Emil C. Lupu

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy in Computing of Imperial College London
and the Diploma of Imperial College London

Declaration

I herewith certify that all material in this dissertation which is not my own work has been properly acknowledged.

Enrico Scalavino

Abstract

Every organisation needs to exchange and disseminate data constantly amongst its employees, members, customers and partners. Disseminated data is often sensitive or confidential and access to it should be restricted to authorised recipients. Several enterprise rights management (ERM) systems and data protection solutions have been proposed by both academia and industry to enable *usage control* on disseminated data, i.e. to allow data originators to retain control over whom accesses their information, under which circumstances, and how it is used. This is often obtained by means of cryptographic techniques and thus by disseminating encrypted data that only trustworthy recipients can decrypt. Most of these solutions assume data recipients are connected to the network and able to contact remote policy evaluation authorities that can evaluate usage control policies and issue decryption keys. This assumption oversimplifies the problem by neglecting situations where connectivity is not available, as often happens in crisis management scenarios. In such situations, recipients may not be able to access the information they have received. Also, while using data, recipients and their applications can create new derived information, either by aggregating data from several sources or transforming the original data's content or format. Existing solutions mostly neglect this problem and do not allow originators to retain control over this *derived data* despite the fact that it may be more sensitive or valuable than the data originally disseminated.

In this thesis we propose an ERM architecture that caters for both derived data control and usage control in partially disconnected networks. We propose the use of a novel *policy lattice model* based on information flow and mandatory access control. Sets of policies controlling the usage of data can be specified and ordered in a lattice according to the level of protection they provide. At the same time, their association with specific data objects is mandated by rules (content verification procedures) defined in a data sharing agreement (DSA) stipulated amongst the organisations sharing information. When data is transformed, the new policies associated with it are automatically determined depending on the transformation used and the policies currently associated with the input data. The solution we propose takes into account transformations that can both increase or reduce the sensitivity of information, thus giving originators a flexible means to control their data and its derivations.

When data must be disseminated in disconnected environments, the movement of users and the *ad hoc* con-

nections they establish can be exploited to distribute information. To allow users to decrypt disseminated data without contacting remote evaluation authorities, we integrate our architecture with a mechanism for authority devolution, so that users moving in the disconnected area can be granted the right to evaluate policies and issue decryption keys. This allows recipients to contact any nearby user that is also a policy evaluation authority to obtain decryption keys. The mechanism has been shown to be efficient so that timely access to data is possible despite the lack of connectivity. Prototypes of the proposed solutions that protect XML documents have been developed. A realistic crisis management scenario has been used to show both the flexibility of the presented approach for derived data control and the efficiency of the authority devolution solution when handling data dissemination in simulated partially disconnected networks.

While existing systems do not offer any means to control derived data and only offer partial solutions to the problem of lack of connectivity (e.g. by caching decryption keys), we have defined a set of solutions that help data originators faced with the shortcomings of current proposals to control their data in innovative, problem-oriented ways.

Contents

Abstract	3
1 Introduction	14
1.1 Motivations	14
1.2 The Challenges	16
1.3 Application Scenario	22
1.4 Contribution	25
1.5 Thesis Outline	27
1.6 Publications	28
2 Background and Related Work	29
2.1 Chapter Overview	29
2.2 Security Requirements	30
2.3 Mandatory Access Control, Information Flow and Multi-Level Security	31
2.3.1 The Decentralised Label Model	34
2.3.2 The Asbestos Operating System	35
2.4 Access Control Policies	39
2.4.1 XML Markup Authorisation Languages	41
2.4.2 Access Control Policies for Hierarchical Domains	46
2.4.3 Trust management languages	55
2.5 Enforcement Platforms for Data Control	62
2.5.1 Non-Interactive Data Sharing Architectures	63
2.5.2 Interactive Data Sharing Architectures	66
2.6 The Trusted Platform Initiative	70
2.6.1 Trusted Computing Applications	74
2.7 Summary and Conclusions	76

3	Architecture	77
3.1	Chapter Overview	77
3.2	Threat Model	77
3.2.1	Protection Objects	77
3.2.2	Adversary	78
3.2.3	Threats	79
3.3	Data Model	81
3.4	Architecture Overview	82
3.4.1	Data Sharing Agreement Architecture	83
3.4.2	Data Publication	86
3.4.3	Policy Evaluation and Enforcement	86
3.5	Trusted Computing Integration for Policy-Driven Remote Attestation	89
3.6	Summary and Conclusions	91
4	A Data Labelling System for Derived Data Control	93
4.1	Chapter Overview	93
4.2	A Notation and Semantics for Policy Languages	94
4.3	Sensitivity Domains and Data Labelling	98
4.4	Policy Lattices	101
4.4.1	Policy Specification	101
4.4.2	Policy Ratification	104
4.4.3	Lattice Derivation	105
4.4.4	Policy Decision	107
4.5	Data Labelling	108
4.5.1	Content Verification Procedures	108
4.5.2	Transformation Labelling	109
4.6	Secure transitions	113
4.7	Derived Data Protection Architecture	115
4.7.1	Data Sharing Agreement	116
4.7.2	Data Publication	117
4.7.3	Policy Evaluation and Enforcement	122
4.7.4	Data Transformation	123

4.7.5	Components	128
4.8	Evaluation	129
4.8.1	Disseminated Data	129
4.8.2	Sensitivity Domains	132
4.8.3	Data Transformations	135
4.8.4	A Comparison with Related Work	140
4.9	Implementation Aspects	141
4.10	Summary and Conclusions	141
5	Policy-based Authority Evaluation Scheme (PAES)	143
5.1	Chapter Overview	143
5.2	The Policy-based Authority Evaluation Scheme	144
5.3	Architecture	147
5.3.1	Data Sharing Agreement	148
5.3.2	Connected Environments	149
5.3.3	Partially Disconnected Environments	156
5.4	Evaluation	171
5.4.1	Mobility Model	171
5.4.2	Results and Evaluation	172
5.5	Implementation Aspects	181
5.6	Summary and Conclusions	182
6	Conclusions	183
6.1	Summary of Thesis	183
6.2	Contributions	184
6.3	Operational Assumptions	186
6.4	Caveat	188
6.5	Future Work	189
	Bibliography	191

List of Tables

1.1	Main requirements for crisis management scenario	25
2.1	Asbestos label sensitivity levels and common usage	36
2.2	Asbestos discretionary labels: data senders can use these labels to enforce discretionary control on data exchanges	38
2.3	Examples of Asbestos data exchanges	39
4.1	Gathered and disseminated information	130
4.2	Transformed and disseminated information	131
4.3	Compound Boolean expressions for the crisis management scenario	134
5.1	Example authority table	161
5.2	Average key delivery times (seconds)	173
5.3	Average data delivery times (seconds)	173

List of Figures

1.1	Sharing data between organisations	16
1.2	Data sharing in an example mobile environment	23
2.1	Control specification models and enforcement architectures	29
2.2	An example of security lattice with two security domains. Both dashed and solid lines represent dominance relationships	32
2.3	An example Asbestos security lattice with two security domains	36
2.4	XrML's core concepts and example <i>license</i>	41
2.5	ODRL's core concepts and example offer	42
2.6	EPAL's core concepts and example	43
2.7	XACML data-flow diagram. The numbers express temporal ordering amongst the exchanged messages	44
2.8	XACML core concepts and example policy	45
2.9	Policy evaluation process for hierarchical domains	47
2.10	Author-X example policy	49
2.11	Authorisation example: registration key and content of DNASignature necessary to access analysis elements	50
2.12	Evaluation points for authorisations in Ponder2	53
2.13	Ponder2 example authorisation policies	53
2.14	Conflicting policies in Ponder2	54
2.15	Example TPL policy	57
2.16	DSA architecture scheme	63
2.17	Non-Interactive Data Sharing Architecture	63
2.18	Example access hierarchy and associated keys	64
2.19	Interactive Data Sharing Architecture	66

2.20	Microsoft RMS data protection	68
2.21	High Level ERM Architecture based on the sticky policies paradigm	70
2.22	Integrity measurement process during trusted boot	73
2.23	Attestation process	73
2.24	The PEI models framework	75
3.1	Attack tree for data protection architecture. Shaded attacks are dealt with by mainstream products. Non-shaded attacks are dealt with natively by our proposed architecture	79
3.2	High level architecture of the proposed data protection solution. Arrows represent logical dependencies and not actual exchanged messages	82
3.3	Data Sharing Agreement for the crisis management scenario	84
3.4	High-level interaction for data publication	86
3.5	Interaction of components for data publication	87
3.6	High-level interaction for policy evaluation	87
3.7	Interaction of components for policy evaluation	88
3.8	High level architecture for policy-driven remote attestation	90
4.1	Function representing the policy evaluation process	96
4.2	Mutable information sensitivity for different domains	99
4.3	Mapping from lattice of data labels to lattice of policy sets	100
4.4	Interaction of components for data publication	118
4.5	An example XML structure labelled by CVPs prior to level propagation	119
4.6	Examples of content verification procedure	119
4.7	Data labelling procedure	119
4.8	An example XML structure labelled by CVPs after level propagation	121
4.9	Protected data package	122
4.10	Data access procedure	123
4.11	Example XQuery transformation	124
4.12	Interaction of components for data transformation	125
4.13	Data transformation procedure	126
4.14	High level architecture for derived data control	128
4.15	Example role hierarchies	130
4.16	Example Content Verification Procedures	135

4.17	Toxic contamination estimate transformation	136
4.18	Toxic contamination assessment transformation	136
4.19	Emergency document transformation	137
4.20	Images blurring transformation.	138
4.21	Victims count transformation	138
4.22	Statement update transformation	139
4.23	Hospital assignment transformation.	139
4.24	The hospital assignment document	140
5.1	Example policy graph	146
5.2	Example policy graph for a connected scenario	150
5.3	Protected data package	153
5.4	High level architecture integrated with PAES for use in connected environments	154
5.5	Interaction of components for policy evaluation using PAES in connected environments	155
5.6	Policy evaluation recursion	156
5.7	Data exchange in a crisis scenario	158
5.8	An example PAES hierarchy for crisis management	159
5.9	Policy graph with multiple strict evaluation nodes	162
5.10	Opportunistic dissemination of authority keys	165
5.11	Interaction of components for policy evaluation in disconnected environments	166
5.12	High level architecture for derived data protection in partially disconnected environments	168
5.13	Mobility model of rescuers in the emergency area	173
5.14	Key delivery and data delivery ratio (30 rescuers, 10m communication range)	174
5.15	Key delivery and data delivery ratio (30 rescuers, 20m communication range)	175
5.16	Key delivery and data delivery ratio (30 rescuers, 30m communication range)	175
5.17	Key delivery and data delivery ratio (75 rescuers, 10m communication range)	176
5.18	Key delivery and data delivery ratio (75 rescuers, 20m communication range)	176
5.19	Key delivery and data delivery ratio (75 rescuers, 30m communication range)	177
5.20	Key delivery and data delivery ratio (150 rescuers, 10m communication range)	177
5.21	Key delivery and data delivery ratio (150 rescuers, 20m communication range)	178
5.22	Key delivery and data delivery ratio (150 rescuers, 30m communication range)	178
5.23	Average key delivery times and data delivery times with different parameters	179

5.24 A simpler policy graph	180
5.25 Key delivery and data delivery ratio with a simpler policy graph	181

Acknowledgments

I would like to express my gratitude to my supervisor, Dr. Emil C. Lupu, whose expertise and understanding were invaluable for this thesis and whose guidance allowed me to arrive at the end of this journey.

I also thank everyone who contributed to the fantastic environment I was so lucky to work in. All the colleagues who shared the office with me and helped me make it look less like a working place. Leonardo, Changyu and Domenico for leading me astray with our late evening challenges and awkward discussions, and Giovanni for taking me back on track. Daniel for laughing with me at everyone else. Alberto for forcing me to drink as every good PhD student should. Srdjan for being a constant reminder that life (and I) could be much worse. Rudi for his unforgettable Cape Town stories told in front of a chicken burger. Anandha for the only sport debates I could have in London, Vaibhav for the constant help and everyone else who worked at Imperial College with me during its latter happy italian years. Outside of the imperial walls, I wish to thank Franco, for reminding me of how young I still am, and all of my friends back in Italy, for being there whenever I leave this island for sunnier lands.

A very special thank goes to my fiancéé Valentina, who most of all had to bear the worst effects the PhD years had on my brain. Finally, I must thank my family for the love and support they have always given me. I would not be here if it were not for them.

1 Introduction

1.1 Motivations

Information management is one of the most important aspects of an organisation's life that can often make the difference between success and failure. While conducting their business, organisations and individuals produce, share and process vast amounts of information. Information on customers, partners, competitors, products, services, business and research activities is gathered, created anew or obtained from third parties. Gathering or creating new information is a very expensive activity that requires organisations to invest substantial resources. Information is thus considered an important asset that must be leveraged by widely disseminating it both internally within the originating organisation and externally with clients and collaborators. In large multinational organisations, data frequently crosses administrative borders and is even subject to different legislative constraints. Inter-organisation partnerships and joint projects are based on the assumption that the cooperating partners are willing to share their data. However, shared and disseminated information must be protected from unauthorised access or use because of its intrinsic commercial or personal value. Unintended disclosures may not only undermine marketing and commercial plans or reveal confidential information about products and research activities, but might also have a cost in terms of public credibility. The need to control data after its acquisition or transmission does not solely arise from the confidentiality of data but also from an increasing amount of legislation regarding the handling of private and medical data (such as HIPAA in the USA [JD96], and the data protection act in the UK [act98]), the freedom of information availability [foi00, foi66], data retention requirements for auditing [sox02]. For example, the latest european directives on data protection [eud12] will force companies to pay a great deal of attention to data protection practices with threats of severe penalties in the form of fines or bans to operate in the abiding countries. Privacy thus represents a large and increasing enterprise risk for companies to take into consideration as data subjects and their personal identifiable information are protected by law. Third-party *controllers*, usually parts of government bodies, monitor the organisations and companies to discover, stop and potentially punish misconducts in personal data handling. Such rules must thus be regarded as important as any marketing, commercial or research strategy in any organisation's operations.

Moreover, the foremost purpose of many inter-organisational cooperations is the creation of new intellectual property (IP) or simply of new and more useful information through the sharing and processing of existing data. The newly produced information is often more valuable or sensitive than the data that was originally shared and its protection is thus a major concern. This is why copyright laws such as the copyright and patent laws in the USA [pat10] and the copyright, designs and patents act in the UK [pat88], exist in most jurisdictions. Sometimes, the newly produced information may be less valuable, or its ownership may shift from one entity to another one (for example, aggregated private information is owned by the aggregating company, not by the data subjects). Motivations for data protection also come from:

- the military environment, where people and national security often depend on data confidentiality;
- crisis situations (e.g. police, firefighters and first-aid services), where unprotected data dissemination could cause public chaos, slow down rescuers and affect efficacy of the rescue operations;
- public research environments, where data is public but its usage must be audited or limited in time;
- digital markets, where resources are sold and licensed and cannot be freely redistributed.

Data loss prevention is necessary and organisations cannot simply rely on post-breach procedures (such as insurances and legal actions) to make up for the damages caused by data leaks. While the former reduces the risk of unauthorised data disclosures, the benefits of the latter alone may be surpassed by their negative side effects: bad publicity, expensive, long and uncertain legal actions, jeopardised relationships with third parties. The two approaches to data security are not mutually exclusive, but rather make up for each other's lacks. Similarly, permissive approaches based on risk mitigation strategies such as user training and strict data handling practices (e.g. based on data classification and users' clearance, possibly specified in a legal contract) can be coupled with automatic preventions strategies and post-breach procedures to build a safe operational envelope and achieve the desired result. Moreover, building a safe but limited operational environment is not alone sufficient, especially when data must travel beyond the organisational boundaries. The operational environment to be consider must thus include all organisations sharing data and solutions to audit each other's policies and behaviours and verify their correctness must be put in place.

Note that although we have referred to *data*, the issue of protecting exchanged items is generic and applies to any digital object including multimedia resources and software. We will therefore refer to it generically as *disseminated data* or *information* and will occasionally use *digital resource* and *object* interchangeably where no ambiguity arises. We will also use the term *data originators* (or simply *originators*) to indicate all entities

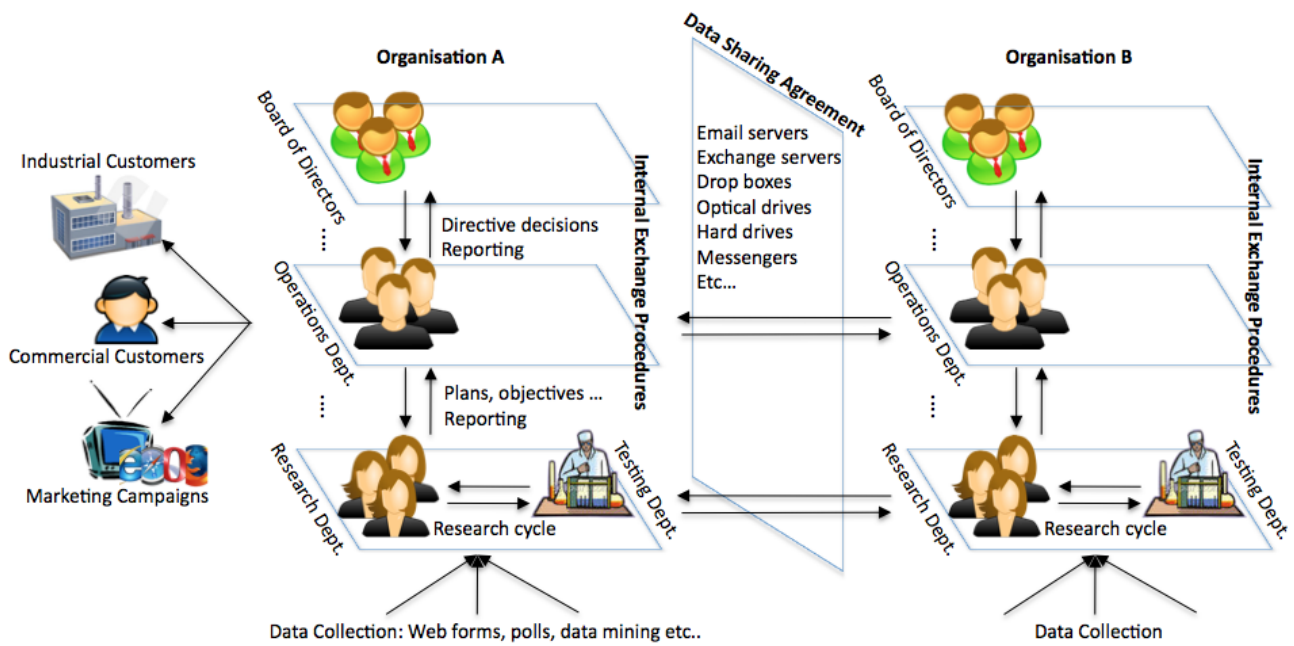


Figure 1.1: Sharing data between organisations

that create or first disseminate data and the term *data recipients* (or simply *recipients*) to indicate all entities that receive or consume disseminated data.

1.2 The Challenges

Data protection is a challenging task because it must be applied regardless of the data's physical location, even after information has crossed the organisational boundaries and has been received by both known parties part of a supply chain and unknown parties. Normally, once data has been sent to a recipient, no automated controls can be imposed on the usage of that data and further dissemination cannot be prevented. This is a significant obstacle that may hinder the sharing of data perceived as confidential, private or commercially sensitive. Data exchange and usage should be governed in fact by many parameters including: the nature of the data and its sensitivity, the data recipients (and their trustworthiness), the "purpose" for which the data is to be used, the context in which the data is used and the potential benefits to be obtained from exchanging the data. However, information either flows freely without restrictions or is not shared at all. Partners mutually rely on each other not to disclose the shared data with external and malicious parties and at the same time attempt to gain confidence about their mutual trustworthiness, e.g. by auditing their operations. Service level agreements may be put in place to guarantee the amount, type and quality of the data exchanged, but these usually do not include details on the handling of sensitive and protected information. Data governance in large companies is often based on internal procedures that employees are expected to comply with and on a range of risk-mitigating measures such as

employee training, while external data dissemination is governed by mutually agreed arrangements regarding the handling of shared information (Figure 1.1). The latter are known as *data sharing agreements (DSAs)* and define at a high level of abstraction the procedures for handling data including access rights, disclosure and notification obligations and constraints on usage that must be respected by the participants to the agreement. The term “data usage” here encompasses a broad set of situations and controls including data lifetime, data retention or destruction, creation of derived data and onward dissemination that data originators need to control based on changing circumstances and conditions. While DSAs are in use today, they are mostly specified in natural language and thus not linked to an automated policy enforcement system. They are in essence legal documents whose enforcement is currently only possible through the courts. The definition of the agreement itself is per se a very problematic phase in the relationship between the organisations that have to come to terms with their different and, more often than not, conflicting requirements. For example, when public research institutions work with national security and military agencies, they must find a tradeoff between the aim to publicly disseminate their knowledge and the need to preserve secrecy. Whenever economic interests are at stake or mismanagement of data could cause a great deal of damage (economical, physical or psychological) to communities or individuals, relying on human interpretation of the law or on informal agreements may prove hazardous. Rather than venturing on expensive and long legal disputes that often cast shadows on the corporate image or risking dangerous data leakages, preventative and automatic controls on data usage would be an ideal solution.

The problem of controlling data usage by unknown recipients was formulated as early as 1989 in the form of the originator control ORCON/ORGCON family of models [Gra89] and later studied by Abrams [AHK⁺91, Abr93] and others. A taxonomy of models was presented under the name of *dissemination control (DCON)* [TS04], and more recently a family of models for *usage control (UCON)* has been presented in [PS04]. UCON is more general in that it is not only concerned with the retention of control by the data originator but also attempts to cater for permissions that change during the usage of the data or digital resource. In this work we will use the term *UCON* to refer in a generic way to usage control models, *DCON* to refer to the sub-category of UCON models where the dissemination of the resource is the only usage that needs to be controlled (e.g. to block the distribution of copyrighted material or to encourage the distribution of publicly available information) and *UCON_{ABC}* to refer to the models proposed by Park and Sandhu [PS04].

Scenarios where usage control is an issue are often based on orthogonal assumptions and address antithetical requirements. For example, key factors are:

- the amount of mutual knowledge entities share (what do originators know about recipients?);

- the control over recipients' devices and communication channels (can recipients modify the software/hardware configurations of their devices?);
- the number of different organisational domains among which data is exchanged (is data disseminated in just one organisation or are there several partners involved?);
- the nature of the connections between entities (are devices connected through *ad hoc* networks, is there an infrastructure or are they completely or partially disconnected?);
- the architecture that can be deployed (centralised or decentralised);
- the “purpose” for which the data is shared. Note that despite its relevance, the usage “purpose” is often not considered by data protection systems because it is practically impossible to determine the reason for which data is used. Simply knowing who is using data (as in *access control and policy systems*, Section 2.4) and how (as in *usage control systems* [PS04]) provides no indication on their real intentions. This is why “purpose-based” usage control is outside the scope of this thesis.

As a consequence of such variety the problem has been confronted, either directly or indirectly, by research studies in different areas of computer science from which this work draws inspiration, ideas and techniques:

- **Access control**, including mandatory access control (MAC) and discretionary access control (DAC), focuses on the verification of the access rights users have when requesting access to digital resources [SdV00]. Resources are usually protected by making decisions over their dissemination (and thus before they are sent to recipients) on the basis of the requesting users' clearance (MAC) or permissions (DAC). By Sandhu's definition [SP03] UCON can be considered a “*a generalization of access control to cover obligations, conditions, continuity (ongoing controls) and mutability*”.
- **Digital rights management (DRM)** focuses on the distribution of copyrighted material where access to and usage of resources is usually subject to some form of payment [fai07, win11]. Protected resources are intended to be used by a restricted set of applications that can be controlled through software generally referred to as *virtual machines (VM)* or *reference monitors (RM)*. DRM technologies focus on the enforcement of a small subset of the controls UCON aims to allow such as payment obligations, temporal limits on usage and bounded re-dissemination of resources.
- **Enterprise rights management (ERM)** focuses on the distribution of commercially sensitive information within and between cooperating organisations. Protected resources include, among others, marketing plans, research data and results, market research and statistics, technical know-how, financial analysis,

reporting and so on. Access to and usage of such resources can be subject to a wide range of different conditions depending on the particular scenario and data type. Typically, protected resources are intended to be used in a controlled environment that includes not only software and hardware but also physical location. For example, recipients may be limited to accessing the data only on the devices provided and managed by their employer, while physically being located in the organisation's premises. ERM differs from DRM in different ways. First, DRM restricts data access to its paying owner and forbids modifications and re-dissemination, while ERM protects data throughout its entire life cycle, which may include collaborative changes to its content. Second, DRM protection is usually integrated within a specific data format (e.g. DRM protection for blu-ray discs) or device (e.g. Apple's FairPlay on iPods), while ERM protection is tied to the user application accessing the data (e.g. Microsoft Office or Adobe's Acrobat). ERM systems can be seen as commercial implementations [liq06, mar09, Mic05] of UCON aimed at corporate environments.

- **Privacy** focuses on the handling of information related to individuals. In this thesis we will refer to the term "privacy" as synonymous of "confidentiality" or "secrecy" of information belonging to individuals and their capacity of retaining control over it [ML00, ZJMW05, MPB03]. In this sense, privacy protection overlaps with UCON in many respects. Specific issues addressed within this category include identifying and restricting access to personally identifiable information based on the consent of the data subject and data protection legislation. In particular, usage consent is problematic as legislations [act98, JD96, eud12] usually protect the right of data subjects to revoke it. When such right is exerted, it is hard to determine to which extent the ownership of the data subject over its data affects the ownership of the results that have been obtained so far by using the data. An example is the creation of aggregated data that collects information from different subjects. The owner of such aggregate is usually the company that invested money to create it, and the original data subjects should have no rights over it. Even so, organisations are forced to treat such matters carefully, as legislations such as the latest european directives on data protection [eud12] threat them with severe penalties in case of misconduct. However, because of these reasons, we emphasise that the main motivations behind this research are drawn from environments where personal data privacy is something to deal with to protect an organisation in face of the existing legislation, but is not a driving factor for its operations.
- **Document protection frameworks** focus on the distribution of documents whose disclosure is restricted by means of cryptographic algorithms and protocols [BCF01, MS03, Cra04]. Documents are processed and encrypted under a number of keys which can be distributed amongst recipients. The usage con-

trol problem in this case is reduced to a key management problem where keys are distributed to users according to varying criteria.

- **Policy-based management** focuses on the declarative definition of rules constraining system behaviour. By Sloman's definition, policies are *rules governing the choices in behaviour of a system* [Slo94]. In this sense, policies are the primary means by which UCON rules can be expressed over disseminated data [xac05, Gua01, Ian02, JSSS01].
- **Distributed trust management** focuses on the declarative definition of rules driving the creation of trust relationships between entities (users, processes acting on behalf of users, automated processes etc.) that have no prior knowledge of each other. The term *trust* indicates the belief that an entity will behave as expected in the context of a specific interaction and, in our case, of a specific usage request. Some trust management solutions [BFG10, BFL96] have been explicitly designed to express and evaluate authorisation policies. Similarly, trust management could also be used as a means to establish trust relationships amongst data recipients and originators in UCON.

Drawing from the research areas described above, we investigated the main limitations of existing solutions for data protection. In fact, while current data protection and DRM/ERM architectures vary in aspects such as policy/rights deployment and data protection mechanisms, common design principles, functionality and flaws can be identified. First, data is associated with access rights (or usage control policies) and cryptographically protected before distribution (i.e. the data is "published"). Then a central trusted authority (TA), often the originator himself, responds to access requests, evaluates recipients' credentials and rights and issues the decryption keys. A trusted component running on the recipient devices, which we will refer to as the *virtual machine/monitor (VM)* or simply *client*, ensures that rights are locally enforced. Specific architectures differ in their management of user authentication, policy and rights retrieval, audit of user actions and other tasks [Aut05, Mic05, MPB03, PSS00, Sec07, PHB06]. Although existing solutions cater for different requirements and scenarios, they all suffer from a few major hurdles that limit their usability:

- **Issue 1: derived data protection.** First, the existing solutions' common perspective considers only passive recipients that access information and never transform it. Recipients instead often transform the received data by adding or removing content. When data is transformed its protection requirements change accordingly and the previously applied policies can become either too restrictive or too loose for the new content. Also, information that is directly derived from protected data should inherit part of the original data's protection requirements, so as not to infringe the original owner's rights, should they have any. Existing ERM systems either ignore the problem or offer only partial solutions. Most of the time

users are forced to republish the data with new policies, which has several limitations. First, it assumes that data is modified only by its owner or by a user who is trusted to republish the data correctly. Second, the effort required from users to keep policies up-to-date with the data content may exceed the effort required to modify the data itself. Finally, modifying policies is an error-prone activity that may lead to data leakages. We call this problem the “*derived data problem*” and it is one of the main focuses of this thesis, together with the problem of policy evaluation in disconnected scenarios.

- **Issue 2: trusted authority selection.** Second, the set of trusted authorities (TA) is statically and explicitly defined. Changes, such as adding a new TA or removing an existing one, require defining new agreements, establishing new trust relationships and implementing them in terms of key exchanges, trust records etc. Moreover, the assumption that recipients are willing to present their credentials to the authority designated by the data originator, even when credentials may be private or confidential, is not always realistic. Untrusted TAs may misuse the credentials by disclosing them publicly or may not perform fair policy evaluations. Finally, in organisational environments where partners do not know the internal structure of the organisation with whom they interact it is difficult for the originator to specify access rules based on internal roles of the recipient or to specify which entities in the partner organisation are entitled to issue or verify specific credentials. Instead, recipients are typically better placed to choose which authority could better evaluate their credentials amongst a scope of acceptable authorities defined by the originator. Note that the problems classified under this issue group do not directly affect data protection but are weaknesses of the trust assumptions made by existing protection systems. The technology solution we propose here is complementary to other controls that can be used to increase confidence in the behaviour of the entities involved in the data exchange (e.g. post-usage evaluations based on audit trails, user complaints and reports etc).
- **Issue 3: data protection in disconnected networks.** The last issue concerns the availability of the TA: if recipients cannot contact the specified TA, they cannot access the data. This problem is particularly relevant when data must be accessed in (partially) disconnected environments, as it is often the case for military or rescue missions. Encryption systems such as attribute-based encryption (ABE) [BF01, SW05, Cha07, BSW07, OSW07] and access hierarchies [AFB05] that allow policies or access rights to be evaluated offline are not a viable solution as they require all recipients to receive credentials and encryption/decryption keys from the same authority before data is disseminated, even if they belong to different organisations. Moreover, these solutions prevent context dependent conditions to be evaluated, as they provide decryption keys only on the basis of statically defined credentials.

In this thesis we aim to propose solutions to the above problems. We thus aim to enable *derived data control* and *usage control in partially disconnected networks*.

1.3 Application Scenario

The main problem civil protection agencies and organisations have to face while conducting their daily activity is to predict when either disastrous events or small scale accidents will happen. Even after a crisis has exploded and rescuers have already intervened on the scene, it still is difficult to predict how the events will unfold and how human settlements, facilities, natural resources or even people's lives will be threatened or affected. The unpredictability of the events at hand and the potential damage they can cause, also in terms of human lives, make crisis management an extremely delicate task. When facing an emergency, decision times must be extremely quick so that the proper actions can be taken to alleviate the current threat. Being able to read the actual situation and its development in all its detail is thus of the utmost importance to guarantee the success of rescue missions. This is no easy feat, especially since most of the time it requires the organisations and companies involved to share all the information that they are aware of. Data availability is thus extremely important. The main obstacles in this sense are the reluctance of the parties to share data, motivated by its sensitive, confidential nature, and the hectic development of events which often hinders or plainly prevents communications between rescuers. Moreover, third parties are often unaware of the crisis management procedures and data management practices followed by civil protection agencies and join the operations with no prior notice.

The need for timely access to disseminated data and the possibility of disrupted communications push to the extreme the traditional requirements for data protection and usage control. Traditional ERM systems assume "friendly" scenarios where no delay incurs in the data exchange or data protection processes, where all possible recipients and originators are known in advance and where all machines are under the administrative control of a specific organisation. These assumptions are not realistic in crisis management, and this is why we chose it as the main application scenario to be used throughout this thesis. In particular, the specific scenario we present here, and further develop in the following chapters, is based on the actual practices regulating data sharing and protection during crisis management operations carried out by public safety organisations in the United Kingdom and studied by *BAE systems plc* [GBLD]. The scenario will provide a context for examples and simulations throughout the rest of this thesis and will be used to illustrate the details and aspects of our approaches.

Consider a traffic accident that rapidly escalates into a threat to the larger surrounding area. Figure 1.2 depicts the scene. A car has been crushed between a petrol tanker and a chemical tanker in the middle of

a tunnel. An explosion at the time of the impact caused thick black smoke to fill the tunnel, while escaped witnesses state that one of the tankers is leaking a “suspicious” liquid. From the street it is possible to see a chemical spilling out of the tunnel and seeping into a nearby sewer, while several injured persons have come out of the tunnel with breathing problems. Nearby vegetation, close to a small residential district, also catches fire. Police, fire brigades and ambulance services immediately arrive on the scene with both vehicles and mobile stations and deploy in the area so-called *bronze or mobile command centres (BCCs)*. The “threat area” they consider is a perimeter of 100m radius centred on the tunnel. Despite this being only an hypothetical scenario, similar situations have happened in the past. For example in 1999 the tragedy of the Mont Blanc Tunnel caused the death of 39 people when a truck caught fire in the tunnel. The rescue operations lasted for 53 hours during which cooperation amongst different agencies belonging to different countries was fundamental. Similarly, the rescue operations for the London underground bombings of 2005, where 52 people died in explosions in underground tunnels, were significantly hindered by the communication difficulties encountered by the rescuers.

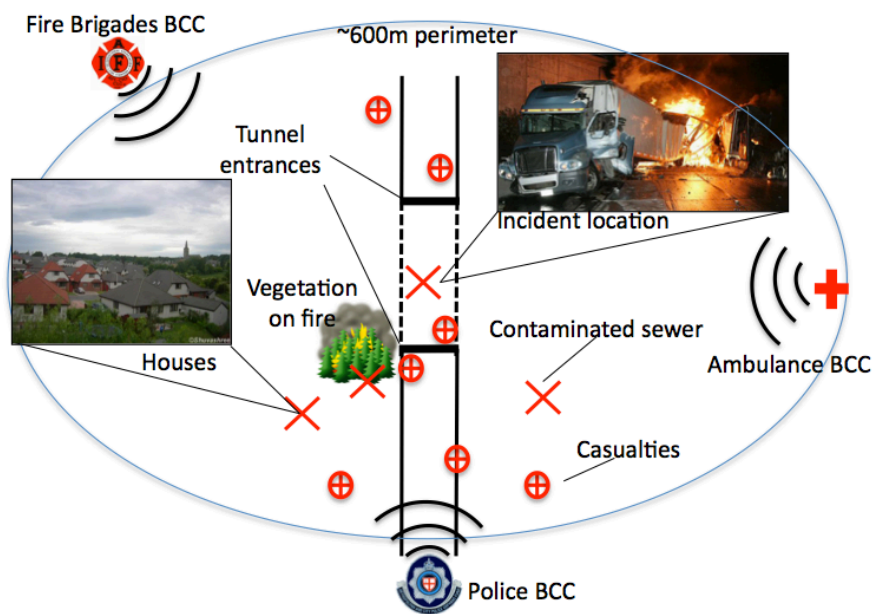


Figure 1.2: Data sharing in an example mobile environment

Examples of information the rescuers need to exchange and that must be protected are:

- personal and medical information of victims;
- the tunnel and sewer plans and information on nearby gas pipelines, gas storages and electrical facilities;
- information on the tankers' content; 4) the map of the surrounding area with information on the houses to be evacuated;

- information on the state of the accident and of the rescue operations.

Also, gathered information may be combined or transformed by BCCs to produce new data that should be protected depending on its new content (e.g. a list of residents in the area combined with their clinical history).

In addition to first responders, many other organisations are called to share data on the accident: the service providers, the local council, an environmental agency and the tankers' transport companies. Subsequent responders are heterogeneous: while some may have agreements with first responders and be aware of the procedures to follow (e.g. they may have stipulated a DSA with the rescue organisations) others may be unaware of the regulations involving data sharing and protection. Examples are private companies whose facilities or vehicles have been involved in the accident and that have no predefined agreement with the rescue organisations, despite the confidentiality of the data they can and must provide. In other words, while some organisations may have pre-agreed DSAs, others may not.

In this context, rescuers' timely access to data represents the basis for a prompt and effective rescue plan, while a failure to share information is a source of later embarrassment and can have a high cost even in terms of human lives. Data confidentiality must nevertheless be protected for several reasons. First, victims' privacy is governed by legislation such as HIPAA in the USA. Second, information on the accident may cause panic while burglars may target houses that have been evacuated and criminals may use information on the local facilities and utilities for future crimes. Third, leaked information revealing a possible mismanagement of the operations may be used to embarrass the involved agencies and individuals. Finally, organisations may merge information coming from different sources to gain a broader insight into the development of the situation. If for example, information from the fire brigade, the company owning the chemical tanker and the ambulance services were used by one of the rescuing organisations to deduce that there is a toxic contamination in the accident area, the new information should be restricted to avoid panic but rescuers should be able to access it so that they can react to possible threats. In other words, derived data control is also an important aspect in crisis management. On the other hand, data must still be accessible to all authorised rescuers not only for reading but for editing as well. Note that the term "authorised user" includes a wide range of usage controls, such as but not limited to identity, role, location and time verification, credentials checks (e.g. to verify users' qualifications to handle highly technical data), limitations on the duration of the access or interruption of the access in case of contextual changes (e.g. data becomes more sensitive as a consequence of the evolving situation), obligatory actions to be performed by the accessing user etc.

In this scenario, the lack of long-range connectivity is a very likely circumstance. The communication infrastructure could have been disrupted by the incident itself (e.g. if the explosion damaged a local antenna) or could have never been present in the tunnel. In this sense, incidents in particular environments such as

Requirements

- 1: Sharing of protected data with rescuers and participating organisations on the base of existing DSAs
- 2: Dissemination of protected derived data (i.e. that cannot be accessed by unauthorised recipients) with no additional delays
- 3: Timely access to data with no connectivity

Table 1.1: Main requirements for crisis management scenario

underground tunnels or caves are generally very difficult to deal with. Also, the communication network may have been shut down for security reasons, or public panic may overload the existing infrastructure making it unusable for emergency services.

Table 1.1 summarises the main security requirements of the presented crisis management scenario. First, responders must be able to exchange and access data with any involved organisation, according to the regulations specified in existing DSAs. Second, transformed data should be re-disseminated in the emergency area with no further delay caused by decisions over the protection to be applied over it. Derived data should be automatically protected upon creation. Finally, data must be accessible even when connectivity is not available.

1.4 Contribution

The principal contribution of this thesis is to present a data protection architecture that can be deployed in partially disconnected environments (such as the one described in the above scenario) and that caters for the derivation of usage/access control policies for derived data. In making this contribution, a number of significant results have been achieved, which can be summarised as follows:

- **Contribution 1 - An Information Flow System for derived data protection:** from our perspective, protecting derived data means determining the usage control policies applicable to the new information depending on the original information used and the derivation process itself. To achieve this, we have developed a novel policy model that uses information flow labels [BL73, Den76] and partially orders policies depending on the level of protection they provide. We thus generate a *lattice of usage control policies* that can be used to label and classify data and enables us to reason about the protection level to be associated with derived information. This is presented in Chapter 4 and addresses the first issue we identified in Section 1.2 and the second requirement of Table 1.1. Derived data is protected automatically (i.e. policies are created automatically and associated with the data) upon creation and with no further delays, so that it can be re-disseminated immediately in the emergency area.

- Contribution 2 - PAES: a Policy-Based Authority Evaluation Scheme for data protection:** In order to allow recipients to decide which authorities they actually trust with their credentials, we designed a policy evaluation scheme based on the idea that authority to evaluate policies can be granted by other policies. In addition to specifying the criteria recipients must meet to access the data, PAES also permits data originators to specify the criteria that authorities must meet to be trusted to evaluate policies. In contrast to existing models, the set of authorities is therefore dynamic and defined by characterisation. This provides increased flexibility in finding authorities mutually trusted by both data originators and recipients. This is presented in Chapter 5, in Sections 5.2 and 5.3.2 and addresses the second issue we identified in Section 1.2. In fact, not only recipients can select their most trusted (to both access their credentials and correctly evaluate policies) evaluation authority amongst a set of acceptable ones, but if an authority is unreachable for any reason, a different one can be chosen easily from those within reach.
- Contribution 3 - An adaptation of the PAES scheme (Mobile-PAES) for partially disconnected environments:** When network connectivity is absent, the movements of users can be leveraged to carry the information to whomever needs it. Networks that leverage node mobility and *ad hoc* connections to disseminate messages are known as opportunistic networks (Oppnets) or delay tolerant networks (DTNs) [RHB⁺07, VB00, CG07]. However, data protection schemes for such networks [LKB⁺06, SK05] are predominantly intended for protecting communications between a specific sender-recipient pair, i.e. to protect data packets while they are stored and forwarded by intermediate nodes. This conflicts with the very definition of data dissemination. We propose a solution to integrate traditional protection mechanisms for disseminated data in oppnets. In particular we adapt the original PAES design for use in oppnets. The aim is to increase the probability that rescuers can meet and contact a trusted authority by means of an *ad hoc* connection, and thus receive the needed authorisations. This third contribution is presented in Chapter 5, Section 5.3.3 and addresses the third issue we identified in Section 1.2 and the third requirement of Table 1.1. Rescuers acting in an emergency area with no connectivity will be able to share and access protected data in a timely fashion.

Throughout the thesis, we will also show how the architecture we propose allows organisations in an emergency area to share data according to the regulations specified in different DSAs, thus satisfying the first requirement of Table 1.1. Whilst the contributions of this thesis facilitate secure data sharing, they are complementary to the traditional controls put in place in such circumstances such as data classification activities and users' clearance management, user training on security awareness and risks, activity auditing, insurances, contractual obligations and so on, which aim to mitigate the risk of users misusing the disseminated data, either intention-

ally or unintentionally. The operational context within which policy-based data protection can be successful must thus include a set of controls that mitigates the risk of human error, discourages malicious behaviour and reduces the damages caused by policy breaches.

1.5 Thesis Outline

The remainder of this thesis is organised as follows:

Chapter 2 - Background and Related Work: In this chapter we introduce the existing technologies for data protection, thus setting the scene for the contributions presented in this thesis. We first present access control models and policy systems. We then present the software architectures proposed for the enforcement of usage control and data protection. The last part of the chapter gives an overview of the software/hardware mechanisms that have been proposed as part of the *Trusted Platform Initiative* [Gro, ELM⁺03, Sch03] and gives an idea of how they could fit into the solutions we propose.

Chapter 3 - Architecture: This chapter sets the ground for the subsequent description of our proposed solution. First, it introduces the threat model which is the basis of this thesis. It then proceeds describing the high-level components that constitute the architecture of our solution and their interactions. Finally, it details how we integrated our proposed architecture with trusted computing technologies. This integration in particular has been made possible by the work of Dr. Anandha Gopalan [GGSL11].

Chapter 4 - A Data Labelling System for Derived Data Control: This chapter is devoted to the presentation of the data labelling mechanism we deploy to derive usage control policies for derived data. An extensive example based on the crisis management scenario is provided to evaluate the efficacy and flexibility of the proposed solution.

Chapter 5 - Policy-Based Authority Evaluation Scheme (PAES): This chapter describes the PAES policy model and the two possible uses we devised. We first show how PAES can be used in fully connected scenarios to address some of the problems of traditional ERM systems (the second issue described in Section 1.2). We then describe how PAES can be used to enable usage control in partially disconnected environments and crisis management scenarios in particular. The chapter also provides a description of the simulations we performed to evaluate the proposed protocol.

Chapter 6 - Conclusions: In the last chapter we summarise the work done, underlying the contributions and limitations of our solutions and discussing possible future directions for research.

1.6 Publications

While a journal paper is still in the writing and will be submitted for the *ACM Transactions on Information and System Security (TISSEC)*, some of the ideas described in this thesis have already been published in several venues:

- A. Gopalan, V. Gowadia, E. Scalavino, E. C. Lupu. “*Policy driven Remote Attestation.*” In Proceedings of the 3th ICST Conference on Security and Privacy in Mobile Information and Communication Systems, Aalborg, Denmark, ICST, May 2011.

- V. Gowadia, E. Scalavino, E. C. Lupu, D. Starostin, and A. Orlov. “*Secure Cross-domain Data Sharing Architecture for Crisis Management.*” In Proceedings of the 10th ACM workshop on Digital rights management, Chicago, IL, US, ACM, October 2010.

- E. Scalavino, V. Gowadia, R. Ball, E. C. Lupu, G. Russello. “*Mobile PAES: Demonstrating Authority Devolution for Policy Evaluation in Crisis Management Scenarios.*” In Proceedings of the 11th IEEE Int. Symposium on Policies for Distributed Systems and Networks, George Mason University, VA, US, IEEE, July 2010.

- G. Russello, E. Scalavino, N. Dulay, E. C. Lupu. “*Coordinating Data Usage Control in Loosely-Connected Network.*” In Proceedings of the 11th IEEE Int. Symposium on Policies for Distributed Systems and Networks, George Mason University, VA, US, IEEE, July 2010.

- E. Scalavino, G. Russello, R. Ball, V. Gowadia, E. C. Lupu. “*An opportunistic authority evaluation scheme for data security in crisis management scenarios.*” In Proceedings of the 5th ACM Symposium on Information, Computer and Communication Security, Beijing, China, ACM, April 2010.

- E. Scalavino, V. Gowadia, E. C. Lupu. “*A Labelling System for Derived Data Control.*” In Proceedings of the 24th IFIP Working Conference on Data and Applications Security and Privacy, Rome, Italy, IFIP, June 2010.

- G. Russello, E. Scalavino. “*Exploiting Node Mobility for Coordinating Data Usage in Crisis Scenarios.*” In Proceedings of the 4th IFIP Workshop in Information Security Theory and Practices, Passau, Germany, IFIP, April 2010.

- E. Scalavino, V. Gowadia, E. C. Lupu. “*PAES: Policy-Based Authority Evaluation Scheme.*” In Proceedings of the 23rd IFIP Working Conference on Data and Applications Security and Privacy, Montreal, Canada, IFIP, July 2009.

2 Background and Related Work

2.1 Chapter Overview

Protecting data and controlling its usage when it is disseminated amongst unknown recipients has been gaining increasing attention in recent years. Military environments have traditionally been the main contributors (either directly or indirectly) to the research in the data protection field, but with the widespread diffusion of personal and mobile devices among the public the problem of data protection has attracted commercial and public interest and efforts [CW87]. The data available and exchanged both online and offline now includes private and commercial data. The number of ways users create, protect and access data has also increased accordingly. In this chapter we present background material on data protection solutions, their limitations and the requirements and challenges that must still be addressed.

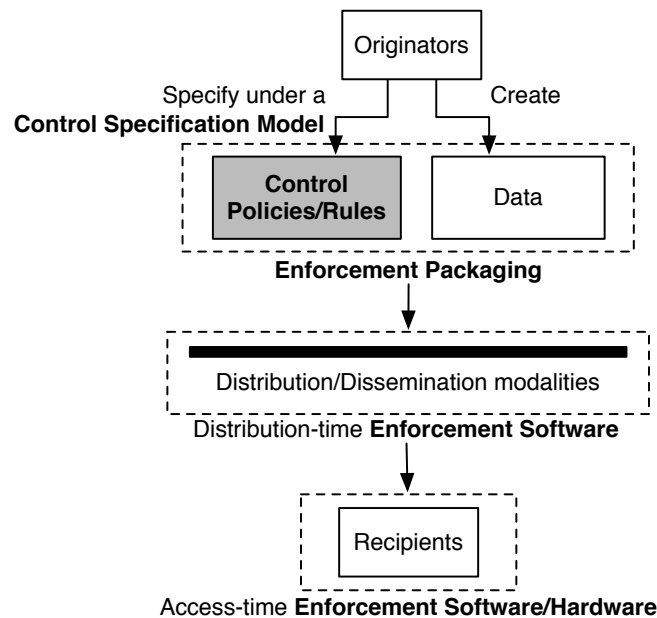


Figure 2.1: Control specification models and enforcement architectures

When dealing with information security (as opposed to system security), we must first distinguish between the two main components that constitute a data protection solution (Figure 2.1): *control specification models*,

i.e. what usage and data controls are specified, and *enforcement architectures*, i.e. the mechanisms deployed to ensure controls are enforced. Such mechanisms usually adopt different combinations of the same high-level components. Data control can in fact be enforced by modifying the disseminated data and enclosing it into a *secured packaging* [PMK99, cry97], by controlling whether recipients are authorised to receive the data before it is sent to them, or by controlling the recipients' devices so that they may receive the data but not be able to access it [fai07, Mic05].

In the first part of this chapter we briefly introduce the mandatory access control and multi-level security (MAC and MLS, Section 2.3) models, from which part of this thesis draws. We then present the more recent solutions based on policy specifications (PBAC, Section 2.4) that extend the basic discretionary access control (DAC) and role-based access control (RBAC) models. While this thesis draws from all of these works, our aim is to abstract from existing paradigms and build on top of them a solution to address the problems of derived data control and usage control in disconnected environments. The second part of this chapter is devoted to describe the existing enforcement architectures, distinguishing between interactive data sharing architectures (IDSA, Section 2.5.2) for connected environments and non-interactive data sharing architectures (NIDSA, Section 2.5.1) for disconnected environments. Particular attention is drawn to the limitations of the proposed solutions, so that we can frame the description of our unified data sharing architecture (UDSA). Finally, we discuss the trusted platform initiative (Section 2.6), i.e. the combination of hardware and software mechanisms that aim to allow users to verify whether a (possibly remote) application can be trusted or not.

2.2 Security Requirements

Traditionally, information security requirements have been classified under three categories: confidentiality, integrity and availability [Dra06, SdV00].

Confidentiality is defined by the International Organisation for Standardisation (ISO) as “ensuring that information is accessible only to those authorised to have access” [IOfS] and is required whenever improper disclosures of data may result in losses not only in terms of money but also in terms of public credibility and public safety. Examples are the disclosure of industrial or military secrets and private documents. The usage control paradigm [PS04] expands upon this concept and aims to allow originators to control not only *who can access data and under what conditions*, but also *how long* the access can last, *what conditions must be verified throughout the whole duration of access (continuous control)* and *what provisional actions must be performed before, during and after the access*.

Integrity is defined as “protecting the accuracy and completeness of information and the methods that are

used to process and manage it” [IOfS]. Integrity is required whenever accessible but incorrect data may be detrimental to the organisation’s operations. Examples are incorrect medical histories in healthcare scenarios or false activity logs in military environments. The usage control paradigm aims to allow originators to control data modifications, and thus data integrity, in the same way that data accesses are controlled.

Finally, **availability** is defined as “ensuring that data is accessible and usable when needed by an authorised entity” [IOfS]. Examples where data availability is pivotal include crisis management scenarios where the efficiency of the rescuers’ coordinated action greatly depends on the information they share. ERM systems assume the existence of an open *dissemination environment* across which protected data can be freely disseminated with no restrictions. While in fact everyone can receive the data, encryption mechanisms prevent unauthorised recipients from accessing it. Availability could also be improved by forcing recipients to re-disseminate data upon reception [LY06, PS02]. However, example cases where availability is at risk exist such as the crisis management scenario presented in Section 1.3. Whenever communication amongst the potential participants to the data sharing is not guaranteed, data may not be available to authorised users when they need it most, either because it has not been received or because it cannot be decrypted. In this thesis we explicitly consider this problem and propose a solution for data dissemination and protection in partially disconnected networks.

2.3 Mandatory Access Control, Information Flow and Multi-Level Security

In mandatory access control (MAC) [SdV00] access to information is predefined by administrative policies that are mandated by general rules that are valid throughout the system and that are related to the sensitivity of the protected information. As a consequence MAC policies do not consider the concept of *user ownership* over data and users have no control over the data they disseminate. Adopted for a long time mostly in military environments, MAC policies have been largely developed as *multi-level security* or *lattice-based information flow policies* for both confidentiality and integrity protection [BL73, San93, Bib77].

Denning et al. [DD77] first defined information flow control as “*methods of regulating the dissemination of information among objects throughout the system*”. While this broad definition encompasses several problems and their solutions, information flow has been most often associated with *multilevel security* systems (MLS) [BL73]. A *multilevel security policy* consists of the classification of information and subjects (i.e. processes that can access information on behalf of users) into *security* (or *sensitivity*) *classes*, of a *flow relation* defining permissible flows amongst these classes and of a method of binding data objects to some class. Any operation, or series of operations, that uses the information contained in some object A to derive the content of another object B, is said to cause an *information flow* from A to B. This flow, and thus the operations that caused it, is

permissible only on the basis of the defined multilevel security policy that organises classes into a *lattice* (i.e. a partial order in which any two elements have a unique least upper bound and a unique greatest lower bound).

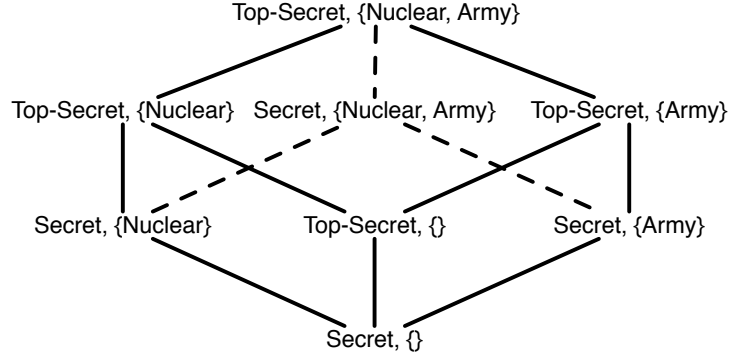


Figure 2.2: An example of security lattice with two security domains. Both dashed and solid lines represent dominance relationships

All MLS systems aim to achieve some variant of the *simple security property* and **-property* of the Bell-LaPadula (BLP) model [Den76, BL73]. The lattice of sensitivity classes is generated by the Cartesian product of two other lattices: a total order of sensitivity levels such as *unclassified*, *classified*, *secret* and *top-secret* and a partial order over the subset relationship $\subseteq$, where the ordered elements are sets of data categories (or security domains), such as *nuclear*, *army* etc. Sensitivity classes are thus (sensitivity level, category set) pairs that can be associated either with data as *security labels* (or data classification labels, indicating the sensitivity level of the data for the corresponding domains, i.e. the potential damage an unauthorised data leak may cause) or with subjects as *clearance labels* (indicating the classes of data subjects are authorised to access). Given the set of all labels LS and the set of all domains DS , a partial order is defined via the label dominance relation $\sqsubseteq$:

$$\forall L_a, L_b \in LS : L_a \sqsubseteq L_b \iff s(L_a) \leq s(L_b) \wedge c(L_a) \subseteq c(L_b) \quad (2.1)$$

where $L_a \sqsubseteq L_b$ indicates that label L_b dominates label L_a and $s(L_a)$ and $c(L_a)$ are the sensitivity level and the set of domains of label L_a , respectively. Figure 2.2 depicts an example security lattice. Non-secure information flows are prevented by forbidding subjects with a given clearance L_s from accessing data with an higher classification L_d . The *simple security property* and **-property* can then be stated as:

- **simple security property:** For every subject s and data object d , s can access d if and only if $L_d \sqsubseteq L_s$.
- ***-property** For every subject s and data object d , s can modify d if and only if $L_s \sqsubseteq L_d$.

that actually realise a no-read-up, no-write-down access control policy for confidentiality protection. Other MLS systems aim instead to achieve some variant of the properties of the Biba model [Bib77] that proposes

a dual no-write-up, no-read-down access control policy for integrity protection. In the original paper, Biba further extended his model with the *low-water mark model* that included two alternative policies:

- **Low-water mark for subjects:** No restrictions are imposed on read operations. A subject s accessing object o has her classification downgraded to the greatest lower bound of her and o 's classifications.
- **Low-water mark for objects:** No restrictions are imposed on write operations. An object o written by a subject s has its classification downgraded to the greatest lower bound of its and s 's classifications.

Despite trying to enforce a more dynamic behaviour where classifications can change, the low-water mark models have two main drawbacks. The low-water mark policy for subjects constrains the privileges that subjects have to perform certain actions on the order the actions themselves are performed. The low-water mark policy for objects instead does not protect integrity, since subjects are allowed to write at levels beyond their own possibly causing highly sensitive information to be tainted by lowly sensitive information. The *high-water mark model* instead relates subjects' privileges with the lowest upper bound of the classifications of objects observed so far. Lipner [Lip82] also proposed to merge the Bell-LaPadula and Biba models adopting independent confidentiality and integrity classes into a single lattice.

The Bell-LaPadula and Denning's information flow models have been widely criticised for two main reasons. The first problem is that they failed to capture the effects that changing the classification of data objects while accessing them have on data protection [McL90]. The problem was addressed informally by enunciating the *tranquillity principle* [BL73], stating that the classification of objects should not be changed during normal operations. Several studies have been conducted to allow modifications of objects' classifications (i.e. to upgrade or downgrade an object's sensitivity levels). *Floating labels models* are a sub-category of information flow models including the decentralised label model [ML00] and Asbestos [EKV⁺05, VEK⁺07] where labels are copied from subjects to data (and viceversa) and from one data item to another, thus actually modifying its classification. Data declassification poses the greatest challenges as it can easily result into information leaks if misdirected. Proposals have been made to allow declassification either by explicitly authorising trusted subjects to downgrade all data objects [SS05], enabling data owners to downgrade their own data only [ML00] or including declassification annotations at programming language level [LM09]. The second problem of the Bell-LaPadula model is that mandatory access controls dictated by a security lattice are not expressive enough for real-world policies. Models have thus been proposed to merge the mandatory access control policies specified by clearance and data classification labels with discretionary access control policies. Intuitively, the discretionary policy (e.g. an access matrix or access control list) would operate within the boundaries of the mandatory policy, restricting only the accesses that would be permitted by the mandatory controls. Exam-

ples are the work proposed by Walter *et al.* [WOR⁺74] and compartmented mode workstations (CMW). The Bell-LaPadula model also included this feature in its first formulation [BL73, Bel73].

2.3.1 The Decentralised Label Model

With the **decentralised label model** Myers and Liskov [ML00] provided a language for describing authorised information flows that could be more flexible and expressive than earlier models. The essentials of their model are principals, i.e. users and authorities such as groups and roles on behalf of whom processes operate on data. For example, users and groups in Unix would be represented by principals. Principals are organised in a partial order defined over the *act for* relationship. We say that a principal p_i *acts for* a user p_j ($p_i \succeq p_j$) when p_i possesses all the privileges of p_j . A data classification label in the decentralised label model is a set of (*owner* : *readers*) pairs called *policies*, where an *owner* is a principal maintaining privileges on the policies she owns and *readers* is the list of principals that the policy allows to read the data. In other words data is labelled with a number of access control lists (i.e. lists of the principals authorised to access) controlled by different principals and can be accessed only by principals appearing in all lists (or that can *act for* at least a principal in all lists). Declassification can be performed by principals only on their owned policies and consists in adding principals to the list of readers. The model is considered “decentralised” as there is no need for a central authority managing policies, but all principals in the system are responsible for the policies they own. Note that the declassification performed by principals is not the only possible one. Access to an object can in fact be extended to other principals by assigning them to a group or role that is already authorised to access, while the policy for the object would remain unchanged.

Information flows are controlled by annotating program variables with labels and ensuring that as data flows, labels only become more restrictive (i.e. there are more owners or some owners allow fewer readers). *Input and output channels* from which data enters and exits the program are also labelled to avoid information leaks. Labels are controlled at compile-time and thus cannot be modified at run-time. Therefore, in order to guarantee confidentiality, writing a value to a variable is only permitted if the variable’s label is at least as restrictive as the value’s one. Similarly, if the content of a variable affects the content of another variable, the label of the second one must be at least as restrictive as the content of the first one. Given two labels L_i and L_j , we say that L_j is at least as restrictive as L_i ($L_i \sqsubseteq L_j$) if every policy in L_i is *covered* (i.e. is enforced) by at least a policy in L_j . A policy I is covered by a policy J ($I \sqsubseteq J$) if two conditions hold. First, the owner of policy J , $o(J)$, must be either the same as the owner of policy I or must be able to *act for* him. Second, the list of readers of policy J , $r(J)$, must be a subset of the list of readers of policy I , taking into consideration the *act for* relationship. These two conditions define the *complete relabelling rule* that indicates when relabelling from a label L_i to a label

L_j is safe, i.e. when an information flow from a value labelled as L_i (either read from a variable or an input channel) to a variable or output channel labelled as L_j is permitted. More formally:

$$\begin{aligned}
L_i \sqsubseteq L_j &\equiv \forall I \in L_i \exists J \in L_j I \sqsubseteq J \\
I \sqsubseteq J &\equiv o(J) \succeq o(I) \wedge R(J) \subseteq R(I) \\
&\equiv o(J) \succeq o(I) \wedge \forall p' \in r(J) \exists p \in r(I) p' \succeq p
\end{aligned} \tag{2.2}$$

where $R(I)$ is the set of all principals that act for at least one principal in $r(I)$:

$$R(I) = \{p \mid \exists p' \in r(I) \wedge p \succeq p'\} \tag{2.3}$$

The model provides for two types of data reclassification. The first is an *increment-only* reclassification. When a program combines two or more values, for examples multiplying the content of two variables, the result's label must be at least as restrictive as the labels of all the input values. This derived label is obtained as the *least upper bound* $L_i \sqcup L_j$ of the inputs' labels, containing the union of all the policies in L_i and L_j . Still, the new derived value can only be stored in a variable (or written to a channel) whose label is at least as restrictive as the derived one. The second type of reclassification is a *selective declassification* that allows principals to retain control over the policies they own and thus to relax overly restrictive labels. Processes acting on behalf of a given set of principals can thus relax the corresponding policies by either adding new readers or completely removing the policy (that corresponds to adding all principals as readers). Beside the label model for confidentiality protection, the authors also proposed a dual decentralised label model that protects integrity.

2.3.2 The Asbestos Operating System

Stemming from a refined version of the Bell-LaPadula model [Bel73], the Jif programming language [Mye99] and the decentralised label model [ML00], Vandebugart *et al.* [EKV⁺05, VEK⁺07] proposed **Asbestos**, an operating system to enforce application-defined information-flow security policies. Sensitivity classes in the Asbestos model are represented by sets of (security domain, security level) pairs. As in the original BLP model, sensitivity classes are associated with data as data classification labels and with subjects as clearance labels. Given the set of all labels LS and the set of all domains DS , a partial order is defined over the label dominance relation:

$$\forall L_a, L_b \in LS : L_a \sqsubseteq L_b \iff \forall t \in DS : L_a(t) \leq L_b(t) \tag{2.4}$$

In other words label L_a is less than or equal to label L_b if for each domain the level in L_a is less than or equal to the level for the same domain in L_b . If one domain's level is strictly less in L_a than in L_b and another

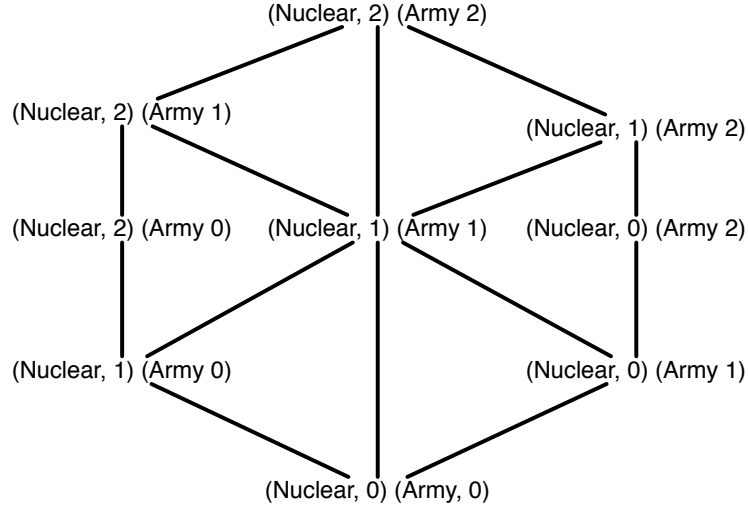


Figure 2.3: An example Asbestos security lattice with two security domains

Level	Usage
*	Privilege for the tag
0	High integrity for the tag
1	Default tracking level - no restriction for the tag
2	Default clearance level, used in taint tracking
3	High secrecy for the tag

Table 2.1: Asbestos label sensitivity levels and common usage

domain's level is strictly less in L_b than in L_a , then L_a and L_b are said to be incomparable. Figure 2.3 depicts an example security lattice in Asbestos, where for example the label (Nuclear 1, Army 1) dominates three lower-classification labels, is dominated by three higher-classification labels and is incomparable with other two. Besides its clearance label C_p , each process p is also associated with a tracking label T_p that records the sensitivity of the most sensitive data received or observed so far (similarly to a high-water mark model). The process' clearance label is thus used to bind the maximum tracking label the process can be associated with. Sensitivity levels for data classification labels range from 0 to 3. Information can flow freely from 0 to 3 (i.e. data can be received by processes, stored into files or sent through channels with higher classification) but the reverse direction, i.e. declassification, can only be obtained by processes holding a privilege for the declassified domain, where the term “privilege” refers to the possibility of a process to declassify data for a specific domain or override the restrictions imposed by the domain itself (as we will see later).

Table 2.1 summarises the available sensitivity levels and their common usage. In the rest of this thesis we will adopt Asbestos' terminology and use the terms “domain” and “tag” interchangeably. While the $*$ level in a tracking label indicates privilege for the domain, level 0 is used for integrity tracking (i.e. tracking whether

an object was modified only by high-integrity processes), level 1 is used as the default tracking level for data and processes (i.e., it represents the initial amount of sensitive information contained in data or known by processes), level 2 is used as the default clearance level for processes and level 3 is used for high-secrecy information. One of the main innovations of the Asbestos' model is that its labels combine the functions of confidentiality tracking and integrity tracking. While confidentiality is tracked by increasing a process' tracking label from 1 to the level of the most sensitive object observed and is protected by enforcing rule (2.8) (see later), integrity tracking can be achieved by assigning level 0 to the tracking label of high-integrity processes for a specific integrity tag. In this way, whenever high-integrity processes receive a message from a low-integrity process, their integrity level is increased and thus spoiled. Labels are ordered according to rule (2.4) and can be combined with the least upper bound ($\sqcup$) and greatest lower bound ($sqcap$) operators that, assuming a linear ordering of the sensitivity levels, take for each domain the highest or lowest corresponding level in either label, respectively:

$$L_a \sqcup L_b(t) = \begin{cases} L_a(t) & \text{if } L_a(t) \geq L_b(t) \\ L_b(t) & \text{otherwise} \end{cases} \quad (2.5)$$

$$L_a \sqcap L_b(t) = \begin{cases} L_a(t) & \text{if } L_a(t) \leq L_b(t) \\ L_b(t) & \text{otherwise} \end{cases} \quad (2.6)$$

Since privilege over a tag is represented by the lowest level $*$, processes would lose privilege when receiving information by non-privileged processes (with tracking label higher than $*$). Therefore, a further operator $\sqcup^*$ is introduced to allow processes to retain privilege over a tag when receiving information with higher classification:

$$L_a \sqcup^* L_b(t) = \begin{cases} * & \text{if } L_b(t) = * \\ L_a(t) & \text{otherwise} \end{cases} \quad (2.7)$$

The second innovation of the Asbestos' model is represented by *discretionary labels*, four label types that can be specified by a process when sending information and can be used as a means to enforce discretionary controls, i.e. to give more flexibility to mandatory controls dictated by the traditional information flow rules. The four discretionary labels are T^+ , T^- , C^+ and V . Their functions are described in Table 2.2, where the symbols $\perp$ and $\top$ are used to indicate the lowest and highest label in the lattice, respectively.

Asbestos also allows processes to restrict the information flows to and from specific *Inter Communication Ports (IPC)* (i.e. the channels used by processes to communicate with each other) by associating them with a

Label	Default Value	Usage
T^+	$\perp = \{*\}$	Increases a message's tracking label. Used when a message contains information at a higher sensitivity than the sending process itself (e.g. for a process with privilege).
T^-	$\top = \{3\}$	Decreases the receiving process' tracking label. Used to declassify data or grant privilege. Setting it to non-default values requires privilege.
C^+	$\perp = \{*\}$	Increases the receiving process' clearance label. Setting it to non-default values requires privilege.
V	$\top = \{3\}$	Declares an upper bound on the sender's tracking label. Used to represent the maximum sensitivity level the sender process intends to use.

Table 2.2: Asbestos discretionary labels: data senders can use these labels to enforce discretionary control on data exchanges

clearance label PC_{port} . Given two communicating processes, a sender P and a receiver Q , P can append to its message the four discretionary labels so that Q can receive the message only if the following rule is satisfied:

$$T_P \sqcup T^+ \sqsubseteq (C_Q \sqcup C^+) \sqcap PC_{port} \quad (2.8)$$

In other words, the least upper bound of the tracking label of the sender process (representing the most sensitive information it might communicate) and the T^+ discretionary label must be dominated by the least upper bound of the receiving process' clearance label and the C^+ discretionary label, constrained by the clearance of the port used for the communication. Once the communication has been permitted and the message has been sent, the receiving process's tracking label is modified according to the following rule:

$$T_Q \leftarrow ((T_P \sqcup T^+ \sqcup T_Q) \sqcap T^-) \sqcap T_Q^* \quad (2.9)$$

In other words, the message's sensitivity is considered to be the least upper bound of the sender process' tracking label, the T^+ discretionary label and the receiving process' tracking label. The T^- discretionary label is then used to declassify the message. Finally, the last part of the formula restores privilege so that the operation does not eliminate it from the receiving process' label.

Let us consider an example with two processes, a sender P and a receiver Q and two tags, *nuclear* (n) and *privacy* (p). Table 2.3 reports the results of several example message exchanges. In the first example, the message exchange fails because the tracking level for tag n of the sender process is higher than the clearance level of the receiving process. In the second example the message exchange is allowed and the tracking level for tag p of the receiver process is increased to 2. However, the receiver process' privilege for tag n is preserved. In the third example the message exchange fails because the sender's process purposely increments the message's

	T_P	T^+	T^-	C_Q	Old T_Q	New T_Q	Result
1	$\{n\ 3, p\ 1\}$	-	-	$\{n\ 2, p\ 2\}$	$\{n\ 1, p\ 1\}$	$\{n\ 1, p\ 1\}$	Failure
2	$\{n\ 2, p\ 2\}$	-	-	$\{n\ 2, p\ 2\}$	$\{n\ *, p\ 1\}$	$\{n\ *, p\ 2\}$	Success, preserved privilege
3	$\{n\ 1, p\ 1\}$	$\{n\ 3, p\ *\}$	-	$\{n\ 2, p\ 2\}$	$\{n\ 1, p\ 1\}$	$\{n\ 1, p\ 1\}$	Failure
4	$\{n\ 1, p\ 1\}$	-	$\{n\ 0, p\ 3\}$	$\{n\ 2, p\ 2\}$	$\{n\ 1, p\ 1\}$	$\{n\ 1, p\ 1\}$	Failure
5	$\{n\ *, p\ *\}$	-	$\{n\ *, p\ 1\}$	$\{n\ 3, p\ 2\}$	$\{n\ 3, p\ 2\}$	$\{n\ *, p\ 1\}$	Success, granted privilege, declassification

Table 2.3: Examples of Asbestos data exchanges

tracking level for tag n to a level higher than the receiver process' clearance level. In the fourth example the message exchange fails because the sender process tries to declassify the tracking level of the receiver process for tag n without having privilege for it. Finally, in the fifth example the sender process manages to grant privilege for tag n to the receiving process and to declassify its tracking label for tag p to level 1.

While capable of enforcing traditional information flow controls, this solution also allows sender processes to manipulate the sensitivity of messages considering the actual amount of information they contain, and allows privileged processes to grant access to other processes (i.e., to increase their clearance) in a discretionary manner. Note that, since data labels can contain a (hypothetically) infinite amount of domains and every application defines its own domains, no interference occurs between the application's protection requirements.

Both Asbestos, the decentralised label model and the existing information flow control systems are limited in the type of controls they can express. Usage control requirements are broader and require a set of controls that go beyond clearance control and access control lists. For example, there are cases when mandatory policies should be overridden in the specific context in which the data access is being requested. For instance, in emergency situations low-clearance rescuers may be authorised to access a victim's private data if his life is at risk or if not enough high-clearance rescuers are available at the scene. Also, the enforcement mechanisms put forward by both Asbestos and the decentralised label model assume that controls over user clearance are performed before the data is actually disseminated by whoever is sending it. This is not always a safe assumption, as users who are not the data owners may have no interest in controlling whether recipients should be allowed to receive the data before re-disseminating it. Despite their limitations, floating labels are necessary and will be used in our solution to the derived data problem described in Section 1.2.

2.4 Access Control Policies

Discretionary policies are rules specifying who (the *user*, *subject* or *principal* of the policy) can, or cannot, execute which actions on which resources (the *target* or *object* of the policy), and under which conditions.

Context-dependent conditions restrict authorisations on the basis of the situation where the access request is issued, for example considering the request's time or the requestor's location. *Content-based conditions* constrain authorisations on the basis of the content of the policy object. *History-based conditions* restrict authorisations depending on the accesses previously authorised (as in the Chinese Wall policy [BN89]). Finally, *obligations* are conditions that make access depend on the execution of specific actions by the requesting user. Obligations can be interpreted in several different ways. We broadly divide them into obligations on the enforcing system (the ones we will consider in the rest of this thesis), i.e. automatic procedures that are performed by the enforcement module of a policy system [DDLS01, xac05], and obligations on the user, i.e. obligations that must be performed by the user accessing the data and whose execution the system must verify [CLM⁺09]. Several declarative policy languages have been devised since the *access matrix model* to specify conditions restricting the validity of authorisations [Lam74]. Each language supports different models for representing both data and subjects. Data can be organised hierarchically (to represent a containment relationship [DDLS01], as for XML data [BCF01, BBC⁺00, DdVPS02, DdVPS00], or to represent an "is-a" relationship, as in object-oriented programming), in sets on the basis of different criteria (as for the Chinese Wall data model [BN89]) or according to the relational data model [SdV00]. Subjects can be organised in groups (e.g. in UNIX operating systems), hierarchically (e.g. in RBAC policies [SCFY96]), or be represented by certificates attesting their attributes or trust relationships [RL96, CEE⁺02, HMM⁺00].

The need to organise users and objects to avoid the burden of a per-user, per-object policy specification, has led to the problem of exception management, i.e. authorisations applicable to all members of a user/object category (being it a user's role, group or an object's class or container) but few. This is why many languages support both *positive* and *negative authorisations*, i.e. the specification of both access permission and denial rules. Traditional approaches relied instead on only one of the two authorisation types, adopting either a *closed policy* denying all accesses for which there is no specified positive authorisation, or an *open policy* permitting all accesses for which there is no specified negative authorisation. However, two problems arise when using both authorisation types. The *incompleteness* problem, occurring when no policy has been specified that applies to a given access request, and the *inconsistency* problem, arising when both positive and negative authorisations have been specified that apply to the same request (i.e. a *modality conflict* occurs). To address these problems, policy languages usually have constructs allowing the specification of a *default policy* that is applied whenever no other policies apply to a given request, and of *conflict resolution strategies* that decide which of several conflicting policies should be applied to a given request.

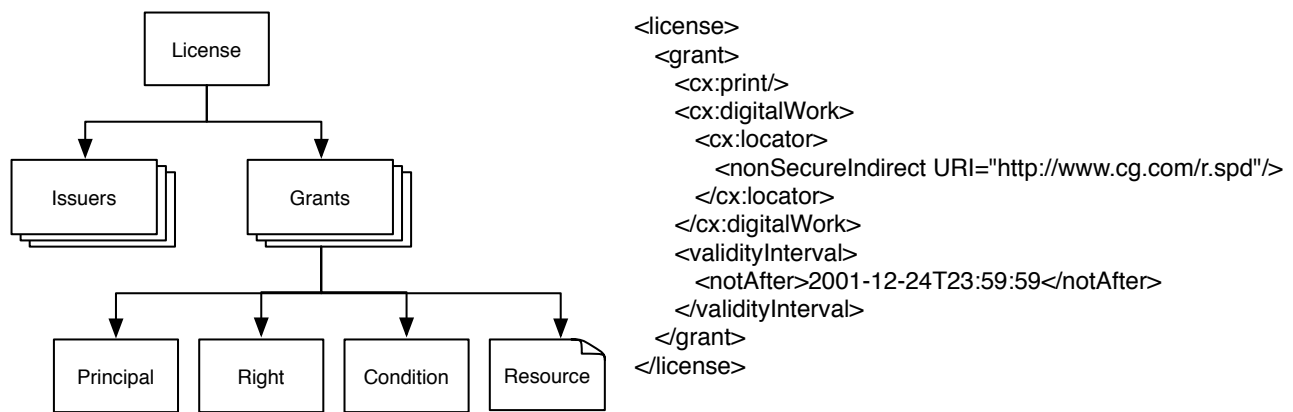


Figure 2.4: XrML's core concepts and example *license*

2.4.1 XML Markup Authorisation Languages

XML (eXtensible Markup Language) has been widely used and extended to support the specification of different access control concepts, resulting in the creation of several rights expression languages (RELs) mainly used for *digital rights management*. Although the several proposed languages differ in scope and syntax, common concepts and design principles can be identified. The **eXtensible Rights Markup Language (XrML)** [Gua01], the **Open Digital Rights Language (ODRL)** [Ian02], the **Enterprise Privacy Authorisation Language (EPAL)** [PSK⁺03] and the **eXtensible Access Control Markup Language (XACML)** [xac05] are maybe the most representative of all.

XrML

XrML is the standard rights expression language for protection of MPEG-21 digital resources. An XrML policy or *license* is a collection of *grants*, i.e. of elements through which the *license's issuers* (usually the owners of the protected resources) bestow an authorisation (a *right*) upon a principal. Both issuers and principals are specified in terms of the public/private key pairs they possess. Figure 2.4 illustrates a simplified model of the core elements of an *XrML license* and an example policy permitting the printing of an online resource before a specified time.

ODRL

ODRL is often used in place of XrML. The elements at the basis of an ODRL policy are *assets*, *rights* and *parties*. Parties include both principals and rights holders (corresponding to *XrML licence issuers*). Rights holders specify ODRL policies (*rights*) as *offers* over resources (*assets*). Whenever a principal requests and obtains access to an asset from its originator, she is issued the corresponding *license* in the form of an *agreement*,

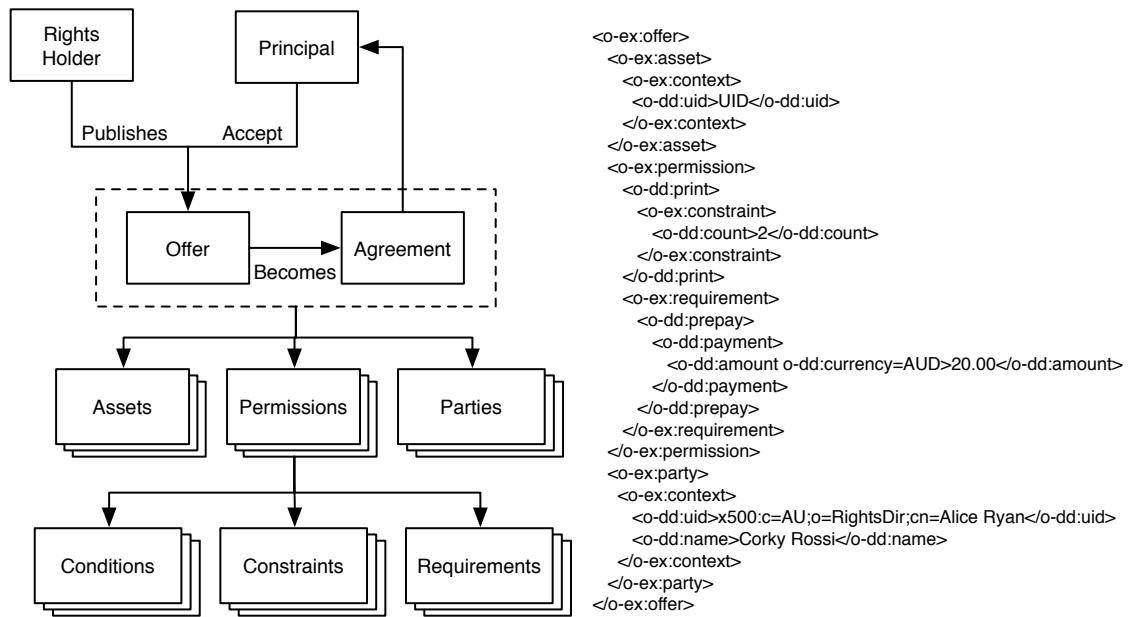


Figure 2.5: ODRL's core concepts and example offer

that is a version of the original offer adapted to her specific identity and the context of her request. Both offers and agreements contain *permissions*, i.e. the set of actions allowed over the asset. Figure 2.5 depicts a simplified model of ODRL's core concepts and an example offer authorising Alice Ryan to a maximum of two accesses to an online resource, provided a payment of 20 Australian dollars is performed in advance. The model explicitly distinguishes amongst: 1) *constraints*, i.e. restrictions on permissions such as temporal restrictions or limited number of allowed accesses; 2) *requirements*, i.e. obligations such as payments that must be fulfilled before access is granted, and 3) *conditions*, i.e. events that may occur during access and that might invalidate the corresponding authorisation. Both parties and assets are specified via unique identifiers. As for XrML, this requires a centralised management of all identities.

EPAL

EPAL is a language for writing privacy policies to handle data through fine-grained positive and negative authorisations. As opposed to XrML and ODRL that assume a set of predefined and universally understood terms, EPAL lets the parties involved in the data exchange define a common vocabulary providing all the terms for user and data categories (that can also be organised in hierarchies), purposes (the declaration of the reasons for which recipients need to access data), actions, containers (the context of access requests) and obligations. Policies can then be specified that refer to specific vocabularies. Figure 2.6 depicts a simplified model of EPAL's core concepts and an example policy authorising paramedics to access for one day John Doe's data for first aid

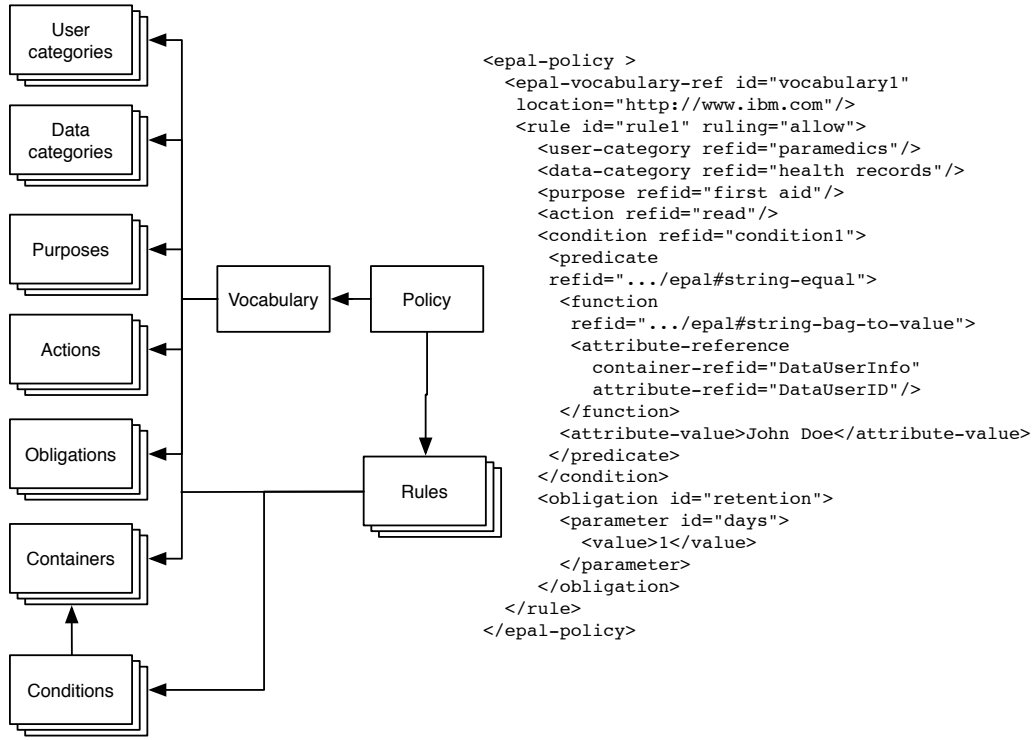


Figure 2.6: EPAL's core concepts and example

purposes.

XACML

XACML is a popular XML-based standard specifying both an authorisation policy language and an access request-decision-response data-flow protocol and message format. While the language shares many elements with XrML, ODRL and EPAL, we consider the policy evaluation and enforcement protocol exemplary. Many of the components that will be part of the solutions we propose throughout this thesis come from XACML's model or are extensions and modifications of its components. Figure 2.7 shows the classical components deployed to perform the protocol. First, a *policy decision point* (PDP) interacts with a *policy administration point* (PAP) to retrieve the most recent available of policies (step 1). Each access request (containing information about the requesting subject, the requested resource, the requested action and the request context) must then be sent from an application to a *policy enforcement point* (PEP) that acts as a protection layer between the application used by the subject to perform the access and the requested resource (step 2). The PEP queries the PDP via a context handler to obtain an access decision (steps 3 and 4). The PDP looks for policies applicable to the request and can request attribute values to the context handler (step 5). A *policy information point* (PIP) is queried to obtain information on subjects, context or on the resource itself (steps 6 to 10). Once policies have been evaluated, a response containing the access decision and a set of obligations is returned to the PEP (step

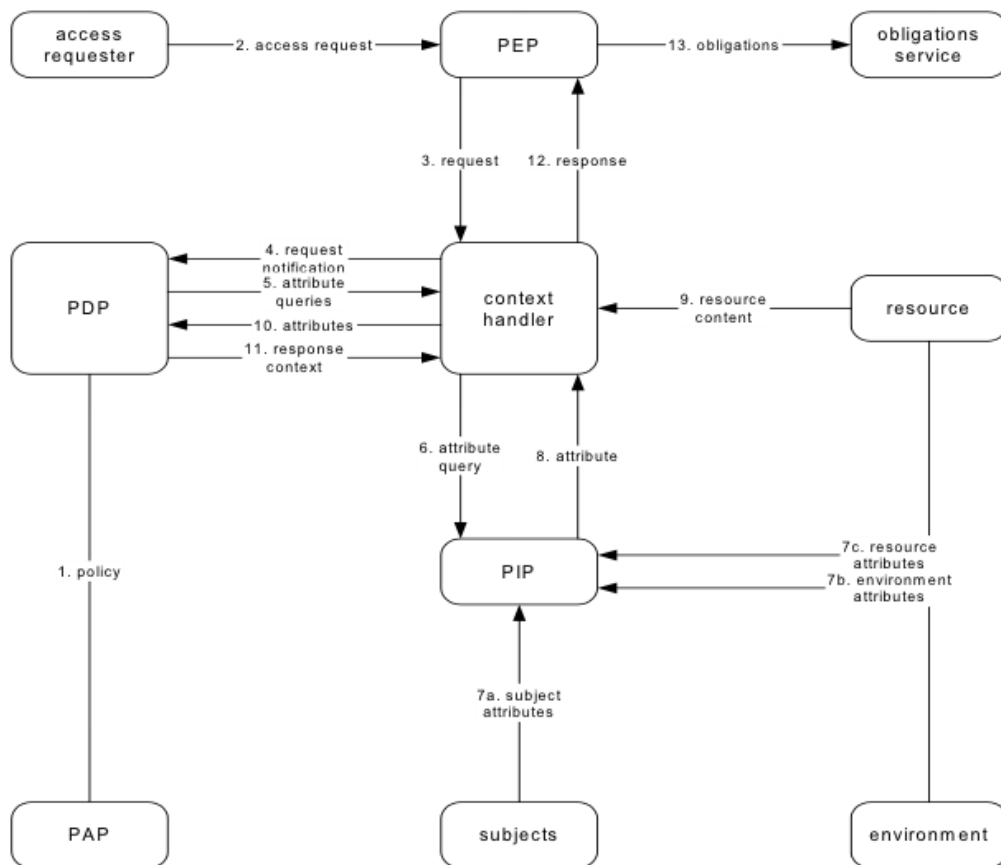


Figure 2.7: XACML data-flow diagram. The numbers express temporal ordering amongst the exchanged messages

11) that can authorise the access (step 12). When enforcing the policy decision and thus before authorising or denying the access, the PEP must perform the set of obligations returned (step 13). This implies that the obligations belong to a predefined set of procedures the PEP can execute. The proposed scheme is at the base of most of the existing policy enforcement platforms.

At the base of any XACML policy is a *Policy* or *PolicySet* element. A *PolicySet* can contain multiple *Policy* and *PolicySet* elements and even references to policies contained in remote locations. When an access request is issued, the PDP finds the applicable policies by matching the values of *action*, *subject*, *resource* and *environment attributes* returned by the PIP or attached to the request with the corresponding constraints specified in the *target* and *condition* elements of the policies. Each attribute can have an issue date and an issuer identifier and conditions can be specified both on the attribute's value and issuance information. Figure 2.8 depicts a simplified model of XACML's core concepts and an example policy permitting access to anyone on a "SampleServer" online resource, provided an email is sent to log the access. Since more than one rule per policy or more than one policy might be applicable to the same request and given that both positive and negative authorisations are possible, XACML allows the specification of *combining algorithms* indicating how conflicts between policies

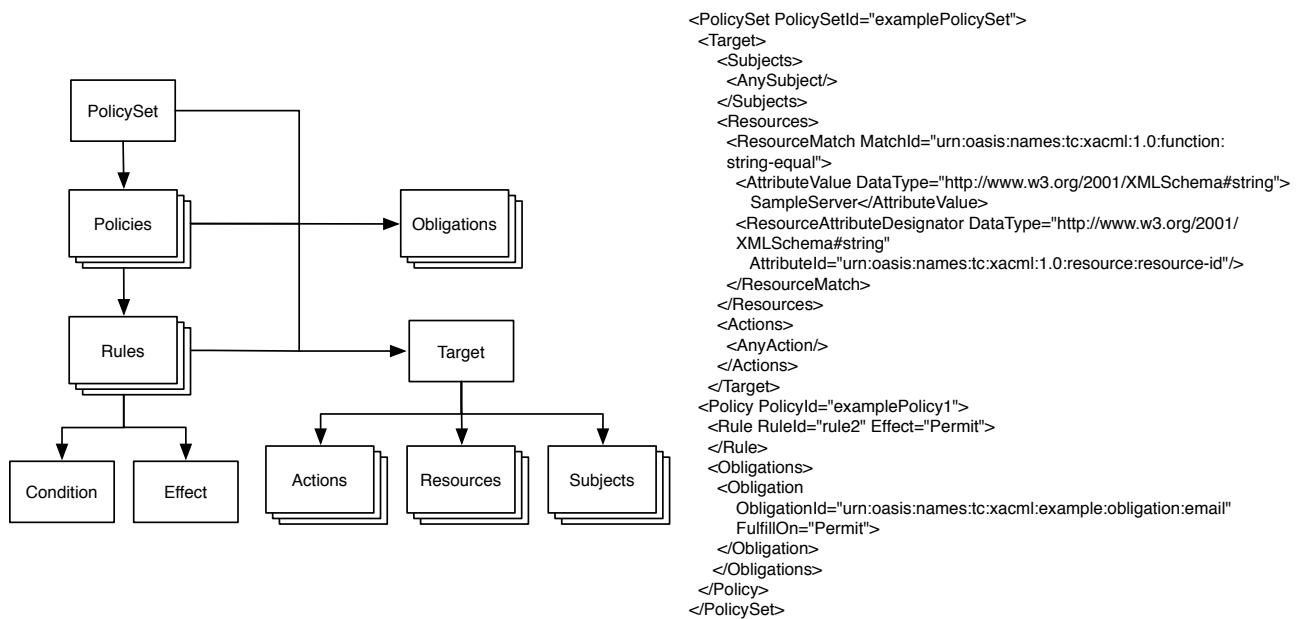


Figure 2.8: XACML core concepts and example policy

can be solved. Examples are: “deny-overrides” (if a rule or *Policy* returns a denial, then the *PolicySet* returns a denial), “permit-overrides” (if a rule or *Policy* returns a permission, then the *PolicySet* returns a permission) and “first-applicable” (the first *Policy* applicable for the specific request overrides all the others).

Although still limited in this sense, the possibility to specify time constraints, limited number of accesses or to interrupt access when certain conditions are verified represent a first step for XML-based authorisation languages towards the specification of usage control policies.

Besides those presented here, several other languages exist and are often considered a good choice for different applications, such as the **Platform for Privacy Preferences Project (P3P)** [CLM⁺]. We here intended to just give a flavour of the kind of access control policies that can be specified and not to fully cover all available technologies. However, the use of these solutions has been limited by their complexity and the lack of a formal semantics that made their actual enforcement very difficult. In fact, whilst the policy syntax is well-specified, the interpretation and evaluation of the policies is left to the specific implementation of the PDP. This resulted in differing implementations that are rarely compatible and thus usable in an open environment. This also makes it impossible to compare policies from different organisations or decide whether a policy has been satisfied or not from, for example, an audit log. As we will describe later, we will use a simplified formalism to specify policies that actually reproduces XACML policies. For the sake of this thesis we must also emphasise that all such languages lack constructs to specify controls on derived data. They can control how data can be transformed (i.e. who can perform which action on which resource) but not what protection will be applied to the output of the transformation (Issue 3). Also, the enforcement models used to actually apply the policies specified in such

languages have mostly tried to reproduce the XACML model and thus its limitations. In particular, the model does not allow recipients to access data when there is no connectivity and thus when the PEP of the remote authority possessing the data decryption keys cannot be contacted (Issue 3). Moreover, it assumes that all users are willing to send their credentials to such remote PEP, despite these being possibly private or confidential and the authority being possibly untrusted by the user (Issue 2).

2.4.2 Access Control Policies for Hierarchical Domains

Managing modern access control systems usually involves a large number of resources and subjects that might need to access them. This complexity inevitably results in high administrative costs and long deployment cycles. To address this problem and make system management more scalable, many solutions rely on a hierarchical organisation of the managed elements. This allows administrators to group subjects and resources into categories explicitly (e.g. roles and groups for subjects, containers and sets for resources) that can be more easily controlled. It is particularly important here to distinguish between *users*, *groups* and *roles*. Users are identified individuals, groups are named sets of users and roles are named sets of privileges that can be bestowed upon users. The hierarchical organisation of resources also results in the definition of several *policy propagation modalities*, i.e. of strategies deciding how the result of policy evaluations at given levels in the hierarchies influences policy evaluations at underlying levels. Some example policy propagation strategies are [JSSS01]:

- **No propagation:** policies are not propagated;
- **No overriding:** all policies applied to an object are propagated to the underlying objects in the hierarchy, regardless of the existence of other contradicting policies;
- **Most specific override:** policies applied to an object are propagated to the underlying objects if no policies specified for lower-level objects exist that contradict them;
- **Most general override:** policies applied to an object are applicable and propagated to the underlying objects if no policies specified for higher-level objects exist that contradict them;
- **Path overrides:** policies applied to an object are propagated to the underlying objects if not overridden. A policy applied to an object *o* overrides a contradicting policy of an higher-level object *o'* for all underlying objects of *o* only for the paths passing from *o* [BJS99].

Similarly, whenever *modality conflicts* arise, i.e. whenever either propagated or non-propagated policies return different results for the same authorisation request, a *conflict resolution strategy* must be applied. Some

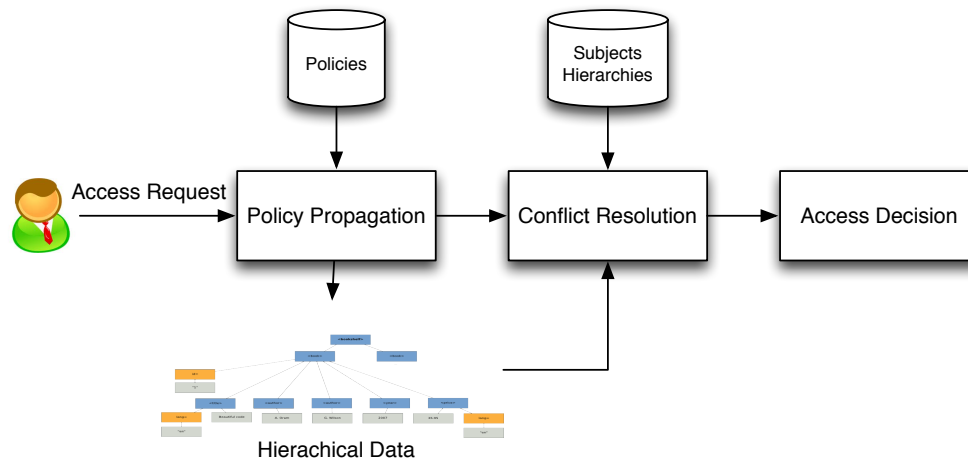


Figure 2.9: Policy evaluation process for hierarchical domains

example conflict resolution strategies are [JSSS01]:

- **No conflicts:** the presence of conflicts is considered an error;
- **Denials override:** negative authorisations take precedence over positive authorisations;
- **Permissions override:** positive authorisations take precedence over negative authorisations;
- **Strong over weak:** authorisations can be specified as being *strong* or *weak*, where *strong* authorisations override *weak* authorisations [BJS99];
- **Nothing takes precedence:** decision is deferred to specifically defined *decision policies*.

Several other solutions are possible. Dong et al. [DRD08] proposed to use labelled logic programs to prioritise policies and thus realise different conflict resolution strategies. We emphasise that the relative pros and cons of both policy propagation and conflict resolution strategies depend on the specific usage scenario and its requirements. Whenever we favour confidentiality over availability, an *all-propagate deny overrides* strategy is preferable, whereas if we favour availability a *no propagation permit overrides* strategy would be. More complex strategies offer more control to the policy writer, but also make it more complex specify policies and verify their future behaviour. Figure 2.9 depicts a simple policy evaluation process for hierarchical domains. Once policies have been propagated and conflicts resolved, there might still be some accesses that are neither authorised nor denied. In this case, *decision policies* (or *default policies*) are used to determine the final access decision. Examples are the *open* and *closed* default policies described earlier.

Access Control for XML Documents

XML documents can be considered collections of nested elements and a perfect example of hierarchical organisation of data. Following the widespread adoption of XML as a standard data model, there has been an ever increasing attention and need for solutions that allow the specification and enforcement of security restrictions on XML documents. **Author-X** [BCF01, BBC⁺00] is one of the first tools for protection of XML documents that have been proposed. The tool had some important features that distinguished XML protection from traditional access control solutions: 1) it supported both *set-oriented* and *document-oriented* protection by allowing the specification of security requirements at both *document type definition (DTD)* level (i.e. requirements applicable to all documents satisfying a certain structure) and document level; 2) it supported both positive and negative authorisations for specific elements in the XML document's graph structure; 3) it allowed policy propagation at different levels of granularity (i.e. it allowed authorisation decisions for certain XML elements to be passed on to the elements' children at controllable levels of depth).

Author-X realises protection at different levels of granularity exploiting the structured characterisation of XML and DTD documents. Policies can be applied to sets of documents satisfying the same DTD or to specific sections, subsection and attributes of specific documents. Authorisations are of 5-tuples of the form *(subjects, targets, privilege, prop-opt, sign)* where the *privilege* term indicates the requested access mode (read, navigate, write and append), *prop-opt* indicates the method of propagation of policies through the different levels of the XML tree (none, cascade, or first level) and *sign* indicates the sign of the permission (denied or granted). Policies are stored as an XML file (the "policy base") while another XML file stores the list of users and their credentials (the "credential base"). Policy targets are specified as either a DTD file name (indicating that the policy applies to all documents of that type) or as an XML file name together with an XPath expression [xpa99] identifying specific sub-elements of the document. Subjects are instead specified as XPath expressions over the credential base. Propagation options control if a policy applies to its target elements only, to its targets' direct children or to all its targets' indirect children. Figure 2.10 depicts an example policy authorising secretaries from the sales department to access purchase documents. Carrier employees working for the company named "CCX" can read a specific purchase document for their company but cannot read the description of the contained items.

The Author-X's conflict resolution strategy is a *more specific overrides - deny overrides* strategy where policies referring to inner elements in the XML structure prevail over policies referring to higher-level elements or DTD files. If policies specified at the same level of granularity conflict, then denial decisions prevail.

When a recipient requests access to a document and once an authorisation decision has been taken for all elements of the document, the server storing the document prunes it prior to release so that elements the recip-

```

<policy_base>
  <policy_spec cred_expr="//secretary [department="sales"]"
    target="Purchase_order.dtd" priv="ALL" type="GRANT" prop="CASCADE"/ >
  <policy_spec cred_expr="//carrier_employee[company="CCX"]"
    target="Purchase_order.xml" path = "//Purchase_order[Purchase_order/
      carrier/name="CCX"]"
    priv="VIEW" type="GRANT" prop="CASCADE"/ >
  <policy_spec cred_expr="//carrier_employee[company="CCX"]"
    target="Purchase_order.xml" path = "//item" priv="VIEW" type="DENY"
    prop="CASCADE"/ >
</policy_base>

```

Figure 2.10: Author-X example policy

ient cannot read or navigate are selectively removed. A different solution devised to support push distribution from the server to recipients encrypts the sections of the document protected by the same set of policies with the same cryptographic key. The resulting document can then be sent to all the recipients that also receive the appropriate subset of keys. For *authoring* rights (i.e. write and append), the policy enforcement is executed by the server when the document is sent back from the user to be stored, as the server removes from the document all non-authorised modifications.

Damiani *et al.* [DdVPS02, DdVPS00] proposed an access control model based on the definition and enforcement of access restrictions directly on the structure of XML documents. The authors defined three hierarchies for users and groups of users, IP addresses and symbolic names (URLs) whose Cartesian product is used as the hierarchy for policy subjects. Authorisations can thus be specified for users regardless of their physical location, for physical locations regardless of the user identity or for both. Policy targets are instead specified as a combination of a URI and an XPath expression to identify specific sub-elements of XML documents. Authorisations can be specified as:

- **Positive or negative;**
- **Local or recursive:** the authorisation can either be applied to the object's attributes only or recursively to the object's sub-elements. Authorisations propagate down the XML tree until a conflicting one is found, thus realising a *more specific overrides* conflict resolution strategy.
- **DTD:** the authorisation can either be specified for a document type or for specific documents. By default, document-level authorisations override DTD-level authorisations.
- **Hard or soft:** Soft authorisations are document-level authorisation that can be overridden by conflicting DTD-level authorisation, while hard authorisations are authorisations that cannot be overridden. If two hard authorisations conflict, document-level authorisations still override DTD-level authorisations.

Whenever several authorisations defined at the same level conflict, a *most specific subject - denials override* conflict resolution strategy is applied. In other words, if a user is part of more than one group in the subject hierarchy so that more than one authorisation can be applied, then the authorisations applicable to the most specific group (i.e. the one at the lowest level) override the others. Amongst these, negative authorisations override positive ones. Once an authorisation has been determined for each element in the XML structure, the document is pruned so that all the subtrees that cannot be accessed are eliminated. The attributes of elements that cannot be accessed but that cannot be eliminated completely (because some of their subelements can be accessed) are also removed.

The main problem with both this approach [DdVPS02, DdVPS00] and Author-X [BCF01, BBC⁺00] is that they do not scale well. For example, the Author-X key generation process may result in a vast amount of keys depending on the XML file size and must be performed every time there is a change in the policy or credential base. Moreover, these approaches place a significant administrative burden on the policy writer who has to specify complex XML-based policies for every document she wants to publish. DTD-level authorisation alleviate this problem, and this is why we take a similar approach in our solution. Also, both approaches do not consider what happens to the policies associated with XML elements when the XML document is transformed, for example with one of the many XML transformation languages and engines available nowadays (e.g. XQuery). Also, these approaches assume all users are connected and able to contact the policy evaluation authority that can issue decryption keys and filter the data. This limits the solutions to specific environments and makes them unusable in situations such as the one described in our emergency scenario (Issue 3).

Miklau *et al.* [MS03] proposed a framework where fine-grained authorisations are specified through a modified version of the XQuery language [xqu10]. Rights to access a document or part of it are specified as possession of given sets of cryptographic keys that are used for encryption. An example authorisation policy is depicted in Figure 2.11 stating that users who want to access the *analysis* element of all “subject” elements in a document must possess the “registration key” (that is distributed by the document owner to rightful users) and must know the content of the *DNAsignature* element of the same subject.

```

SUFFICIENT
FOR          $x in /doc/subjects/subject
KEY          getKey("registration")
                $x/analysis/DNAsignature/text()
TARGET      $x/analysis

```

Figure 2.11: Authorisation example: registration key and content of DNASignature necessary to access analysis elements

According to the specified policies, every element in the XML document is associated with a Boolean ex-

pression over a set of encryption keys (e.g. $(k_1 \wedge k_2) \vee k_3$) indicating the sets of keys necessary to decrypt it. The document is then *normalised* so that every element is associated and encrypted with one key only. This is obtained by creating new hidden “fake” elements protected with the original keys and containing shares of the new key needed to decrypt the original element. Encryption is then performed according to the XML Encryption Syntax and Processing standard [IDS02]. Whilst this solution allows to specify new types of conditions such as the knowledge of a specific document subelement, it still does not address the main issues we consider in this thesis. First, policies are not transformed to adapt to documents being modified (Issue 1). Second, since keys are distributed beforehand to users that thus do not need to contact remote authorities, it is not possible to evaluate usage conditions that depend for example on changing context (e.g. time of the day, location etc). Similarly, access and usage monitoring and revocation is impossible. Once keys have been distributed, they cannot be claimed back by the data owner. These are the main limitations of this type of solutions to the problem of usage control in disconnected environments (Issue 3).

Crampton [Cra04] proposed instead to generate a key hierarchy (see Section 2.5.1) [AT83, MTMA85] such that knowledge of the key associated with an element of an XML document would allow a user to derive the keys for all subsequent elements. The structure of the hierarchy is obtained by specifying authorisations through XPath Filter expressions [xpa02] and using their *containment* relationship (intuitively, an expression e_1 is contained by an expression e_2 if the set of elements it returns is a subset of the elements returned by e_2). Unfortunately, deciding whether an expression contains another one is undecidable in general [Woo02, NS02, MS04] and the proposed approach is thus restricted to fragments of the XPath Filter language. Also, the solution has the same problems related with access and usage monitoring and revocation described above for the proposal of **Miklau et al.** [MS03].

Flexible Authorisation Framework

The **Flexible Authorization Framework (FAF)** [JSSS01] is an access control framework based on a logic-based policy language called the **Authorization Specification Language (ASL)**. Authorisation subjects in FAF can either be users, groups of users or roles and can thus be part of one of two distinct hierarchies (groups and roles hierarchies). Authorisation objects can either be elements from a data structure (e.g. a file system) or roles and can thus as well be part of one of two distinct hierarchies (objects and roles hierarchies). A FAF authorisation can be seen as a tuple $(o, s, sign)$ where o is a member of one of the two object hierarchies, s is a member of one of the two subject hierarchies and $sign$ indicates whether the authorisation is positive or negative. An ASL authorisation is a rule of the form:

$$cando(o, s, sign) \leftarrow L_1 \& \dots \& L_n \quad (2.10)$$

where o is the authorisation object, s is the authorisation subject, $sign$ is the sign of the authorisation and $L_1 \dots L_n$ are literals (atomic possibly negated conditions) that must hold for the authorisation to be valid. Literals can express ordering or membership relationships in one of the available hierarchies (with the *in* and *dirin* predicates), application-specific predicates (e.g. with the binary predicate *owner* indicating the owner of an object) or history-based conditions (with the *done*(o, u, r, a, t) predicate indicating whether user u with role r has executed action a on object o at time t). ASL also includes predicates to represent derived (or propagated) authorisations (*dercando*), decision policies (*do*), conflict resolution strategies (*over_{AO}* and *over_{AS}*) and a propositional symbol for integrity constraints (*error*), i.e. restrictions on the content of data after it has been accessed.

Despite its flexibility and the wide range of policies ASL can express, it still does not allow users to specify policies for derived data (Issue 1). Moreover, its evaluation process assumes that the central evaluator possesses all credentials for all users who request access, despite these being possibly confidential or private (Issue 2).

Ponder2

Ponder2 [DDLS01, RDD07] is a declarative, object-oriented language for the specification of management and authorisation policies. The core idea of the language is that resources and subjects are organised in the same distributed domain structure under the common definition of *managed objects*. Domains form a tree structure (as those depicted in Figure 2.14) where nodes can overlap (contain the same objects) and be physically placed on different remote locations. Managed objects can interact with each other through procedure or function calls or by triggering events. Policies are used to control the allowed procedure calls via authorisation policies and to control responses to triggered events via *event-condition-action (ECA) rules*. Domains, events and policies are themselves managed objects, thus easing the administration process. A Ponder2 authorisation policy is composed of the following terms:

- **Sign:** either negative or positive to indicate whether the policy is a negative or positive authorisation;
- **Subject:** the managed object in the domain structure invoking a remote procedure call on another managed object;
- **Action:** the invoked remote procedure call;
- **Target:** the managed object in the domain structure responding to the remote procedure call;

- **Condition:** the conditions over the call's parameters that must hold for the policy to be valid;
- **Focus:** the indication of the side of the interaction where the policy must be evaluated. It can either be "t" (target) or "s" (subject).

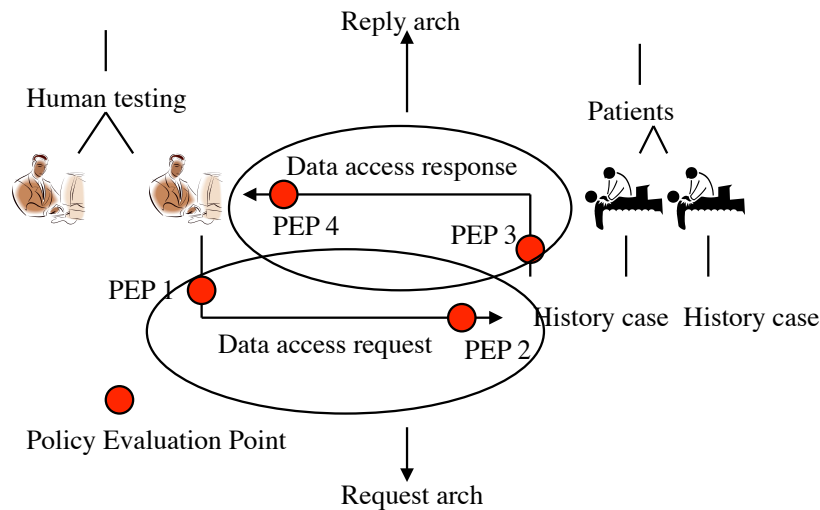


Figure 2.12: Evaluation points for authorisations in Ponder2

An authorisation's focus represents at which point of the interaction between client object and server object the policy is evaluated (Figure 2.12). Policies can: 1) permit or deny the client object to invoke a service; 2) permit or deny the server object to accept an invocation; 3) permit or deny the server object to send back certain return values; 4) permit or deny the client object to receive certain return values.

Figure 2.13 shows two example policies, the first preventing researchers from the Medical Research Centre (MRC) from requesting medical information from untrusted doctors in the Sacred Hearth Hospital (SHH), the second preventing doctors from SHH from issuing medical information to untrusted researchers in MRC.

```

subject: root/MRC/HR_dpt/Human_testing
action: "medicalInfo"
target: root/SHH/Untrusted/Doctors
focus: "s"
reqneg

subject: root/MRC/Untrusted/Researchers
action: "medicalInfo"
target: root/SHH/MedAffairsDpt/Neurology/Doctors
focus: "t"
reqneg

```

Figure 2.13: Ponder2 example authorisation policies

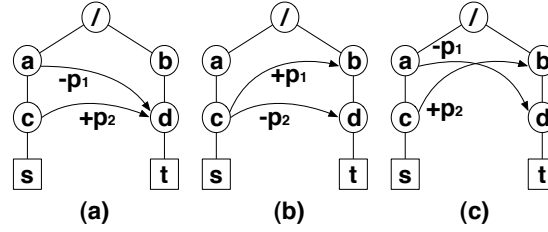


Figure 2.14: Conflicting policies in Ponder2

In Ponder2, policies are propagated in the hierarchy of managed objects so that policies applied to a given domain are also applied to all its nested domains and objects. Ponder2 resolves conflicts by allowing policies to be defined as *final* (so that they cannot be overridden) and by applying a default conflict resolution strategy for unsolved conflicts. The strategy is based on the calculation of the length of the path between subject and target of a given call. We must thus distinguish between two cases: conflicts between final policies and conflicts between normal policies. In the first case, policies with the longest path between subject and target have precedence. In the second case, policies with the shortest path have precedence. The rationale behind this is that more specific policies should take precedence over more generic ones. However, since final policies are intended to override more specific ones, their conflict resolution strategy must be reversed. If policies with the same path length conflict, then negative policies have precedence. In the examples in Figure 2.14, the managed object s in domains c and a , sends a call to managed object t in domain d and b . Assuming policies are not final, in example (a) the positive authorisation p_2 with subject c and target d has precedence over the negative authorisation p_1 with subject a and target d ; in example (b) the negative policy p_2 has precedence, while in example (c) the negative policy p_1 has precedence.

While Ponder2 gives the possibility to specify different and complex policies in a very simple syntax, its hierarchical domain model where all managed objects are connected and managed by the same administrator is not suitable for dissemination control, where different entities might be disconnected and have little or no knowledge of each other (Issues 2 and 3). Also, its authorisation model implies subjects are characterised only by the domains they belong to and no conditions can be specified apart from boolean expressions on the function call's parameters. Finally, Ponder2's policy model is oblivious of whatever happens to managed objects. Their content or even their behaviour might change but the policies applied to them would remain the same (Issue 3).

2.4.3 Trust management languages

“The most fundamental difference between traditional security systems and trust management systems is that in the former trust is static and pre-defined while in the latter trust is dynamic and established on-the-fly” [Don08].

In traditional access control systems, users are first authenticated and universally identified and only after they are assigned roles, rights, keys or can be properly evaluated on the basis of the available policies. In other words, the access control system determines whether users are *trusted* when ascertaining their identity. This is usually achieved via a centralised authority who knows how to evaluate all identity and attribute credentials users can present. However, in modern large scale networks, applications are distributed across several administrative domains that know little about each other’s users and their attributes. In such a scenario, trust relationships must be created and modified on the fly as new users enter and leave the system. *Trust management* solutions depart from the traditional separation of concepts between authentication and authorisation and make use of *digital certificates*, i.e. of signed statements certified by *certification authorities*, to describe chain of trusts between entities such as an access control server and an unknown user.

As we will describe in the following, trust management systems suffer from a few shortcomings. First, entities that must take a trust decision on others must be able to gather all digital certificates that could build a chain of trust. This is not an easy feat, especially in highly distributed environments or even more in disconnected environments such as our crisis management scenario (Issue 3). Second, certificates may contain private information and users may not be so keen on releasing them (Issue 2). Third, the creation of chains of trust relies on users delegating authority over their trust relationships to other users. For example, Alice may delegate the decision over her list of trusted people to Bob, stating that she trusts whoever Bob trusts. However, Bob may further delegate his authority, and Alice would loose control over the list of her trustees completely. Finally, while the policy languages for trust management we will describe in the following are also used to specify access control requirements, they all lack constructs to specify how derived data can be protected (Issue 1).

Policy Maker

Policy Maker [BFL96] was one of the first proposed trust management systems. A Policy Maker policy, or *assertion*, makes use of public cryptographic keys to identify users and to bind their identities with privileges. An assertion has the form:

Source **ASSERTS** *Authority Structure* **WHERE** *Filter*

where *Source* represents the source, or writer, of the assertion and can be either a public key to represent an

external user or a policy identifier to represent a policy specified by the local evaluator. An *Authority Structure* indicates the users that the assertion refers to and thus contains one or more public keys. Finally, a *Filter* contains the constraints to be satisfied for the assertion to hold. For example, the assertion:

$$policy \textbf{ASSERTS} key_{Alice} \textbf{WHERE} action = read$$

specifies that, for the local policy writer/evaluator, Alice can perform the *read* action. Chains of trust can be created when several assertions written and signed by different sources are combined:

$$policy \textbf{ASSERTS} key_{Bob} \textbf{WHERE} authorisereadby(Bob)$$

$$key_{Bob} \textbf{ASSERTS} key_{Alice} \textbf{WHERE} action = read$$

The first assertion states that the local policy writer authorises Bob to authorise other users to perform the *read* action, while with the second assertion Bob actually authorises Alice to read. The final result is that the local policy writer actually authorises Alice to read.

Policy Maker does not specify any format for the signed credentials containing the assertions and any syntax and semantics for filters, leaving the responsibility to the calling application. It does instead offer a tool to query the trust management engine and verify whether specific actions can be authorised or not. Queries have the form:

$$key_1, key_2 \dots key_n \textbf{REQUESTS} ActionMatch$$

where the *ActionMatch* is the context of the query that must satisfy the policies' filters. An example query satisfying the previous chain of assertion would be:

$$key_{Alice} \textbf{REQUESTS} action=read, authorisereadby(Bob)$$

Trust Policy Language

With the **Trust Policy Language (TPL)** [HMM⁺00], IBM merged RBAC with trust management. In RBAC users are not assigned permissions directly but rather through a level of indirection represented by roles. Roles are assigned permissions and organised in an inheritance hierarchy while users are assigned to roles to obtain the corresponding permissions. TPL extends RBAC by using trust management concepts to assign users to roles. Each role has one or more rules stating what conditions a user must satisfy to be assigned to the role, where rules are conditions on the set of credentials signed by third parties and that the user must present. Again, roles can be chained specifying rules that require certificates to be signed by any member of other roles.

```

<GROUP NAME="Hospitals">
  <RULE>
    <INCLUSION ID="from_self" TYPE="Recommendation" FROM="self">
    </INCLUSION>
    </FUNCTION>
  </RULE>
  <RULE>
    <INCLUSION ID="reco" TYPE="Recommendation" FROM="hospitals" REPEAT=2 DEPTH=3> </INCLUSION>
    <EXCLUSION ID="warn" TYPE="Warning" FROM="hospitals">
    </FUNCTION>
    <GT>
      <FIELD ID="reco" NAME="Level"></FIELD>
      <CONST>1</CONST>
    </GT>
    </FUNCTION>
  </RULE>
</GROUP>

```

Figure 2.15: Example TPL policy

Policies are written in XML and consist of a sequence of role definitions through the `<GROUP>` tag. Each `<GROUP>` element can contain several `<RULE>` elements, each specifying the set of credentials that must be presented in the `<INCLUSION>` tag and the condition that they must satisfy in the `<FUNCTION>` tag. For each credential specification it is also possible to specify whether a minimum number of the same credential signed by different issuers should be presented and whether the chain of trust should be limited to a certain depth. The approach is similar to those proposed by other trust management solutions, but TPL explicitly caters for *negative certificates*, i.e. certificates on the basis of which a role membership should not be granted. These can be specified via the `<EXCLUSION>` tag whose semantics is “there does not exist a certificate X (in the local repository) such that the rule’s function holds”. Figure 2.15 shows an example policy stating that entities can be assigned to the “hospitals” role if they possess a recommendation from the policy writer himself or if they possess at least two recommendations signed by entities that are already members of the “hospitals” role. Any entity with a “warning” certificate signed by a hospital should instead be excluded.

TPL also lets the policy writer specify where credentials can be found in case the user under evaluation cannot (or does not want to) provide all of them. This is done via the `<REPOSITORY>` tag that indicates a location in terms of a URL or in terms of other roles. However, while the latter possibility introduces the problem of contacting all members of the specified roles to obtain the credentials (they might be unreachable or might simply refuse to cooperate), the former reduces the system’s degree of distribution as repositories are in the end just centralised servers. Also, even if repositories could take decisions on who is authorised to access the stored credentials, privacy is a big issue of the system. Credentials subjects and issuers should be the only entities to decide who should be allowed to access the credentials themselves, lest the risk that their private information is disseminated amongst unknown and undesired recipients (Issue 2). In this case instead the credentials subjects and issuers have no control over the locations and recipients their information is sent

to.

SPKI/SDSI

Rivest and Lampson originally proposed the **Simple Distributed Security Infrastructure (SDSI)** [RL96] as an alternative, decentralised public-key infrastructure. At the same time, the **Simple Public Key Infrastructure** [Ell99] was developed by the Internet Engineering Task Force (IETF) with the motivation that the existing X.509 infrastructure was too complex. Given their many similarities, SDSI and SPKI were later merged in a unique framework called SPKI/SDSI [CEE⁺02], and this is how we will refer to them in the following.

In SPKI/SDSI every public key is associated with a *local name space*, i.e. with a set of identifiers whose values (the entities they refer to) make sense only for the owner of the corresponding public key. No *global names*, i.e. identifiers that have a unique value for all public keys (users), exist. A *local name* is thus a (key, id) pair where *key* represents the public key of the owner of the name and *id* is the actual identifier of the name. For example, $(K_{Alice}, friends)$ is the local name used by Alice to identify all of her friends. Users can then use their private keys to sign certificates. Two types of certificates are possible:

- **Name certificates** are statements binding a local name to a value, where a value can be either a set of keys or can refer to other names, thus possibly interconnecting different namespaces. A name certificate can be seen as a tuple (K, A, S, V) where K is the issuer's public key whose private key must also be used to sign the certificate, A is an identifier that together with K defines a local name, S is either a key or a name that is being associated with the local name, and V is the certificate's validity. For example, the certificate $(K_{Alice}, friends, K_{Bob}, (August, September))$ signed by K_{Alice} indicates that Alice considers Bob to be her friend between August and September. The certificate $(K_{Alice}, friends, (K_{Bob}, friends), (August, September))$ indicates instead that Alice considers Bob's friends as her friends between August and September.
- **Authorisation certificates** are statements granting or delegating a privilege to a user. An authorisation certificate can be seen as a tuple (K, S, d, T, V) where K is the issuer's public key whose private key must also be used to sign the certificate, S is either a key or a name for the users who are receiving the privilege, d is a Boolean value indicating whether the granted privilege can be delegated or not, T is the *authorisation specification* and V is the certificate's validity. For example, the certificate $(K_{Alice}, K_{Bob}, true, read\ profit\ statement, 1\ year)$ signed by K_{Alice} indicates that Alice is granting Bob the right to read her profit statement for 1 year and that he can pass this privilege down to other users. The certificate $(K_{Alice}, friends, false, read\ profile, 1\ year)$ signed by K_{Alice} indicates instead that Alice is granting all of

her friends the right to read her profile for 1 year, but they cannot pass this right to anyone.

It is easy to see that certificates can be used to specify access control lists and that they can be chained to construct a *trust path* between two users who know nothing about each other. In [CEE⁺02] the authors proposed an algorithm to find a suitable chain of certificates proving a user is authorised to perform a given action, assuming all necessary certificates are available. However, as for other trust management solutions, assuming that all users possess all signed credentials necessary to build a trust chain from them to any evaluator might be not realistic. Also, such an assumption violates the privacy of the owners and subjects of the certificates who might not want to disclose information about their local namespaces (Issue 2).

RT

The **RT framework** [LMW02] is a family of languages for the specification of role-based trust management and authentication policies. The several versions of the language build on top of the basic language RT_0 and add new possibilities to what could already be expressed with TPL, SPKI/SDSI, Policy Maker or other systems like KeyNote [BFIK99] and REFEREE [CFL⁺97]. The access control model that RT languages represent is also called attribute-based access control (ABAC) and is based on the idea that authority over attribute assertion can be delegated, is completely decentralised and attributes can be inferred from other ones.

RT_0 assertions represent signed credentials that assign roles or delegate authorities. There are three types of credentials:

- **Simple membership:** roles are denoted by terms of the form $A.R$, where A is the authority assigning entities to role R . Therefore, a credential signed by entity A and containing an assertion of the form $A.R \leftarrow E$ asserts that entity E has been assigned role $A.R$ by A . Only A can assign role $A.R$, so every entity defines its own local name space. For example, the assertion $Alice.friends \leftarrow Bob$ indicates that Alice considers Bob as one of her friends.
- **Simple inclusion:** A can also define a role $A.R$ to contain different roles $A_j.R_i$ where A_j can either be or not be the same authority as A . The statement $A.R \leftarrow A_j.R_i$ defines a domination or containment relationship between the two roles $A.R$ and $A_j.R_i$. The semantic is: “each entity with role $A_j.R_i$ is automatically assigned also to role $A.R$ ”. Intuitively this credential represents a delegation of authority over role $A.R$ from authority A to authority A_j . For example, the assertion $Alice.friends \leftarrow Bob.friends$ indicates that Alice considers all of Bob’s friends as her friends too.
- **Linking inclusion:** Delegation can also be expressed with more complex patterns such as $A.R \leftarrow A.R_i.R_k$ where the authority over role $A.R$ is delegated to all entities with role $A.R_i$. For example, the assertion

$Alice.friends \leftarrow Alice.relative.friends$ indicates that Alice considers all of her relatives' friends as her friends too.

RT_1 extends RT_0 by allowing parameterised roles. In RT_1 a role is specified with the term $A.R(h_1 \dots h_n)$ where $h_1 \dots h_n$ are parameters representing role attributes or connecting different roles. Attribute-based delegation can thus be obtained with the assertion $A.R \leftarrow A_j.R_i(h_1 \dots h_n).R_k$, that states that A delegates the authority over role $A.R$ to every entity $A_j.R_i$ satisfying the attributes $h_1 \dots h_n$. RT_1 also introduces a fourth type of credential: **intersection inclusion**. For example, the assertion $A.R \leftarrow A_1.R_1 \cap A_2.R_2 \cap \dots \cap A_n.R_n$ states that every entity assigned to all roles $A_1.R_1, A_2.R_2 \dots A_n.R_n$ is also assigned to role $A.R$.

RT_2 extends RT_1 with the notion of *o-sets*. An *o-set* is a group of logically related objects that are syntactically represented as RT_1 roles. We will not explain in details RT_2 , but just illustrate an assertion example showing how the new concept can be used to describe permissions. The assertion $A.fileAccess(read, ?F : A.documents(?proj)) \leftarrow A.team(?proj)$ states that A allows all the members of a project to read documents of the same project. The question mark in the assertion is used to indicate named variables, while the colon is used to introduce a constraint over an attribute (in this case a value set).

RT^T introduces *manifold roles* and two role operators that allow the specification of separation of duties (where an entity can be granted a right only if a specific different one has not been already granted) and threshold (where the authorisation of multiple parties is required) policies. RT^D introduces instead the concept of role activation and its delegation, i.e. the possibility to select what role can be activated for each access session (i.e. a time frame where some granted rights are valid). This allows access requestors to avoid using credentials and consuming them (if they are consumable) when they are not needed and to overcome separation of duty policies when possible.

An issue with the RT languages is that intermediate steps in the delegation chain cannot be controlled. Only the first step of the delegation can in fact be constrained with attribute conditions, while all the others are completely left to the discretion of the delegated authorities.

SECPAL

SecPAL [BFG10] is an extensible policy language for authorisation specification. Access requests are treated as logical queries evaluated against a knowledge base of predefined logical clauses. Predicates are used to assign rights to users and delegate them. A SecPAL authorisation policy is composed of a set of assertions of the following form:

$A \text{ says } fact \text{ if } fact_1, \dots, fact_n, c$

where A is the principal issuing the assertion, $fact_1, \dots, fact_2$ are the *conditional facts* that must hold for the assertion to be valid, and c is a *constraint* on the values of the variables contained in the assertion (e.g. temporal, inequality, path and regular expression constraints). Each fact $fact_i$ specifies a property of a principal. Facts are composed of at least an entity identifier and a verb phrase, where verb phrases can have several forms such as:

can doAction [*parameters*]
has access from [*parameters*] **till** [*parameters*]
is attribute
can say_D *fact*
can act as *entity*

where the verb phrase *can say_D* can be used to nest facts and obtain delegation. In particular, D is the delegation flag and can be either 0 or ∞ , indicating whether the delegated right can be exerted only by the direct delegated principal or can be infinitely re-delegated, respectively. For example, the parameterised delegation policy:

FileServer **says** *user* **can say _{∞}** *x* **can access** *resource* **if** *user* **can access** *resource*

states that users can pass on their access rights to other users. Attribute-based delegation, i.e. delegating the authority to assert whether a subject holds an attribute only to other parties that hold other attributes, can also be obtained. The following policy states that a shop gives a discount to students every Friday, where subjects can be defined as students by universities that are in turn certified by the board of education:

Shop **says** *x* **is entitled to discount** **if** *x* **is a student** **till** *date*,
 $currentTime() \leq date, currentDate() = Friday$
Shop **says** *univ* **can say₀** *x* **is a student** **till** *date* **if** *univ* **is a university**
Shop **says** *BoardOfEducation* **can say₀** *univ* **is a university**

Delegation can also be bound to a specific length of the delegation chain different from 0 and ∞ by chaining a specific number of *can say₀* verb phrases. The verb phrase *can act as* can instead be used to express role membership and role hierarchies. For example, the policy:

NHS **says** *FoundationTrainee* **can read** *resource*
NHS **says** *SpecialistTrainee* **can act as** *FoundationTrainee*
NHS **says** *SeniorMedPractitioner* **can act as** *SpecialistTrainee*
NHS **says** *Alice* **can act as** *SeniorMedPractitioner*

builds a role hierarchy valid for the *NHS* principal where a privilege is first assigned to a role and then inherited by its senior roles. Negative conditions are also possible, thus allowing separation of duties, threshold

and prohibition policies. Policy evaluation in SecPal is performed by translating assertions into equivalent constrained Datalog programs. However, this is based on the assumption that all assertions are available to the Datalog engine. As for other trust management solutions, this assumption is not only unrealistic but again violates the privacy of the individuals to whom the assertions refer (Issue 2). In fact, once all assertions have been given to the service running the Datalog engine, users have no control over them.

2.5 Enforcement Platforms for Data Control

Once an organisation has chosen the most suitable access control model or policy language to express its data protection requirements, mechanisms must be put in place that actually enforce the specified controls on data access and usage. Enforcement architectures can vary widely according to the application scenario they are needed for [PSS00]. In this thesis we distinguish between two main types of architecture:

- **Non-interactive data sharing architectures (NIDSA):** non-interactive architectures require potential data recipients to interact with the data originator or a delegated entity *only once*, prior to any access request. The initial interaction usually results in the distribution of a set of privileges or access rights on the basis of the available policies (usually in the form of a cryptographic key that can be used to decrypt several documents) that expire after a predefined time interval during which they can be reused with no further interaction with the original issuer.
- **Interactive data sharing architectures (IDSA):** interactive architectures require data recipients to interact continuously with the data originator or a delegated entity to obtain access to data. Each access to each document requires a different interaction with a remote service that evaluates the requesting user on the basis of the available policies and releases the access (usually by issuing cryptographic keys that can be used to decrypt one document). The XACML enforcement architecture described in Section 2.4.1 has been the basis for most IDSA architectures.

We underline that all solutions that fall in these two categories and that we describe in the following do not consider the problem of derived data (Issue 1). Once data has been published and associated with policies, these cannot be modified when data is transformed.

Since it is often the case that data is required to cross the originating organisation's boundaries to be used by partner organisations or external individuals, enforcement mechanisms are often integrated with *Data Sharing Agreement (DSA)* architectures. A DSA [SSR06] is a common set of rules regulating the exchange, handling and usage of shared data that partner organisations agree on and are expected to enforce in their respective

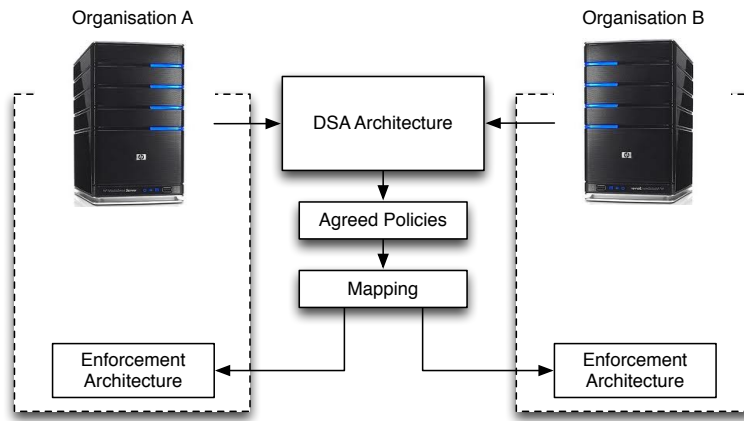


Figure 2.16: DSA architecture scheme

corporate environments. Therefore, with the term *DSA architecture* we indicate the set of components deployed between partners organisations to manage the joint specification of one or several DSAs and to map the agreed rules to the specific models and languages of each organisation so that they can be enforced (Figure 2.16). As such, a DSA architecture can be deployed on top of both IDSA and NIDSA enforcement architectures to which it provides the policies to be enforced.

2.5.1 Non-Interactive Data Sharing Architectures

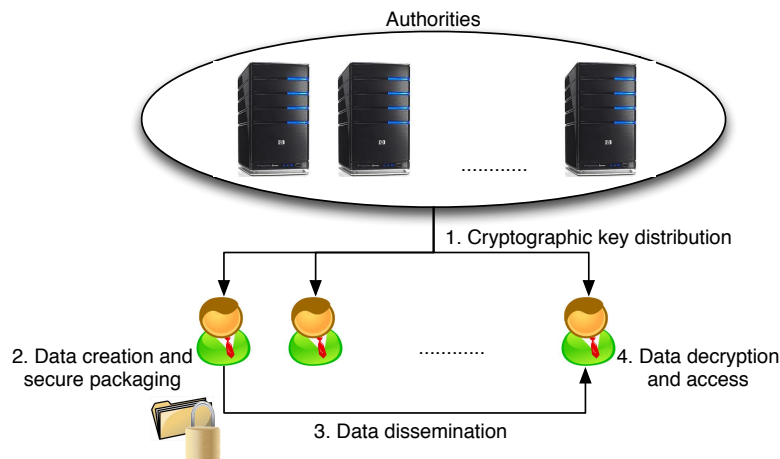


Figure 2.17: Non-Interactive Data Sharing Architecture

Non-interactive data sharing architectures are based on the *a priori* distribution of one or more cryptographic keys amongst potential data recipients, according to the roles or attributes they have in the organisation they belong to. The keys can then be used to access only a subset of the disseminated documents (Figure 2.17).

The advantage of using NIDSA architectures is that documents can be protected (or *published*) before dis-

semination at any time, also when connectivity is not available, and can be similarly accessed. On the other hand, NIDSA architectures suffer from two main shortcomings. First, it is not possible to revoke access rights. Once keys have been distributed by an authority to recipients, they cannot be prevented from accessing the documents the keys allow them to access, nor it is possible to know when they access them. Also, since keys are distributed prior to any data dissemination on the basis of static criteria, no dynamic evaluation of context conditions can be taken into consideration to influence access decisions. Second, it is extremely difficult to disseminate data across organisational boundaries without *re-publishing* it, since key distribution requires a unique key generation and distribution authority shared amongst all organisations cooperating in the data exchange. When passing from organisation A to organisation B, data should first be decrypted by a member of A and passed to a trusted member of B for re-encryption.

Key Management for Access Hierarchies

Access hierarchies are structures defined by an inheritance relationship over privileges and are a common concept in access control. Each node in an access hierarchy represents an access level, i.e. a set of privileges that are inherited by the higher levels. An example access hierarchy is depicted in Figure 2.18. The CEO of a company has all the privileges of both the head of the R&D department and the head of the production department, who in turn have all the privileges granted to all engineers working for the company.

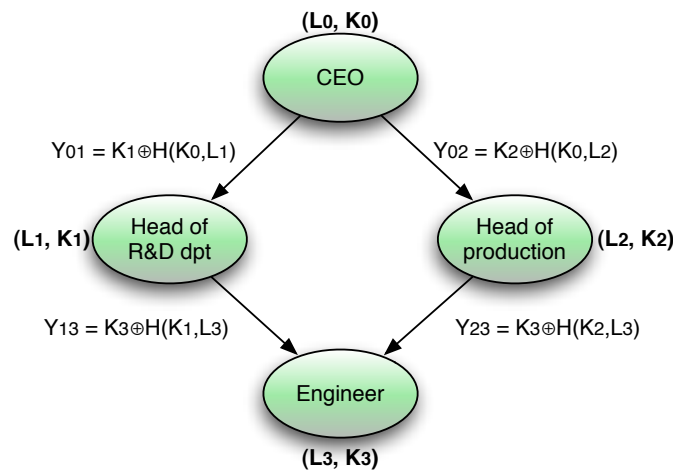


Figure 2.18: Example access hierarchy and associated keys

Atallah et al. [AFB05] proposed a cryptographic scheme to assign privileges organised in an access hierarchy in the form of a secret cryptographic key, so that no further evaluation of the users is necessary to assess their access level when they try to access disseminated data. Figure 2.18 shows an example of the scheme. Each access level i is assigned a public label L_i and a secret cryptographic key k_i while each edge connecting

two levels i and j is assigned a public label $y_{ij} = k_j \oplus H(k_i, L_j)$ where H is a secure hash function and $\oplus$ the xor operator. It is easy to see that any user at access level i can derive the secret keys for the underlying levels j as $k_j = y_{ij} \oplus H(k_i, L_j)$ and thus decrypt all documents encrypted with it.

Taylor et al. [AT83, MTMA85] developed a different model where every node n_i in the hierarchy is assigned a key k_i generated as follows. A public integer t_i is assigned to each node n_i such that t_j divides t_i if and only if n_j dominates n_i . An authority then selects a random secret key k_0 and a secret pair of large primes p and q and publishes their product $M = pq$. Each key is then generated as $k_i = k_0^{t_i} \pmod{M}$. If n_j dominates n_i , then t_i/t_j is an integer and k_i can be computed as $k_j^{t_i/t_j} = k_0^{t_j(t_i/t_j)} \pmod{M}$. If t_i/t_j is not an integer this computation is considered infeasible as it requires extraction of roots modulo M , the product of two unknown primes. To avoid collusion between users, the authors also propose a collusion resistant scheme based on a particular selection of the t_i values.

Attribute-Based Encryption

The term “**attribute-based encryption**” is used to indicate a family of cryptographic schemes [BF01, SW05, Cha07, BSW07, OSW07] where data encryption can be bound to an authorisation policy specifying conditions on the set of attributes recipients must possess to be able to decrypt the data. Attributes are usually distributed by an attribute authority in the form of a set of private parameters. The same authority also publishes a set of public parameters that can be used to encrypt and, combined with private keys, decrypt data. One of the basic schemes [SW05] allows data originators to specify a set of attributes and an error tolerance parameter d such that recipients must possess at least d of the specified attributes to access the data. Attributes are represented by the first $|U|$ elements of Z_p^* , where U is the attributes set and p is a large prime number. The published public parameters are:

$$T_1 = g^{t_1} \dots T_{|U|} = g^{t_{|U|}}, Y = e(g, g)^y \quad (2.11)$$

where $y, t_1 \dots t_{|U|}$ are chosen uniformly at random from Z_p , e is a *bilinear map* (i.e. a function such that for all a and b , $e(g^a, g^b) = e(g, g)^{ab}$) and g is a generator for the group whose prime order is p . A private key for the set of attributes $w \subseteq U$ consists of the set of components $(D_i)_{i \in w}$ where $D_i = g^{\frac{q(i)}{t_i}}$ where q is a polynomial of degree $d - 1$ such that $q(0) = y$. Encryption of a message M for the set of attributes w' is obtained as:

$$E = (w', E' = MY^s, \{E_i = T_i^s\}_{i \in w'}) \quad (2.12)$$

where s is randomly chosen from Z_p . Given a ciphertext E encrypted for the attribute set w' and a private key for the attribute set w such that $|w \cap w'| \geq d$, the cleartext can be obtained as:

$$\frac{E'}{\prod_{i \in S} (e(D_i, E_i))^{\Delta_{i,S}(0)}} = \frac{Me(g, g)^{sy}}{\prod_{i \in S} (e(g^{\frac{q(i)}{t_i}}, g^{st_i}))^{\Delta_{i,S}(0)}} = \frac{Me(g, g)^{sy}}{\prod_{i \in S} (e(g, g)^{sq(i)})^{\Delta_{i,S}(0)}} = M \quad (2.13)$$

where S is any subset of dimension d of $w \cap w'$, $\Delta_{i,S}(0)$ is the Lagrange coefficient and the last step in the equation is obtained via polynomial interpolation.

Successive developments of the basic scheme led to solutions where more complex conditions could be specified to restrain access. In *key-policy attribute-based encryption* [GPSW06, OSW07] ciphertexts are associated with sets of descriptive attributes and users' private keys with policies. Policies are Boolean expressions over attribute values indicating which ciphertext users can access. In *ciphertext-policy attribute-based encryption* [BSW07] users' private keys are associated with sets of descriptive attributes and ciphertexts with policies indicating who can access them. A *multi-authority* scheme [Cha07] was also proposed where private keys can be distributed by different authorities.

2.5.2 Interactive Data Sharing Architectures

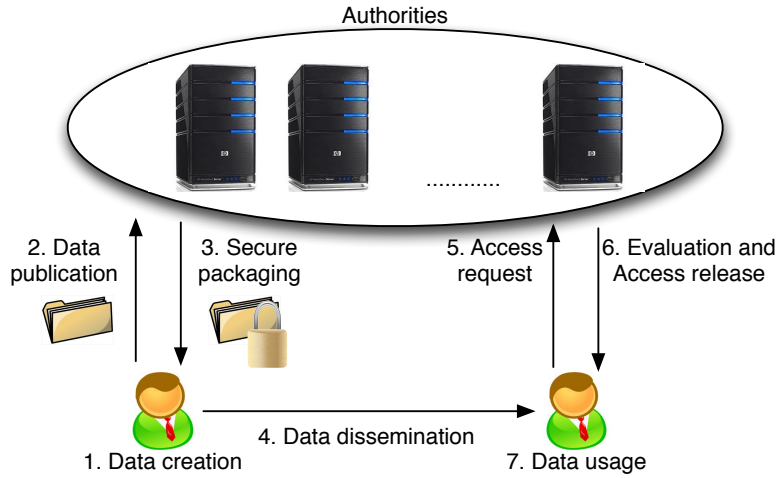


Figure 2.19: Interactive Data Sharing Architecture

Interactive data sharing architectures are based on the online authentication and authorisation of data recipients at the moment they request access to data. Both academia and industry have proposed several approaches to the problem, which is often referred to in industry as enterprise rights management (ERM) or digital rights management (DRM). Although the proposed approaches vary in aspects such as policy/rights deployment and data protection mechanisms, common design principles and functionality can be identified (Figure 2.19). First,

data is associated with access rights (or usage control policies) and cryptographically protected before distribution to recipients (*publication*). Then, a trusted evaluation authority (TA), often the originator himself, responds to access requests, evaluates recipients' rights and issues the decryption keys. A trusted component running on the recipient devices that we will refer to as the virtual machine (VM), ensures that rights are locally enforced. Specific architectures differ in their management of user authentication, policy and rights retrieval, audit of user actions and other tasks [Aut05, Mic05, MPB03, PSS00, PHB06].

The advantages of IDSA architectures are the possibility to evaluate contextual conditions and revocation lists for credentials at the moment recipients request access, and the possibility of deploying inter-organisational links so that recipients can either contact a TA in their own organisation or in an external partner organisation. Although used in many ERM products and in research, this architecture has several limitations. First, the set of TAs is statically and explicitly defined. Changes in this set such as adding a new TA or removing an existing one, require defining new agreements, establishing new trust relationships and implementing them in terms of key exchanges, trust records etc. A second issue concerns the availability of the TA: if recipients cannot contact the specified TA, they cannot access the data received. Furthermore, current protocols assume that recipients are willing to present their credentials to the authority designated by the data originator even when credentials may be confidential. A last issue arises in particular in organisational environments where partners do not know the internal structure of the organisation with whom they interact. In such situations, it is difficult for the originator to specify access rules based on internal roles or to specify which entities in the partner organisation are entitled to issue or verify specific credentials. Instead, recipients are typically better placed to choose which authority could better evaluate their credentials amongst a scope of acceptable authorities defined by the originator (Issue 2). Finally, IDSA architectures suffer from the obvious problem described by Issue 3, as lack of connectivity makes them almost completely unusable.

Microsoft Rights Management Services

Microsoft Rights Management Service (MS RMS) [Mic05] is Microsoft's ERM solution for data protection in enterprise environments and is perhaps one of the solutions that mostly represent the generic IDSA architecture described above.

Each organisation willing to participate in the data sharing must provide a *root RMS server* connected to an Active Directory system for user authentication. Authentication results in the issuance of a *rights account certificate (RAC)* containing the user's ID and email address. To realise cross-organisational authentication, servers also establish trust relationships with each other so that RACs signed by a server can be verified and accepted by several others. Root servers can also authorise other servers to work as *publishing/licensing servers*

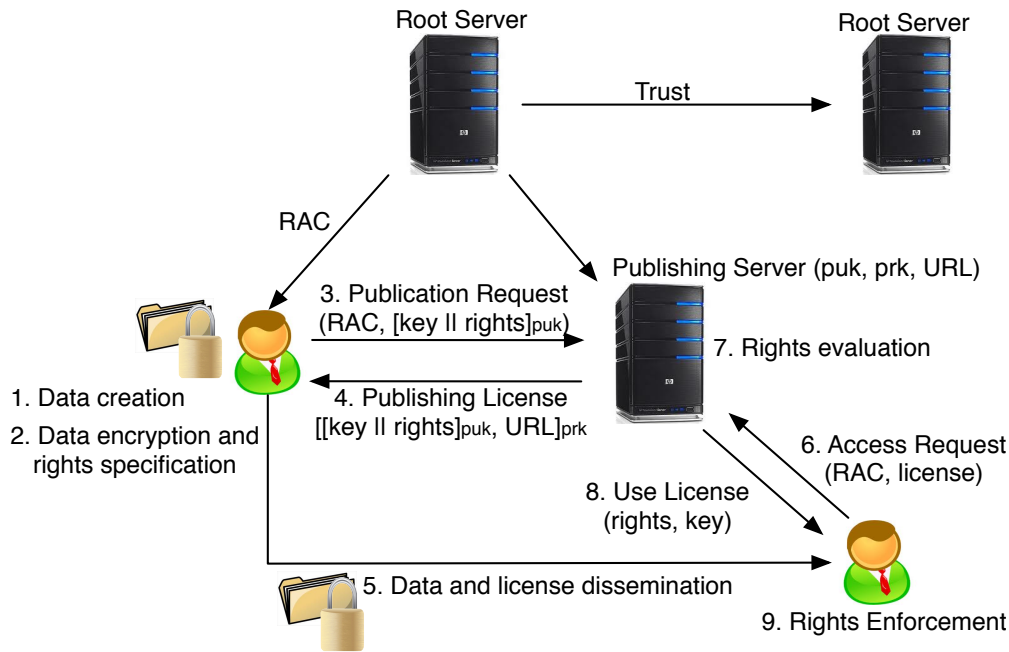


Figure 2.20: Microsoft RMS data protection

(i.e. TAs), generating a *publishing/licensing server hierarchy*. Figure 2.20 illustrates MS RMS's protection scheme. VMs encrypt data with a symmetric key and protect it for a publishing server. The encrypted key is then sent, together with the list of user IDs authorised to access the data, to the publishing server that includes them in a signed *publishing licence*. An offline publication, i.e. the creation of a publishing licence without contacting a publishing server, is also possible but only for privileged users. When recipients try to access a protected document, their VM sends the publishing licence and the recipient's RAC to the publishing server. If the recipient's ID is included in the access control list of the publishing licence, the server releases the decryption key in the form of a *use licence*.

Despite its IDSA architecture, rights revocation and modification is an issue of RMS. Rights are in fact contained in publishing licences and are distributed together with the encrypted documents. Revoking a right would mean deleting all the distributed copies of its licence. Revocation lists can be used to invalidate issued licences or certificates and to prevent users with specific RACs from requesting new licences. However, no fine-grained rights modification is possible. Once a licence has been revoked, recipients must obtain a new, more recent one.

EMC Documentum Information Rights Management Services

EMC Documentum Information Rights Service (formerly Authentica ERM) [EMC, Aut05] is a completely centralised solution where, apart from policy evaluation, trusted servers are also used as storage points. Data publication is performed by encrypting the data with a symmetric key that is then stored, together with an access control list and the data's identifier, on a server. The encrypted data can then be disseminated together with a reference to the publishing server (if more than one exists) so that recipients can contact it to obtain the corresponding decryption key. Servers also manage audit logs and authentication (that can be realised via different systems such as LDAP, X.509 digital certificates, Windows NT domains etc). While with this solution offline publishing is not possible, users can easily modify their published access control lists at any time after the publication by simply contacting the publishing server. The system also provides an off-line access mode. When access to a document is requested for the first time, if the corresponding policy allows for off-line access, then a temporary *lease* (corresponding to an RMS use licence) is generated and cached on the client's machine. Here, the VM is trusted to delete it after expiration. Another feature of this solution is the possibility to enforce mandatory policies, i.e. rights and rules specified at corporate level that are automatically associated with documents when they are distributed through specific controlled channels (e.g. e-mail gateways, network folders and other data sharing systems). The servers managing the services (e.g. SMTP or FTP servers) simply associate the mandatory policies with all incoming data automatically, with no distinction.

Sticky Policies via Identity-Based Encryption

Casassa-Mont *et al.* [MPB03] proposed a high-level model and a related technical solution that allows users to enforce their privacy policies and, at the same time, makes organisations more accountable whilst dealing with users' data. The solution is based on the so-called **sticky policies paradigm**, i.e. on a tamper-resistant binding between data and the policies used to protect it. The binding assures that the policies are available wherever the data is received so that a proper evaluation can be performed. Figure 2.21 depicts the proposed high-level architecture. Data originators who must send sensitive data to external unknown recipients first encrypt it and bind it with sticky policies. This is obtained by encrypting the data with the *identity-based encryption* (IBE) scheme that allows any string to be used as an encryption key. In this case, a policy string is used as the key, where any policy model or language can be used. To obtain a valid IBE decryption key, the data recipient, who has received the encrypted data and the policy, must interact with a TA and provide the information and credentials required by the policy. The TA must also be provided with the correct policy or it will not be able to generate the correct decryption key. This is why the data-policy binding is considered to be tamper-resistant. Upon positive evaluation of the policy, the TA can generate the decryption key and release it to the recipient

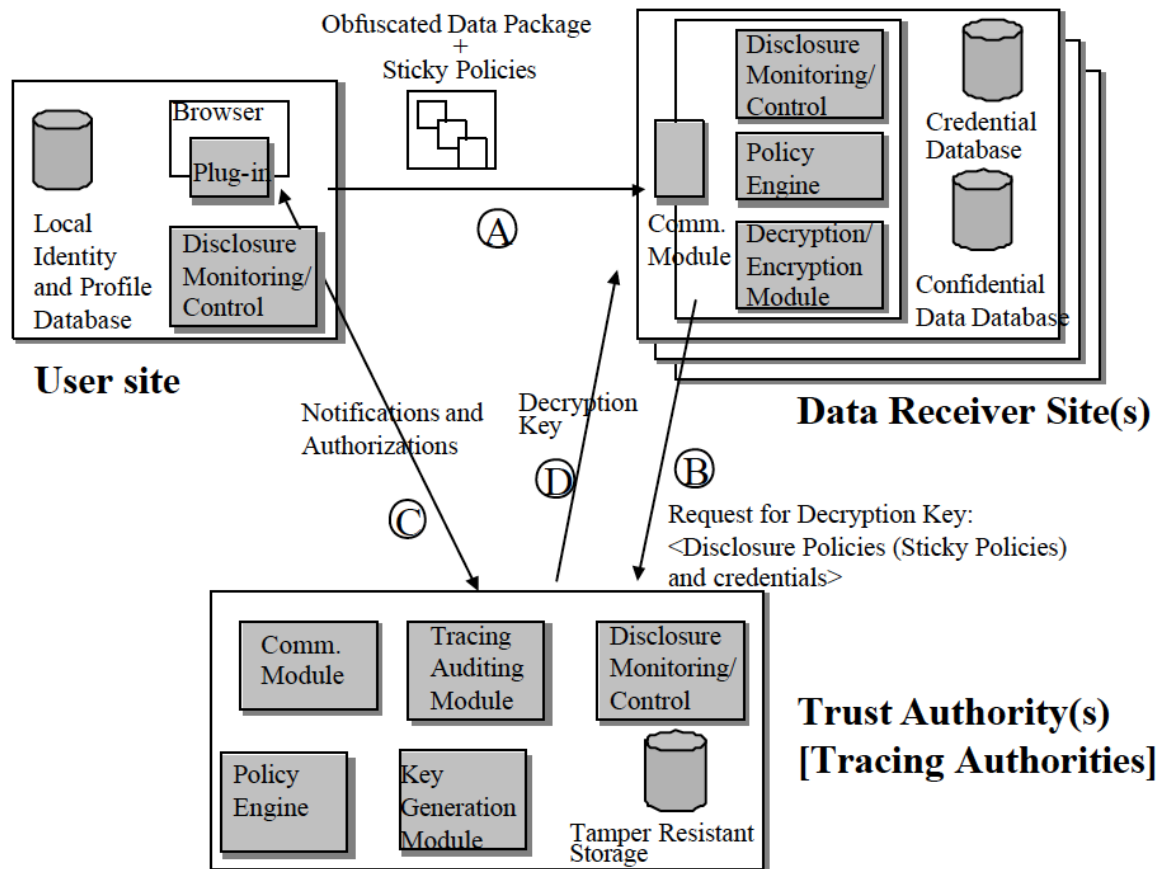


Figure 2.21: High Level ERM Architecture based on the sticky policies paradigm

who can then access the data.

With respect to the Microsoft Rights Management Services, this solution has the advantage that the publication phase can be performed while being offline, with no need to contact the remote trusted authority to obtain encryption keys or signed publishing licences. Information can also be encrypted several times for different authorities, so that the recipient must receive authorisation from all of them. Similarly, different parts of the same document can be encrypted for different authorities, each responsible for a subset of the protected information. Accountability is obtained by making TAs log all access requests and released decryption keys.

2.6 The Trusted Platform Initiative

The root of trust of all the solutions we have presented so far, i.e. the basic set of components in a system that can be trusted to behave correctly and that ensures all other components behave correctly, is represented broadly by two software components: the virtual machine running on the data recipient's machine that ensures data is used according to the data originator's policies, and the evaluation authority evaluating policies and issuing decryption keys to virtual machines. In this case the term *correct behaviour* must be intended with

respect to the expectations of both the data originators and recipients. In other words, a *correctly behaving* data protection system ensures that only users authorised by the data originator can access the data, while at the same time no information of the data recipients is used for purposes other than policy evaluation. The two components can be broken down further and identified with the set of components of the XACML reference model (Figure 2.7). It is easy to see how compromised components can affect the behaviour of the system and be detrimental for both data recipients and originators. A compromised PDP could return arbitrary results, granting access to data to unauthorised users or denying access to authorised ones. It could even disseminate the data recipient's private credentials it received for the policy evaluation. Similarly, a compromised PEP could release decryption keys (once received from an evaluation authority) or unprotected data, or not perform required obligations. A compromised PIP could hide certain credentials or generate false context information, while a compromised PAP could issue wrong policies. In both cases the PDP would be tricked into wrong evaluations. A compromised or malicious application receiving the disclosed data from the PEP could release it publicly.

As data protection systems offered to the public were continually affected by the problem of untrusted software environments, it became clear that new solutions were needed. Not only the components of the protection system had to be trusted, but the entire environment where they were installed and run, including the operating system and all of the applications running on it. In fact, most DRM systems were being circumvented by malicious applications capturing unprotected data as it was being disclosed (e.g. played by speakers or displayed by a monitor). **Trusted computing (TC)** [Gro, Sch03, Fel03] indicates a new family of technologies aimed to improve computer security through changes in existing hardware architectures. TC originates from two major projects (the Microsoft Next-Generation Secure Computing Base and the Trusted Computing Group, former Trusted Computing Platform Alliance) driven by a wide collaboration between original equipment manufacturers and software firms. Motivations come from the need to ensure the integrity of software components so that they can be trusted to enforce security controls correctly. One approach that has been traditionally adopted by most of the existing ERM systems is to manage and restrict administrative privileges on client machines so that users are not able to modify critical system components. However, it is hard to check or verify the integrity of remote systems without appropriate hardware support. TC provides this support by reducing a machine's *root of trust* to a single chip, the *Trusted Platform Module* (TPM).

TC encompasses five main functionalities, whose implementation or very availability depend on the specific TC product:

- **Endorsement key (EK):** every TC platform possesses a public/private key pair (although some manufacturers do not provide one, in the rest of this thesis we will assume platforms with an available EK)

that can be used for secure communications with external components or to sign data for authentication. The key is provided at manufacturing time and never leaves the TPM chip;

- **Memory curtaining:** sensitive areas of volatile memory are isolated and made inaccessible to untrusted or unauthorised software components;
- **Secure input and output:** protected paths are established between software/hardware components for secure data communication. Malicious or tampered components can be detected and excluded;
- **Sealed storage:** private information is bound to and encrypted for a specific software/hardware configuration. Cryptographic keys are not memorised but created at run-time according to the current execution environment. Configurations different from the original one cannot decrypt formerly secured information;
- **Remote attestation:** Authorised third parties can detect changes to a remote computer configuration and calculate a trust level for its run-time environment. The TPM can generate certificates specifying the applications currently running or installed.

The Trusted Computing Group's architecture [DR] is based on the existence of three *roots of trust*, i.e. three entities that must be trusted implicitly because there is no other way their trustworthiness can be explicitly verified. The trustworthiness of the remaining components in the architecture directly descends from one of the three roots of trust: the *core root of trust for measurement* (CRTM), the *root of trust for storage* (RTS) and the *root of trust for reporting* (RTR). While the three roots of trust are stored in the TPM's non-volatile memory, the TPM also includes 16 volatile *platform configuration registers* (PCR) that are used to store platform configuration measurements in the form of SHA-1 hash values of applications running on the platform or of configuration and data files. Measurements are stored by *extending* a PCR, i.e. by running the following operation:

$$\text{PCR} := \text{SHA-1}(\text{PCR} \mid \text{measurement})$$

Whenever a PCR is extended, the corresponding measurement is also stored in a *stored measurement log* (SML) that resides in the hard drive and is thus not protected by the TPM. The rationale behind this is that while the SML can be read to know what components are running on a platform, it is not possible to alter it since it would then not correspond to the stored PCR values that can be easily recalculated.

The CRTM is the BIOS boot block code. It measures and stores the integrity of other parts of the BIOS before passing control to them. The BIOS measures the hardware configuration of the machine and the integrity of the bootloader and passes control to it. Finally, the bootloader measures the integrity of the operating system's

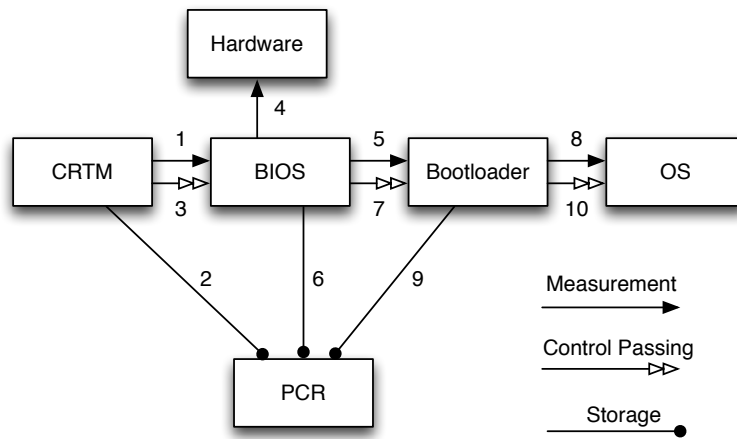


Figure 2.22: Integrity measurement process during trusted boot

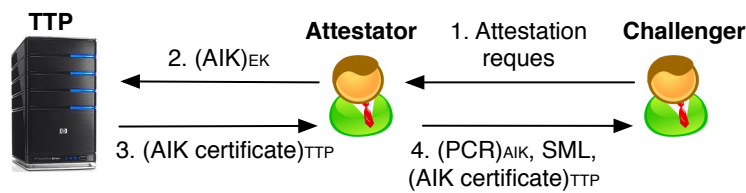


Figure 2.23: Attestation process

kernel. In the end, all components running in the machine up to the OS level are measured and the measurements stored in the PCR. Note that this *trusted boot* process (Figure 2.22) does not prevent untrusted components from being run, but memorises them so that *remote attestation* is possible.

Before starting an interaction, a remote party (the challenger) might require an attestation of the platform's integrity. To do so, the attestator must send to the challenger its SML and PCR values. The challenger can then first verify if the attestator has an untrusted configuration by comparing the SML records with a local database of trusted and untrusted components, and then calculate their hash values and compare them with the received PCR values. To ensure that such values have not been tampered with, they should be signed by the attestator's TPM with its EK. Since the EK is uniquely linked to the machine it belongs to, this process might disclose the attestator's identity. To avoid this, the attestator's TPM generates an *attestation identity key* (AIK), signs it with the EK and sends it to a remote *trusted third party* (TTP), usually an hardware manufacturer. The TTP verifies that the AIK has been signed by a key belonging to a valid TPM (e.g. not included in a revocation list), and generates and signs an AIK certificate that the attestator can then send to the challenger, together with the SML and PCR values signed by its TPM with the AIK. The challenger can now verify if the attestator is trustworthy. The EK thus works as the RTR (Figure 2.23).

Sealed storage is realised by generating a hierarchy of encryption keys whose root is the *storage root key*

(SRK) that works as the TRS, intermediate nodes are RSA keys (*storage keys*) used to encrypt their child keys in the hierarchy and leaf nodes are RSA keys (*binding keys*) used to encrypt symmetric keys. The AIK is an example storage key encrypted by the storage root key. Data can then be protected with a simple symmetric key that can be moved to a different machine, or can be bound to a specific TPM platform or particular configuration. This is realised by generating a binding key that is directly bound to a set of PCR values, so that the TPM releases it only if the current PCR values are the same.

Several proposals have been made to extend this basic architecture. Maruyama *et al.* [MSN⁺04] proposed a trusted boot process to load a valid operating system capable of enforcing mandatory controls (e.g. SELinux). With its Next-Generation Secure Computing Base (NGSCB) [ELM⁺03] Microsoft proposed to separate a platform into a trusted environment and an untrusted environment, so to provide robust access control while retaining the openness of traditional architectures. Applications that need to protect themselves or their data can thus run either entirely or in part in the trusted environment, while other applications can run as they normally would. The integrity measurement architecture (IMA) proposed by Sailer *et al.* [SZJvD04] realises remote attestation by measuring all the programs and code before they are loaded into the system to run.

2.6.1 Trusted Computing Applications

TC has a number of different applications, from hard drive encryption to ERM/DRM systems. Sandhu *et al.* [SRZ06] proposed a framework of three layered models to analyse requirements for data protections in the context of TC. The three layers are *policy models*, *enforcement models* and *implementation models*, hence the name *PEI* models (Figure 2.24). Intuitively, any of the solutions presented in Sections 2.3 and 2.4 could be used as a policy model while any of the architectures and specific products presented in Sections 2.5.1 and 2.5.2 could be used as enforcement and implementation models, respectively.

The different implementation models can then be deployed on top of a TC platform that guarantees the correct behaviour and integrity of the running components. An example architecture for the enforcement of access control rules exploiting a TPM chip was proposed in [SZRC06]. As the authors emphasise, TC is particularly important when enforcing dissemination controls, as it makes it easier to determine the identity of a machine through its TPM and AIK and thus decide whether it should be allowed to receive a disseminated object. The proposed solution is based on an extension of the trusted boot process. After the OS kernel has been loaded and launched, it verifies the integrity of a component running in user space called *trusted reference monitor* (TRM), equivalent to the virtual machine of traditional IDSA solutions. Both the OS and the TRM are assigned a public/private key pair by the TPM. The OS's pair is signed and sealed by the platform's AIK and signs and seals in turn the TRM's pair. The TRM acts as a safe wrapping around data that grants access to

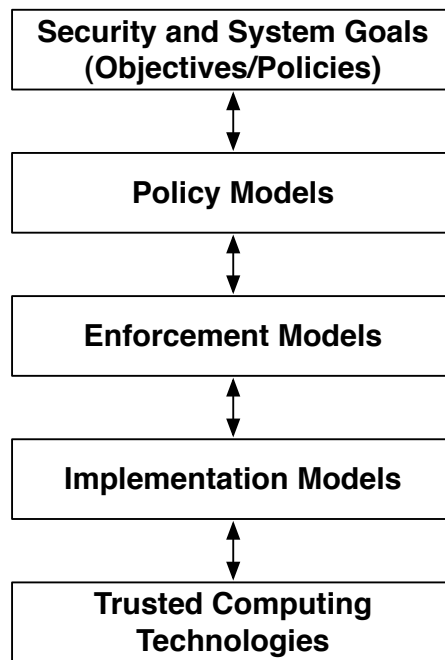


Figure 2.24: The PEI models framework

applications. Whenever data access is requested by a user to a remote party, remote attestation is performed to verify not only the TPM's and OS's integrity, but also the integrity of the TRM. If the verification succeeds, the remote party encrypts the data for the requesting user's TRM and sends it together with a set of access control policies. These contain the list of integrity measurements the requesting user's machine must satisfy for the TRM to release access to the application.

In other words, TC can be used as a means to ensure a virtual machine or reference monitor properly protects data and accessing applications properly handle it. Problems can arise in open settings where different applications or versions of the same application can be used to access data and universally valid integrity measurements are difficult to define. More complex forms of remote attestation such as property-based attestation [SS04] and dynamic property-based attestation [NV10], which focus on the certification of high-level properties of a platform that can be deduced by its low-level configuration, have been proposed to address this issue. Also, if different organisations that use different certification authorities and trusted third parties take part in the data sharing, there might be problems in the validation of key certificates and integrity measurements. However, this is not a problem if we assume that all users access data via the same application and use the same certification authorities, as we did in Section 3.3. We also emphasise that while TC ensures components behave as expected, it cannot control users' behaviour, especially when users can influence the operations a trusted application is performing (e.g. via a command line). Trust on users depends on what actions they can perform. In other words, there are cases when release of sensitive data cannot depend solely on the verification of a

TRM's integrity, but also on what kind of trust can be placed on the TRM's final user.

2.7 Summary and Conclusions

In this chapter we have presented the three main components that constitute modern data protection systems: policy models and languages, enforcement mechanisms and trusted platform technologies. The works presented serve a two-fold purpose. First, we introduced the basic concepts the rest of this thesis draws from: information flow, trust management and policy languages for authorisation specification. Second, we pointed out some of the main limitations of the presented approaches. While existing policy languages allow the specification of a wide variety of policies, they do not allow data originators to express how such authorisations may be inherited by derived data. Policies are static in that once they have been associated with data, they cannot be modified to satisfy the protection requirements of entities whose data is changing. Similarly, policies cannot transit from the original data to its derivations.

We then presented some of the existing enforcement mechanisms and architectures and pointed out their shortcomings, so to define the information security scenario to which this thesis contributes. We distinguished in particular between interactive data sharing architectures (ISDA), that cannot be used when connectivity is not available, and non-interactive data sharing architectures (NISDA) that cannot enforce context-based conditions or cannot work with users outside the organisation distributing the permissions/keys. This distinction is particularly important to us since rather than using NISDA architectures for policy enforcement, we chose to adapt ISDA architectures by merging them with trust management solutions so that they can be used in partially disconnected networks.

The high-level aspects of the architecture we propose, together with our preliminary work on trusted computing technologies, is described in the next chapter.

3 Architecture

3.1 Chapter Overview

This chapter aims to describe the basic components of the architecture at the basis of our solution and to highlight which novel elements it includes. These will be further investigated in Chapters 4 and 5 where we will also progressively build proof of concepts of the design presented in this chapter. In Section 3.2 and Section 3.3 of this chapter we specify the threat model and assumptions we considered for our design. We then give a general overview of the data protection architecture in Section 3.4. Finally, in Section 3.5 we describe the integration of the architecture with trusted computing technologies.

3.2 Threat Model

3.2.1 Protection Objects

While the main objective of data protection systems is to ensure the confidentiality and integrity of disseminated data, we must also consider the following other information objects involved in the primary protection process and requiring protection:

- **Metadata:** Data is usually associated with descriptive information that can be used by both data recipients and by policy decision points when making authorisation decisions. Integrity of the associated metadata needs to be guaranteed as compromised metadata can greatly affect the decision process. For example, if metadata contained the ID of the object's owner and this were modified, a possibly unauthorised subject could get access to the data. Moreover, in many scenarios metadata may be considered confidential as it can reveal information on the data content.
- **Policies:** Policy integrity must be guaranteed to ensure correct authorisation decisions are taken for every request. Policies can also be considered confidential as they may contain the identity of users authorised to access the data or may be used to obtain the needed credentials in non-legal ways.

- **Credentials:** As for metadata and policies, integrity of credentials used in the policy evaluation process must be guaranteed. While we deem the use of digital signatures in traditional certification standards (e.g. X.509) sufficient to verify credentials' integrity, we consider a different problem. Users' identity and attribute credentials may in fact be considered private and should thus be protected when being sent from a data recipient (or credential provider) to a trusted policy decision point.

3.2.2 Adversary

The overall trust model considered in this thesis is that of a *trusted data originator* who needs help in protecting her resources, but also of a *potentially malicious recipient* - i.e. a user that, after data reception, might try to use it in violation of the originator's policies. Our solution is thus aimed at proposing a non-circumventable information security mechanism that data originators can rely on when disseminating data amongst unknown and potentially malicious recipients. These include malicious users in honest organisations (i.e. organisations participating in a DSA), curious but not malicious users that can attempt to stray from the intended data usages if the opportunity arises, and malicious unknown users (i.e. users that do not belong to any organisation participating in a DSA). We *do not consider* the case of a malicious organisation agreeing and participating in a DSA, as most of the time decisions on the amount of trust that can be put on high-level organisations rely on controls put in place by traditional legislative systems or on the reputation, brands and ongoing relationships of the involved entities.

Besides malicious users, when dealing with changing data we must also consider the case of *honest but incompetent recipients*, i.e. recipients who can access the data and modify it, but that cannot be trusted to perform correct modifications or to specify correctly what policies the new data should be associated with. This is a particularly important aspect, as many of the security attacks nowadays mainly exploit users' mistakes or naïvety or conversely the provision of unusable security systems. A wrongly specified policy can cause a data leakage as much as a weak password or a weak cryptographic key. Also, while *we do not consider* malicious data originators, we must consider *honest but curious data originators*, i.e. originators who attempt to learn potentially private details on users by disseminating protected data and forcing recipients to provide their credentials for policy evaluation.

Note that this adversary model is asymmetric, as it does not consider malicious organisations falsely accusing data recipients of misusing disseminated data. This aspect of the mutual relationships between data originators and recipients is outside the scope of this thesis.

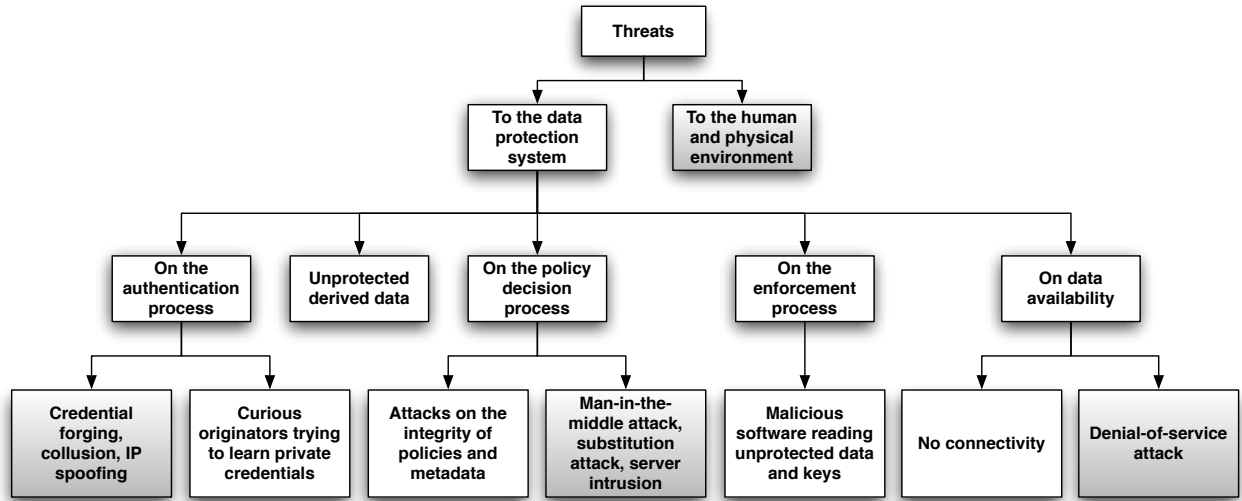


Figure 3.1: Attack tree for data protection architecture. Shaded attacks are dealt with by mainstream products. Non-shaded attacks are dealt with natively by our proposed architecture

3.2.3 Threats

Figure 3.1 depicts the indicative set of threats we considered when designing our architecture, where we consider as threats all the attacks on the protection system that users can perform to obtain unauthorised access to data. These were chosen as the most likely attacks used nowadays by users trying to compromise existing DRM and ERM systems or to gain information they would normally have no access to. Malicious users can attack data protection systems by either trying to compromise the authentication process, the policy decision process, the enforcement process or the system’s availability. We emphasise that in this thesis we do not deal with threats that aim to attack human vulnerabilities of an organisation rather than its information system such as bribery of employees, industrial espionage etc. Such threats must still be dealt with to guarantee data confidentiality, integrity and availability through the provision of a safe operational envelope comprising physical controls, human training and best usage practices within which the automatic protection tools can operate.

Attacks that try to compromise the authentication process (e.g. credential forging, collusion, IP spoofing etc) have been the focus of intensive research and we will not deal with them in this thesis. A different attack we consider instead comprises curious data originators trying to learn private details on data recipients when evaluating their access requests. This can be achieved by disseminating protected data whose publishing server (in the case of Microsoft RMS, described in Section 2.5.2) or policy evaluation authority is under the originator’s control and can thus disclose the content of the received credentials.

The policy decision process can be compromised by either modifying the policies or the metadata needed during the evaluation. In both cases, the attack type depends on how policies and metadata are stored. If

policies and metadata are stored on a server, the attack can be performed by compromising the communication between the server and the data originator at publishing time (e.g. with man-in-the-middle attacks, substitution attacks etc.), or by attacking the policy and metadata store directly. Otherwise, if the policies and metadata have been attached to the data and are disseminated along with it, the attacker can try to modify them directly. Many security protocols and cryptographic techniques have been proposed to both attach policies and metadata to data and to protect their integrity. However, since we propose a new data protection and policy decision protocol, we also described how the integrity of policies and metadata is protected.

The most common attack on the policy enforcement process is based on installing malicious software in the same environment where the PEP (see Section 2.4.1) or virtual machine run. Eventually, data must be streamed without protection to authorised recipients, to be either displayed on a video, played on speakers and so on. During the streaming, malicious software can capture the streamed data and save it back with no protection. Similarly, malicious software can try to read data decryption keys once they have been downloaded to access received data rightfully. Another attack consists in substituting critical software components in the enforcement architecture with malicious ones (that for example do not force the recipient to perform obligations, or do not enforce continuous controls on data accesses). In general, these attacks all deal with the inner vulnerabilities of a software environment over which malicious data recipients have complete control and in which policies must be enforced. While in traditional ERM systems deployed in corporate environments users have no control over the machines they use, which are rather controlled by a trusted administrator, we also consider the case of malicious data recipients acting outside of any trusted organisation and who thus maintain control over their own platforms. These users could install any malicious software beside the virtual machine needed to access data and thus bypass the data protection. Therefore, these *untrusted software environments* constitute a problem we must deal with. This is why a trusted computing platform is required to ascertain the integrity of the enforcement.

Availability can also be compromised by adversaries in a number of ways (e.g. denial-of-service attacks). This traditional type of attacks have been extensively studied. Otherwise, availability could be compromised when users retain data and refuse to exchange it. Cryptographic protocols do not deal with Denial of Service attacks caused by retained or not exchanged messages, and we do not try to detect message loss as communication protocols already deal with it. However, we must still deal with the case where data unavailability is not caused by external attacks, for example when remote components of the overall data protection system (such as publishing servers or credential providers) are not reachable because connectivity is not available. This is an intrinsic condition of the dissemination environment rather than the result of a malicious attack, but it is still a risk to the system's operations.

Finally, we consider a particular set of attacks that can be performed when no control is enforced on data derivation. If a user is authorised to modify some data she received, she might introduce sensitive information that requires a higher level of protection than what the original policies provide. Similarly, a user may create new sensitive data and not associate it with any policy at all, or with insufficiently strict policies. She might even specify policies that do not allow anyone to access the disseminated data, thus making it useless. In other words, a wrong policy specification can lead to either data unavailability or to leakages of sensitive data. This is due to the discretionary aspect of policy specification, whereas in corporate environments mandatory policies should also be specified at organisational level.

3.3 Data Model

Contrary to those who conduct information flow studies, we do not make any assumptions about the means through which data is disseminated or locations where the data can be received. Once the data leaves its originator device, it becomes impossible to predict the paths it may follow, i.e. it is impossible to predict the behaviour of recipients. Recipients can re-disseminate the data in different ways (via USB keys, emails etc.) and to recipients the originator may not know. However, we do assume a “homogeneous environment” where every user is equipped with the necessary hardware and software to access the data, as described in the remainder of this thesis, and where all recipients share a common vocabulary/ontology and trust an agreed set of authorities for the certification of identities and basic attributes, so that signed credentials (e.g. X.509 certificates) can be universally interpreted and verified.

While data is encrypted and can be freely transmitted between users, there are specific moments in our protection protocols where decryption keys are exchanged and must be protected (e.g. after a policy has been positively evaluated). In these cases, the communication is considered to be secured for both integrity and confidentiality via standard protocols (e.g. a Diffie-Hellman exchange or other protocols). In scenarios where connectivity is not available (as in the crisis management scenario described in Section 1.3) and credentials cannot be verified with their issuers, we consider a validity time frame (e.g. of the duration of the crisis at hand) during which revocation lists are assumed not to change (even if we propose an approach for these situations in Section 5.3.3). For these scenarios, we do not make any assumptions about the short-range communication technologies that can be employed (e.g. bluetooth, active RFID etc). Communications that need to be secured (e.g. to exchange keys) could thus use the specific technology’s security protocols or higher level protocols built on top of the technology’s basic communication features.

Despite the generality of the approach we propose, the examples we provide will focus on the protection of

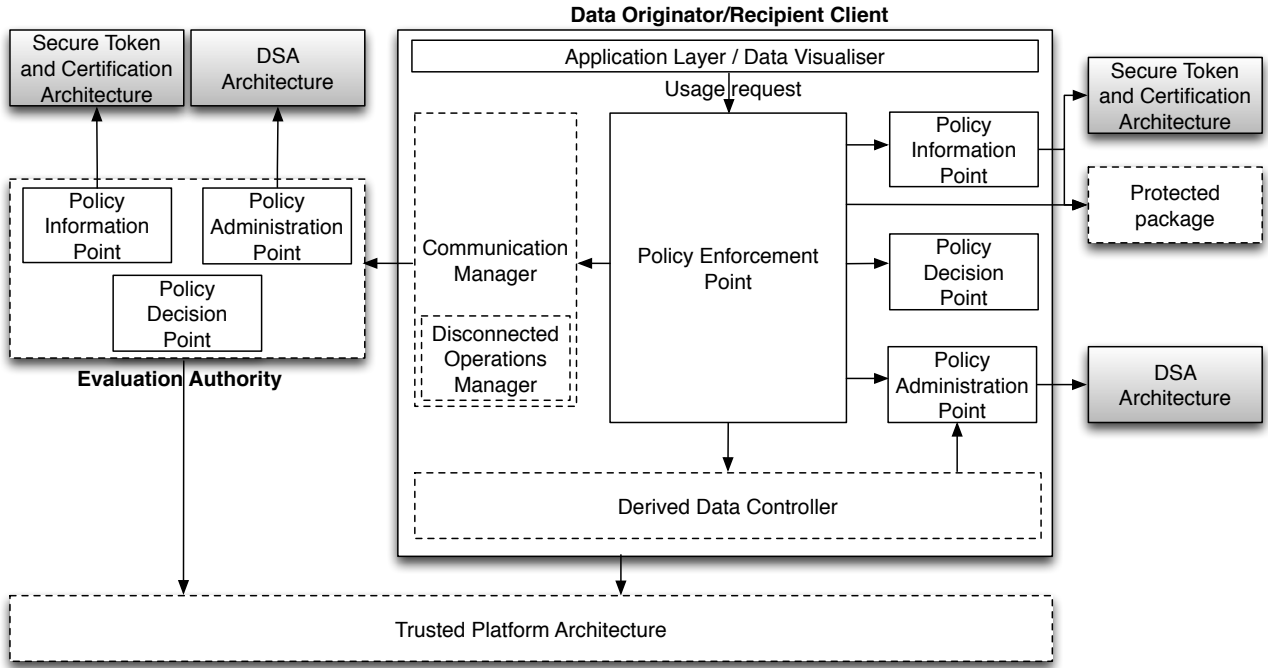


Figure 3.2: High level architecture of the proposed data protection solution. Arrows represent logical dependencies and not actual exchanged messages

XML documents. XML has been chosen because of its generality (many data models can be translated into an XML structure), the existence of several development tools and because it allows us to reason about the effects that associating different policies with different parts of the same data structure has on data protection.

3.4 Architecture Overview

Figure 3.2 depicts the high-level data protection architecture we propose to enable usage and derived data control and usage control in partially disconnected networks. The dotted components implement the novel contributions of this thesis. On one hand, we introduce new components (for example, with respect to the XACML reference architecture) such as the *derived data controller* and the *disconnected operations manager*. On the other hand, we propose new implementations or integration of already existing components such as the *policy decision point* located on remote *evaluation authorities* or the *trusted platform architecture* components. Shaded components represent aspects of the architecture that are outside the scope of this thesis, meaning that we will only describe their expected behaviour but we will not provide any details on their possible implementation. In the following we introduce the well-known components, their interactions and how trusted computing technologies have been integrated into the architecture to ascertain the integrity of the recipient's enforcement. The arrows connecting the components in Figure 3.2 represent logical dependencies and not actual exchanged messages.

In the remainder of this thesis we will use the terms *client* and *virtual machine* interchangeably to indicate the set of interconnected components installed on user devices and with which applications interact to access the data. Clients interact with remote services (the secure token and certification architecture, the DSA architecture and evaluation authorities) and with the protected data package disseminated amongst users. As we will see later, the protected package is not an actual component as it is unable to perform any computation. However, it contains all the information needed to access the data and evaluate the applicable policies. This is why we explicitly represent it in Figure 3.2. We depict the trusted platform architecture as a stand-alone component as it is completely transparent to the rest of the architecture, meaning that its removal does not compromise the operation of the other components. However, we stress that removing it implies that users can and must be completely trusted to correctly enforce policies associated with the data they receive and use.

3.4.1 Data Sharing Agreement Architecture

Disseminated data often crosses organisational boundaries and needs to be accessed by users belonging to organisations different from that of the data originator. To guarantee that usage controls can be enforced in all organisations where data can be received, it is important that the organisations exchanging data agree on a common policy vocabulary so that policies and identity and attribute credentials can be interpreted in the same way in every organisation's environment. Since organisations must specify policies that allow recipients to access their data, they need to know how to refer to such authorised recipients when they belong to other organisations. For example, they may need to know the role structure of the other organisations. This is also why organisations must agree on a common set of certification authorities for identity and attribute certificates. Organisations also need to know how their data will be used once it has been received by other organisations. Not only who can and should access it, but also how the data will be processed. In other words, organisations must agree on the set of procedures through which data can be accessed and transformed. Similarly, originators need to know what obligations recipients in other organisations can perform, so that they can write policies accordingly. For example, originators may require recipients to notify them of each access performed on data. Such requirements and policies must be agreed by all organisations so that controls can be enforced properly and so that no arbitrary policy is specified that causes data to be unusable for some organisations that need it. Organisations must also agree on a set of trusted evaluation authorities, i.e. on a set of entities they all trust for the evaluation of policies. Each organisation may be required to provide its own evaluation authorities, or third-party services could be used. A data sharing agreement is a necessary tool for organisations to agree on these details and write policies valid throughout all environments where data could be disseminated.

The **data sharing agreement architecture** is the set of software services that allow organisations to agree

on a common set of usage control policies and that make such policies available whenever required. A DSA is expected to be in place prior to any data dissemination as it contains the constraints under which data dissemination and usage can be carried out.

Besides the standard legal notation referring to the relevant legislation regulating data secrecy, a typical DSA must specify: 1) the operations, objects and subjects which are under the DSA's regulation, 2) the security policies (or vocabulary to write them) and 3) the penalties for violating the agreement. Figure 3.3 depicts an example scenario where the police force and ambulance services agree on sharing data. They first sign a DSA containing all the policies regulating the data exchange. If any policy is violated and the violation is detected, both the partners can then rely on more traditional law enforcement mechanisms to solve their disputes. DSAs may also include additional specifications, such as procedures to resolve disputes, to revise the agreement etc. In this work we focus on the core DSA components. Discussion of further extensions, detection of violations and penalties are beyond the scope of this thesis. While mechanisms to recover from violations are left for future work, our proposed solution does consider *break-the-glass* policies where violations are allowed by design, for example when a privileged rescuer deems data access more important than data protection and disseminates the data to users that would not normally be authorised (see Section 5.4).

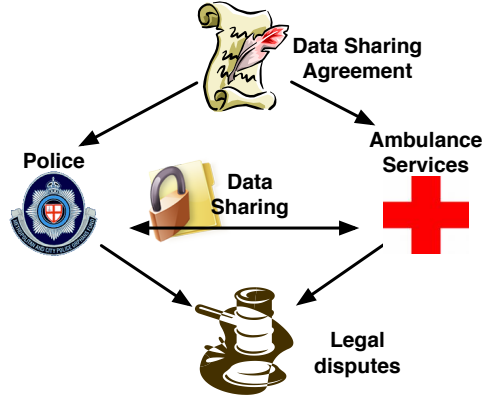


Figure 3.3: Data Sharing Agreement for the crisis management scenario

Definition 3.1 A data sharing agreement is a 7-tuple $(ID, S, A, EA, CA, UP, V)$ where ID is the unique identifier of the agreement, S is the scope of the agreement, A is an attribute vocabulary describing subject, data, and context attributes, EA is the specification of the trusted policy evaluation authorities, CA is a set of secure token and certification authorities, UP is the specification of the agreed usage control policies and V is a set of violation procedures applicable whenever the terms of the agreement are breached by one of the partners. The scope S is defined as a triple (E, t_s, t_t) where E is the set of all partner organisations signing the agreement and t_s and t_t are respectively the start and end dates of the agreement, specifying the time period

during which the agreement is considered valid.

For the moment we will not delve into the details of how both the set of evaluation authorities *EA* and usage control policies *UP* are specified, as this is part of the solutions we describe in the next chapters (Chapter 4 introduces the notation and format we use to specify usage control policies, while Chapter 5 describes how evaluation authorities can be specified). When an entity agrees to a DSA it promises to enforce the policies contained in *UP* for the data provided by the partner entities. This ensures that the security requirements of the data originator are satisfied. Given our assumption that credentials are universally interpreted and understood, the common vocabulary *A* is sufficient to avoid the need for a translation from high-level DSA policies to low-level and organisation-specific enforceable policies. In other words, once the set *UP* has been specified on a DSA, all partner organisations can equally evaluate and enforce the policies. Note that despite the fact that we are here considering a DSA containing the set of usage control policies *UP*, this is not strictly necessary if the organisations agree on allowing their individual originators (e.g. employees, teams etc.) to specify their own policies for their disseminated data, independently. In this case, the DSA would only contain the details necessary to properly interpret policies (such as the vocabulary). If an organisation agreed to such a DSA, it would promise to enforce any policy associated with the data by its originator.

Therefore, the simplified DSA architecture we consider consists of just a set of remote repositories where details on the DSAs can be stored and that clients and evaluation authorities can poll at start-up and periodically later to retrieve the latest available policies. The **policy administration point (PAP)** in the client architecture is responsible for querying the DSA repositories, keeping a local copy of the current policies and DSA and returning them to the other components when required for policy evaluation. The updates of the available policies can be either pushed by the DSA repositories to the clients at regular intervals, or pulled by the clients at every access request. The choice of the best approach depends on the specific application scenario. A push approach is for example preferable when updates are performed only at specific and predefined intervals, but would require a Service Level Agreement between the DSA repository and the involved organisations to guarantee the updates are performed whenever necessary, so that no stale policy is ever evaluated. To reduce the overhead and the number of communications necessary to disseminate the new policies, the repository could for example analyse the policy updates and determine their impact and scope, so to decide whether distribute them and to whom. On the other hand, a pull approach guarantees that policies are always correct, but has a cost in terms of non-necessary communications whenever no new policies have been changed.

We emphasise that while DSAs are useful to define how data is shared, used and protected amongst partner organisations and to establish a common vocabulary for the policy specification, policies can be enforced also in the absence of a DSA and we do not assume a DSA to be necessarily present in the scenario considered in



Figure 3.4: High-level interaction for data publication

rest of this thesis. When users with no pre-existing agreement meet and share data, this is protected with the policies each individual user has available. If data from the two users is merged, the new data is protected by a set of policies that equally results from the merging of the two users' policies. More details are given in the next two chapters.

3.4.2 Data Publication

Data publication is the process of protecting data before it is disseminated, so that only authorised recipients can access it. The architecture we propose is capable of working in different modes of operations so that it can cater for both scenarios where connectivity is available and scenarios where it is not. This condition is periodically checked by the **communication manager** that can then switch between connected and disconnected operation modes. In the following we will describe how connected operations are carried out, while disconnected operations will be covered in Chapter 5.

The central component of the architecture is the **policy enforcement point (PEP)** that acts as an interface between the applications used to access data and the data itself. Figure 3.4 depicts the high-level interaction between the data originator's client and the external services needed for data publication while Figure 3.5 depicts the message exchange triggered by a publication request. The PEP first queries the communication manager on the state of the connection. If connectivity is available (as we assume for now) the PEP asks the PAP to obtain the latest DSA updates from a remote DSA repository. It then encrypts the data (and metadata), generates a publication license containing the decryption keys and attaches to it the data and metadata. The details on how this is actually realised will be described in Sections 4.7.2, 5.3.2 and 5.3.3. The licence and the encrypted data can then be disseminated amongst recipients.

3.4.3 Policy Evaluation and Enforcement

All requests to access and use published data are forwarded from a client application to the PEP that coordinates the other components to return an authorisation decision and disclose (i.e. decrypt) the protected data. Figure

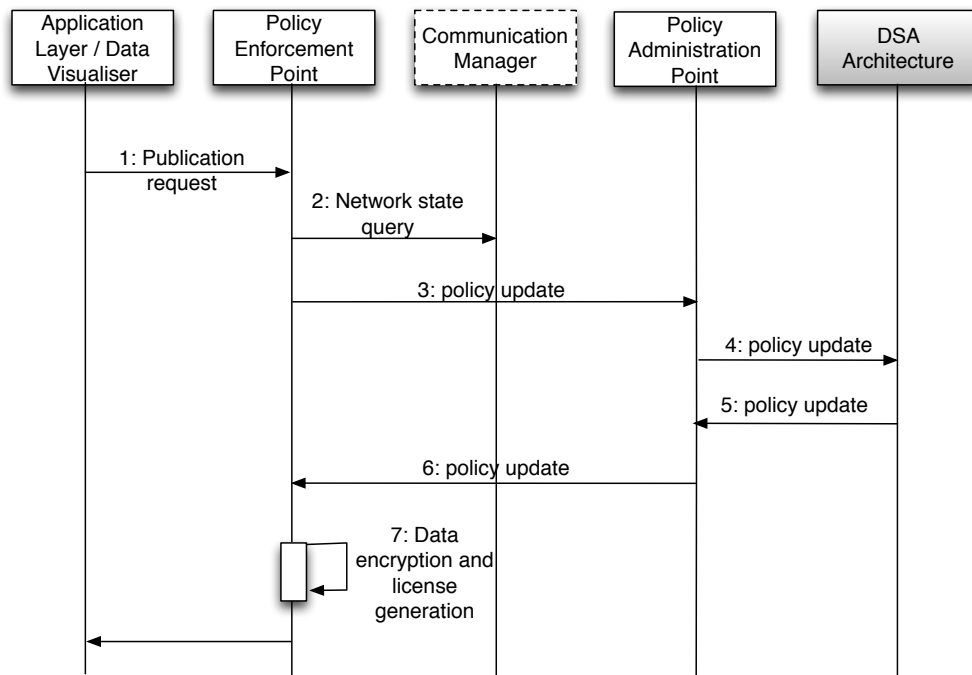


Figure 3.5: Interaction of components for data publication



Figure 3.6: High-level interaction for policy evaluation

3.6 depicts the high-level interaction between the data recipient's client and the external services needed for policy evaluation while Figure 3.7 depicts the message exchange triggered by usage requests. The PEP first queries the communication manager on the state of the connection. If connectivity is available (as we assume again) the PEP asks its local **policy information point (PIP)** to access the data package's metadata and publishing licence and sends them along with an evaluation request to an evaluation authority. In Chapter 5 we will describe how different evaluation authorities can be selected. Once it has received the usage request, metadata and publishing licence, the remote evaluation authority's PEP queries its PAP for policies applicable to the request (depending on the attached license) and returns to the client the list of necessary subject credentials. The client's PEP then queries its local PIP and returns the required credentials (or part of them) to the evaluation

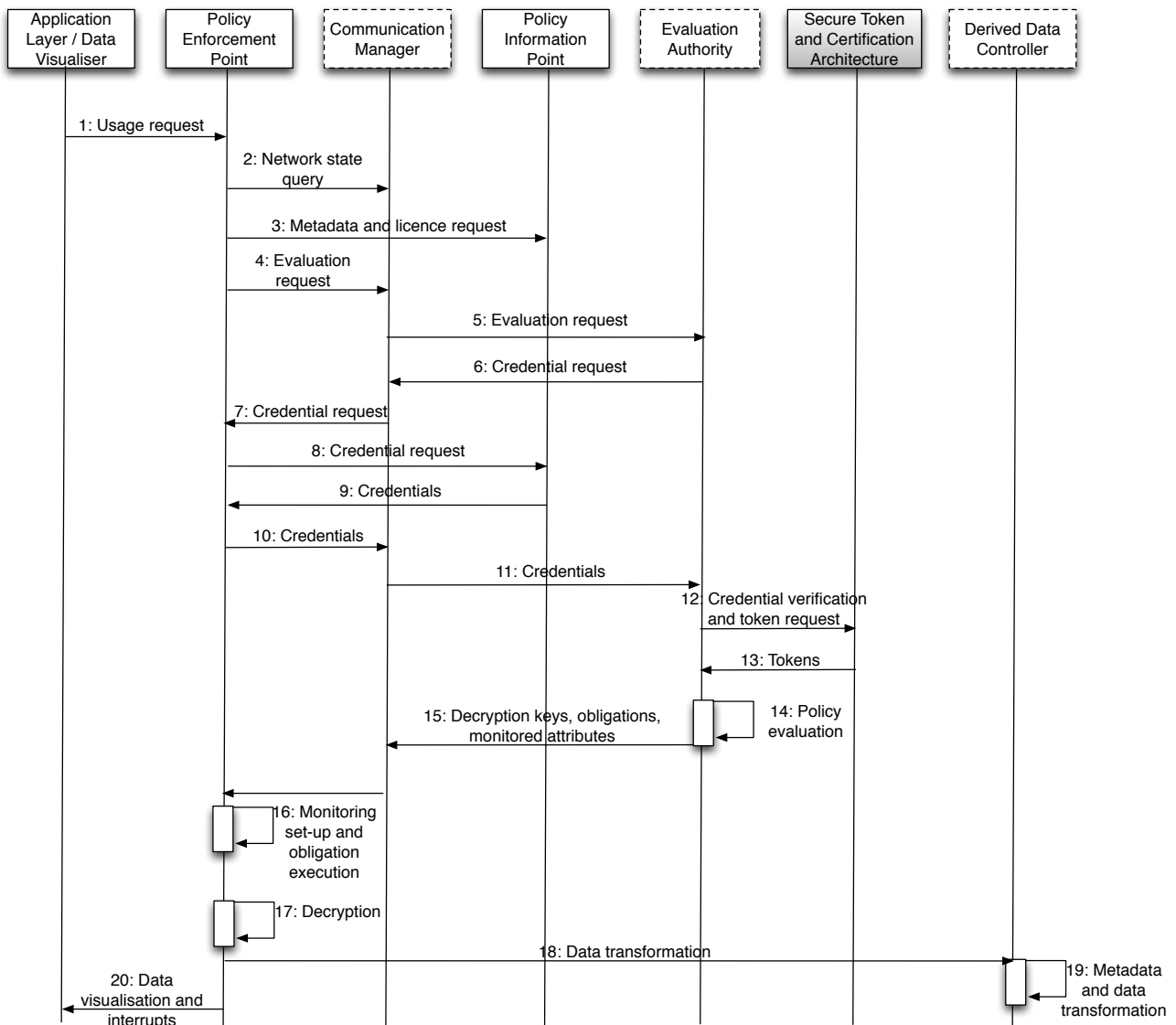


Figure 3.7: Interaction of components for policy evaluation

authority. The authority's PIP can then query the **secure token and certification architecture** to verify the received credentials, revocation lists and to retrieve tokens containing the context attributes or missing user credentials required by the evaluated policies. Once all necessary information has been retrieved, the authority's **policy decision point** can evaluate all the policies applicable to the request and return an authorisation decision. If the authorisation is positive, then the evaluation authority returns to the client the list of attributes that must be monitored during the use, the set of obligations to be performed and the decryption keys necessary to access the data package (or part of it). An internal thread of the client's PEP is then instructed to check the attributes to be monitored periodically, so that a local policy evaluation (i.e. an evaluation performed by the client's local PIP, PDP and PAP) can be triggered periodically (e.g. whenever an attribute changes) for the policies whose constraints refer to monitored attributes. If any new authorisation decision is negative, the current use is

interrupted. In this thesis we consider obligations as procedures performed by the PEP upon a request (as for XACML [xac05] and Ponder [DDLS01]). In other words the PEP simply executes a set of internal well-known procedures such as sending a notification e-mail or executing an online payment. After obligations have been performed with no errors, the PEP can decrypt the data package with the received keys and return the disclosed content to the requesting application. If the requested action causes the data to be transformed, the **derived data controller** component will take care of transforming the data and modifying the associated metadata so that new policies (that satisfy the new data's security requirements) will become applicable. We will describe the derivation process in Chapter 4.

The PEP is also responsible for keeping track of the current *access sessions*. Several subsequent evaluations of the same request might in fact be unnecessary if the interval between one request and the other is short (depending on the system's configuration). Examples are several subsequent "save" actions on unmodified documents. An access session is the time interval during which only one policy evaluation is required for the the same access requests, i.e. for requests to perform the same action on the same object. The duration of the access sessions is a parameter that depends on the PEP configuration and that can be specified in the DSA.

3.5 Trusted Computing Integration for Policy-Driven Remote Attestation

A policy evaluation that returns a positive authorisation for a usage request does not guarantee by itself that the policies will actually be enforced once decryption keys have been released to the requesting recipients. Correct enforcement depends on the integrity of the virtual machine's components and processes running on recipients' devices. As we described in Section 2.6, verifying the trustworthiness of the accessing applications, as well as of the entire software and hardware environment where data is received, is pivotal for data protection. To this end we integrated our data protection architecture with trusted computing technologies in general and remote attestation mechanisms in particular.

As described in Chapter 2 remote attestation requires that a trusted verifier is able to verify the integrity of a software environment based on a digitally signed list of hash values. The process of verifying this evidence requires the verifier to maintain a large up-to-date database of acceptable software components. However, a simple Boolean decision regarding the presence of untrusted software running on a machine may not be enough to verify its integrity. For example, "acceptable" behaviours may depend on specific compositions of system components and their versions. Or, a data originator might not require a trusted platform verification, preferring easy and fast data dissemination over protection (as it may be the case in emergency scenarios) or deeming simple policy evaluation sufficient. Considering these requirements, usage control policies should

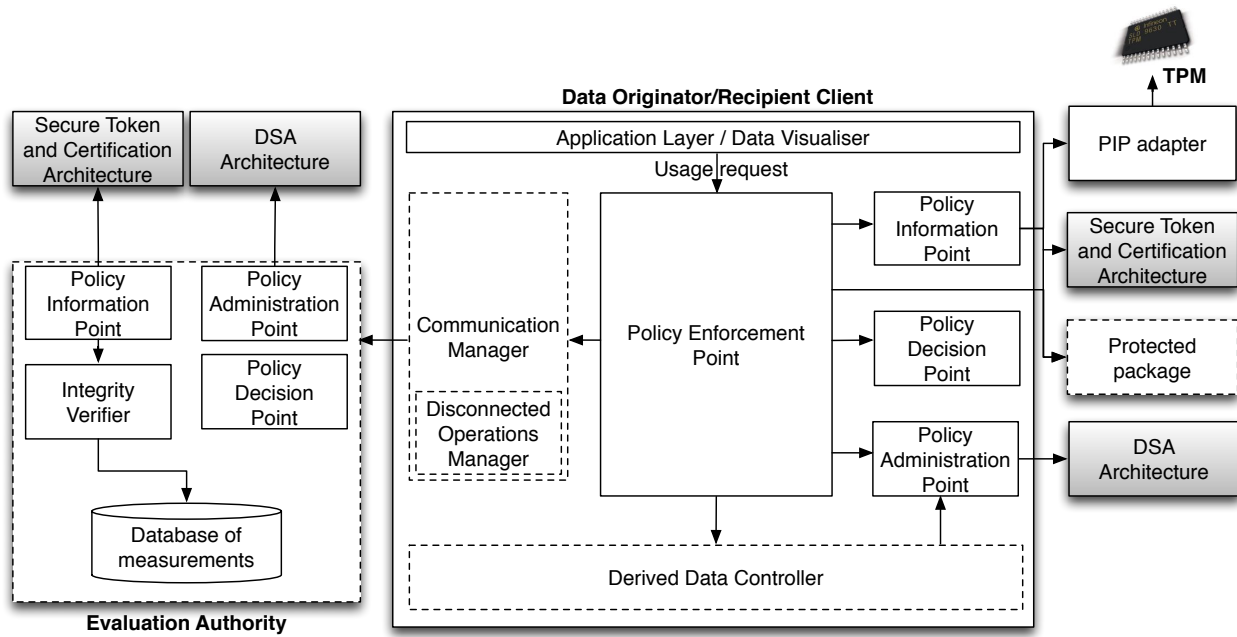


Figure 3.8: High level architecture for policy-driven remote attestation

be able to specify whether remote attestation is required or not and, if required, which conditions should be satisfied by system components.

Figure 3.8 depicts the high-level data protection architecture with the actual components introduced for remote attestation. The **PIP adapter** is a daemon running on the client machine and acting as a “listener” service that waits for requests on a particular port. It interacts with system-level libraries that measure the system’s state using the TrouSerS (trusted computing software stack) API for accessing the various functionalities of the TPM. In particular, the trusted platform module takes measurements according to the trusted boot process and integrity measurement architecture (IMA) described in Chapter 2. When a policy explicitly requires a measurement from the TPM, the PIP fetches it from the PIP adapter and forwards it to the evaluation authority as a credential. It consists of:

- TPM quote (i.e. a signed composite hash of the selected platform configuration registers, PCRs);
- TPM credentials;
- values of selected TPM PCRs (chosen depending on what the policy wants to check - this includes PCR-10);
- the IMA measurement list (i.e. the list of all the executable and application-related files loaded).

We entrust evaluation authorities to act as attestation authorities as connecting to specialised verifiers could be difficult when connectivity is not guaranteed. As we will see in Chapter 5, this also means that every

client machine must be able to act as an attestation authority for other clients and thus store an up-to-date list of the acceptable and expected hash values of various programs. When the evaluation authority receives such a credential, it passes it to its local PIP for further evaluation. The **integrity verifier** verifies the TPM quote using the credentials provided for the TPM and the received PCR values. In other words, it verifies the signature on the PCRs and uses the IMA measurement to re-compute the value of PCR-10 by aggregating all PCR values. The re-computed value is then compared with the received value to verify the IMA measurement list was correct. Finally, the hash values of the applications (and their versions) in the verified IMA list are checked against the available database to verify that they can all be trusted. Depending on the outcome of the verification process, the integrity verifier returns to the PDP an attestation certificate with the required credentials (e.g. specifying the version of certain libraries, the running applications etc.), as specified by the evaluated policy. The certificate can then be used like any subject or context certificate received by the secure token and certification architecture or directly by the recipient's client. The same approach could also be used in combination with more complex forms of remote attestation such as property-based attestation [SS04] and dynamic property-based attestation [NV10]. We here propose the basic policy-based attestation mechanism that allows a policy to specify constraints on the specific components in the attested platform. Extending this mechanism to allow policies to specify constraints on a platform's *high-level properties* [SS04] would be straightforward.

Using this approach, trusted platform verification is treated as an optional requirement to be included into usage policies and that depends on the discretion of data originators or, in our case, of the authors of the DSA. If policies do not require trusted platform verifications, the PIP adapter and integrity verifier could be removed from the proposed architecture with no effect on the rest of the components, as if they were external certification authorities. In other words, the main advantage of this approach is that it completely separates the policy enforcement components and architecture from the trusted platform components, allowing greater flexibility in the policy specification and enforcement.

3.6 Summary and Conclusions

This chapter mainly served the purpose of “setting the scene” for what is described in the rest of this thesis. First, we described the high-level threat model and basic assumptions we considered when developing our architecture. We believe the assumptions we considered are realistic and do not hinder the validity of our work. We then introduced the concept of data sharing agreement (DSA). Despite not being the focus of our work, the existence of a DSA architecture is at the basis of our proposal. It allows us to specify how policies are created

in the first place and distributed amongst data originators. The solutions we present in the next chapters all start from the specification of a set of policies in a DSA, either to define the set of usage control policies (in the set *UP*) or the set of trusted evaluation authorities (in the set *EA*). In the rest of this thesis we expand upon the presented architecture introducing the new components that constitute the major contributions of our work. These components modify the data publication, policy evaluation and enforcement procedures thus enabling derived data control in partially disconnected networks. The content of the DSA is also described in more detail to include the information needed by the new components to work properly.

Finally, we described how we integrated trusted platform technologies into the presented architecture. The system relies on trusted hardware and uses the integrity measurement architecture to verify whether the environment where policies must be enforced can be trusted. In other words, policy evaluation authorities can verify remotely the set of components installed in recipients' machines and decide to not issue decryption keys on the base of the configurations found. For example, a machine may be considered untrustworthy if it has installed an anti-DRM software such as *QTFairUse*, or if it has an old version of the PEP component known for a security vulnerability. Our policy-driven approach allows us to prove the integrity of a system while decoupling authorisation logic from remote attestation. This grants organisations more flexibility when specifying authorisation policies that can now contain conditions on the hardware and software configuration of the data recipients' machines. For example, this solution allows the specification of policies where access to data on machines with untrusted components is constrained by more or different obligations, for example higher payments, to balance the risk of data leakage.

4 A Data Labelling System for Derived Data Control

4.1 Chapter Overview

One of the possible purposes of inter-organisational cooperations is the creation of new IP through the sharing and processing of existing knowledge. On one hand, new produced information is often more valuable than what was originally shared and its protection is thus a major concern whenever companies initiate business or research partnerships. Protecting only the data that is initially shared is often not sufficient. On the other hand, the newly created information is sometimes less valuable than what was originally shared, or simply there is a need to disseminate it more quickly and to a broader public (e.g. if confidential information such as the location of a weapon storage site were used to create an off-limit warning information on a digital map). However, decisions on what kind of protection should be applied to the newly created data cannot be left solely to the user or organisation who directly created it, as the input information might come from different partners who have the right to impose their own requirements. In this Chapter we present a solution to derive protection requirements for derived data. Our solution considers not only the requirements specified in a DSA by the partner organisations, but also the requirements of the input data used in the derivation process, the decisions users take (whenever allowed) on the protection of the data they create and the characteristics of the derivation process.

The problem of controlling derived data was informally described by Park *et al.* while discussing their $UCON_{ABC}$ family of models [PS04]. The authors defined derived data as an object created as a consequence of exercising rights on an original one (e.g. a log file). Stemming from this definition, we define derived data as *an object created as a consequence of performing a transformation on one or several original ones*. The concept of *transformation*, i.e. any function applicable to one or several data objects that either modifies them or returns a new one, is the fundament of our solution. To be able to control derived data, it is necessary to define a strictly controlled set of transformations that can be used to modify data. It is in fact necessary to know, for each transformation, what kind of output it creates so that it can be associated with the correct policies. Each of these transformations must then be characterised to capture part of the protection requirements of the output

it produces.

We chose not to commit to a particular language from the many available, but rather to define an high-level notation to express XACML-like policies in a simple and readable form. The notation is introduced in Section 4.2 while Sections 4.3 and 4.4 describe how policies can be combined into a *lattice of policy sets*, the policy structure that forms the basis of our solution. Section 4.5 introduces the concept of *content verification procedure* and *transformation* and describes the mechanism for derived data control while Section 4.7 describes how our solution is applied to protect XML documents and integrated with the architecture presented in Chapter 3. Finally, the solution is evaluated with an extensive example drawn from the emergency scenario described in Section 1.3 and compared with existing solutions.

4.2 A Notation and Semantics for Policy Languages

Our solution is based on the XACML-based notation and policy semantics used by Rao *et al.* [RLB⁺] that we modify to include new constraints and obligations. Also, while Rao *et al.* (and XACML) consider actions as part of the policy constraints (thus allowing the specification of a policy valid for different actions), we bind actions directly with the policy ID, so that each policy specifies constraints for one action only. As we will see later, this is done to ease the “partial ordering” of policy sets. Intuitively, *the notion of policy semantics is defined by the set of requests that are permitted and the set of requests that are denied by the policy*. We first give some basic definitions:

- ATT is a finite set of attribute names. Attributes characterise context, data and subjects, have a name $a_i \in ATT$ and a domain of possible values $dom(a_i)$. We assume names are unique, so that no ambiguities can arise;
- $Op = \{>, \geq, =, \neq, \leq, <\}$ is the set of all inequality and equality operators used to compare attribute values; inequality operators can be used only to compare values from ordered or partially ordered domains;
- A is the set of all actions that can be performed. Actions have a name $act_i \in A$.

For the sake of simplicity, in the following we will ignore the distinction at syntactic level between attributes whose values belong to ordered or partially ordered sets and attributes whose values are not ordered, as the meaning will be clear from the context. We define a *request context* as:

Definition 4.1 A context $q \in Q$ is a set $\{(a_1, v_1) \dots (a_k, v_k)\}$ where $a_1, a_2, \dots, a_k \in ATT$ are attribute names, $v_i \in dom(a_i)$ for $1 \leq i \leq k$ are attribute values. Set Q is the set of all possible request contexts.

Intuitively, a request context is the set of subject, object and context attributes that can be verified at the moment a subject requests permission to perform an action on an object. We therefore define an *action request* r as:

Definition 4.2 An action request $r \in R$ is a pair (q, act) where $q \in Q$ is a request context and $act \in A$ is an action. Set R is the set of all possible action requests.

A policy can then be defined as a function $P : R \rightarrow \{Y, N, NA\}$ that given an action request $r \in R$ returns a value in $\{Y, N, NA\}$, indicating whether the action is allowed, denied or a response could not be determined. The policy semantics is defined as:

Definition 4.3 The semantic meaning of a policy $P \in PS$ is a 2-tuple $\langle R_Y^P, R_N^P \rangle$ where R_Y^P and R_N^P are the set of allowed and denied requests, respectively, and $R_Y^P \cap R_N^P = \emptyset$. Set PS is the set of all policies.

The condition on the disjunction of the two sets R_Y^P and R_N^P is an internal consistency condition that is enforced by the policy syntax and construction at the base of our solution. The described semantics is general enough to represent several policy languages such as XACML. Similarly, we must define a general policy syntax. To do so we define an atomic Boolean expression and a compound Boolean expression as:

Definition 4.4 An atomic Boolean expression ae can be:

- A constraint on an attribute value of the form $a_i \triangleright c$ where $a_i \in ATT$, $\triangleright \in Op$ and $c \in dom(a_i)$;
- A constraint on two attribute values of the form $a_i \triangleright a_j$ where $a_i, a_j \in ATT$, $\triangleright \in Op$ and $dom(a_i) = dom(a_j)$;

Definition 4.5 A compound Boolean expression $ce \in CE$ is the association of a set of atomic Boolean expressions via the logical operators $\wedge$ and $\vee$. Set CE is the set of all compound Boolean expressions.

Examples of compound Boolean expressions are: $(time < 8am \vee time > 8pm)$ and $(role = manager \wedge location = South Kensington Campus)$. We specify a policy as:

Definition 4.6 A Policy $P \in PS$ is a tuple $(act, DS_{act}, AS_{act})$ where $act \in A$ is an action and $DS_{act} \subseteq CE$, $AS_{act} \subseteq CE$ are respectively a deny constraint set and an allow constraint set, containing the conditions that must hold to respectively deny or allow an action request.

In other words policies specify a set of constraints over typed attributes whose values range over predefined domains. Policy decision can then be defined as a function $P_{act}(r)$ (Figure 4.1) that given an access request r

$$P_{act}(r) = \begin{cases} \text{if } r.act \neq act & NA \\ \text{else if } r \models \bigvee_{ce \in DS_{act}} ce & N \\ \text{else if } r \models \bigwedge_{ce \in AS_{act}} ce & Y \\ \text{otherwise} & NA \end{cases}$$

Figure 4.1: Function representing the policy evaluation process

substitutes free variables in its Boolean formulae with values from r 's context and returns an access decision depending on whether the disjunction and conjunction of compound Boolean expressions in the deny and allow constraint sets evaluate to true. Also, we consider that the evaluation procedure is based on the *closed world assumption*, so that atomic Boolean expressions are evaluated to false if r 's context does not provide a value for any of their variables. For the sake of simplicity, we did not include a negation symbol in the notation, since negation is implicitly included in the atomic expressions. Note that the ordering of evaluation of the policy expressions implies a *deny overrides* rule for conflict resolution. We decide to support both negative and positive authorisations as exceptions often need to be dealt with (e.g. denials for large groups of users but few [SdV00]). The *not applicable* (NA) return value is introduced as not all policies might be applicable to all requests. In this sense, NA cannot be treated as a default denial or permission as it would disrupt the result given by other applicable policies. Consider for example a policy P_1 granting access to Alice only and a policy P_2 granting access to Bob only, both associated with the same data item. When Alice files an access request, P_1 will return Y and P_2 will return NA . If the result returned by P_2 was considered N by default and the conflict resolution strategy were a *deny override* strategy, Alice would be denied access. The NA result is instead ignored and does not have any effect on the final access decision. Default decisions can instead be applied if no policies are applicable to a specific access request, i.e. if all policies return NA , as we will see later.

This particular representation might seem redundant as compound Boolean expressions could be used rather than constraint sets to represent policy constraints. However, this format eases the lattice construction we will introduce later and still allows users to specify any Boolean expression as access constraints. In other words, this makes the notation with which we specify how policies are organised simpler. Usage control requirements

such as obligations and continuous controls can also be introduced. We can define obligations as:

Definition 4.7 *An obligation $o \in O$ is a procedure call $act(parameters)$ where act is an action to be performed before access to data is granted. Set O is the set of all obligations.*

In this sense, the semantics of obligations is the same considered in the XACML model. Obligations can be easily introduced in the definition of a policy function as:

$$P_{act}(r) = \begin{cases} \text{if } r.act \neq act & \langle NA, \emptyset \rangle \\ \text{else if } r \models \bigvee_{ce \in DS_{act}} ce & \langle N, \{o_1 \dots o_k\}_N \rangle \\ \text{else if } r \models \bigwedge_{ce \in AS_{act}} ce & \langle Y, \{o_1 \dots o_k\}_Y \rangle \\ \text{otherwise} & \langle NA, \emptyset \rangle \end{cases}$$

where policy decisions also contain the corresponding set of obligations that must be performed when requesting access. As for the other constraints, obligation's *free* parameters are also substituted with values from r 's context. As we have already described in Chapter 3, obligations are intended as procedures to be automatically performed by the PEP when receiving the policy evaluation result from the PDP and before granting access to the data. If there are errors during the execution or the obligation cannot be performed, the PEP denies access (even if the policy result was Y) and logs the failure. In this way the system is never in an inconsistent state as only the completion of an obligation is considered as the “committing time” for policy decision and decryption key release. Policies can thus be represented as $(act, \{o_1 \dots o_k\}_Y, AS_{act}, \{o_1 \dots o_k\}_N, DS_{act})$ tuples. Continuous controls (i.e. the monitoring of specific conditions throughout the whole duration of the access to the data that could invalidate the original policy decision and revoke the access) do not need to be included in the notation as attribute monitoring depends on the enforcement mechanism rather than on the policy model and simply consists of a continuous re-evaluation of the same policies. In other words, after access has been granted, the PEP keeps requesting a new policy evaluation to its local PDP at regular intervals (depending on the system's configuration) or whenever the value of an attribute used in the policy changes, so that access can be revoked if the policy evaluation result changes into a denial. More complex conditions could be specified where only certain attributes and conditions need to be monitored. However, this is a matter of optimisation rather than modelling since it simply allows the PEP to request re-evaluations of only a subset of policies. Note that this differs from the $UCON_{ABC}$ model where ongoing obligations can also be specified (i.e. obligations

to be performed at regular intervals during the access). In our implementation, obligations are performed only *once* when access is requested and policy re-evaluation does not force the PEP to perform them again. We stress that the inclusion into our model of such additional controls would be straightforward, but we did not do so as this type of obligations have been already treated in other works [PS04, CLM⁺09] and are beyond the scope of this thesis.

An example policy for the crisis management scenario described in Section 1.3 to protect personal information of victims in the accident area may state that access should be granted to all police or ambulance rescuers in the area or in a hospital and that such access should be logged. Such policy can be specified as:

$$P_{read}(r) = \begin{cases} \text{if } r.act \neq read & \langle NA, \emptyset \rangle \\ \text{else if } r \models zone \neq hospital \vee zone \neq emr.area & \langle N, \emptyset \rangle \\ \text{else if } r \models role \geq Volunteer \vee role \geq Constable & \langle Y, \{(log("read", role, name, dataID))\} \rangle \\ \text{otherwise} & \langle NA, \emptyset \rangle \end{cases}$$

where we assume rescuers are organised in two role hierarchies, one for the ambulance services and one for the police force whose leaves are volunteers and constables, respectively. Note that the policy does not specify any target. While the compound Boolean expressions might simply include conditions to identify target objects (e.g. an equality condition on a document ID or document type) we decided not to do so because policies are associated with objects through sensitivity levels, as we will describe later. Similarly, the action may be included in the compound Boolean expressions rather than be bound directly to the policy ID. This would make for a simpler notation. Our choices come from the need to strictly organise policies and will be clarified later.

4.3 Sensitivity Domains and Data Labelling

Our solution stems from the idea that data can be protected under different domains and for different independent reasons. For example, data may be protected because it contains *private* or *commercially sensitive* information, or it may be related to *public safety* or *national security*. As in multi level security (MLS) systems and Asbestos in particular (see Section 2.3.2), we identify each of such domains (or competence areas) with a *security tag* and associate it with the data. When the data is modified, the amount of sensitive information it contains for each of the applied domains can either increase or decrease, as shown in Figure 4.2.

Data sensitivity can vary on a fixed range of discrete sensitivity levels for each of the defined domains. However, multi level security and mandatory access controls might be too rigid for real-world applications, even

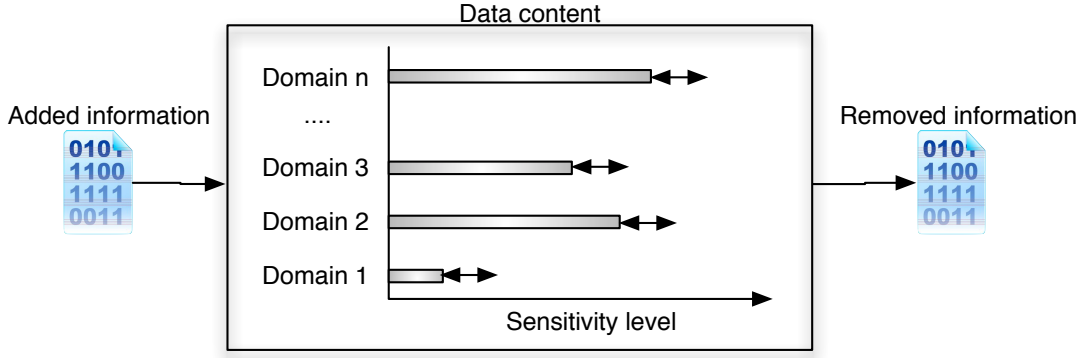
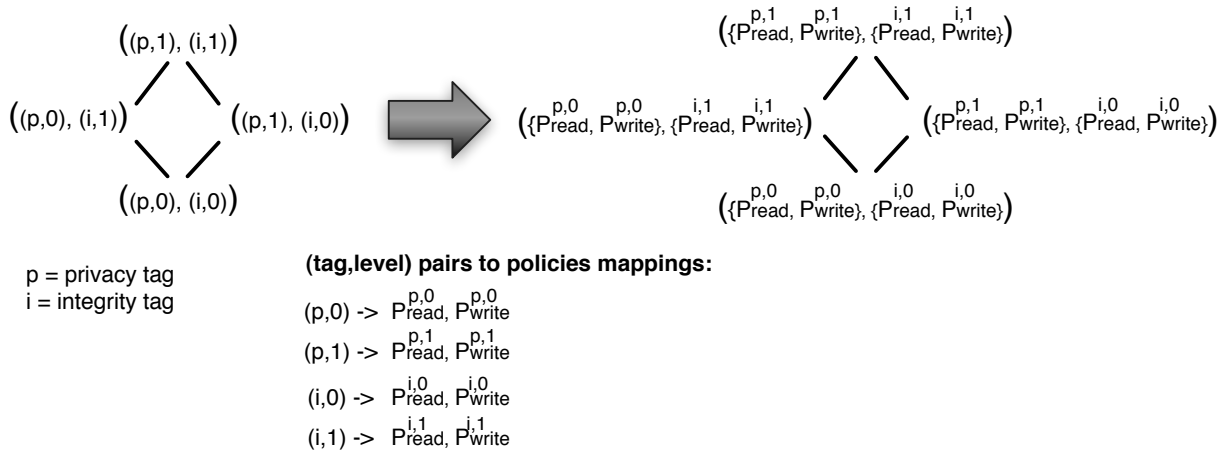


Figure 4.2: Mutable information sensitivity for different domains

when discretionary policies are specified to operate *within the boundaries* of the mandatory policy. Exceptions cannot be dealt with and the *tranquillity principle* of the Bell-LaPadula model must be respected, i.e. objects' classifications cannot be changed during normal operations.

While inspired by MLS systems, our approach does not use clearance labels. User clearance should in fact be determined at run-time depending on both context (e.g. time and location) and user attributes and not with a statically assigned label. Moreover, we consider that the classification of an object can vary if (and only if) its content is modified (in other words, as long as data is not modified, it maintains its original classification). As such, data sensitivity class changes according to the use data is subjected to when received and accessed by recipients. To cater for changing classifications, we devised a solution where each sensitivity class is associated with a set of policies that are applied to objects with that classification. When an object's classification changes, its applicable policies change accordingly. We associate sensitivity classes with policy sets by defining a lattice of policy sets isomorphic to the lattice of sensitivity classes. In other words, sets of policies must be ordered into a lattice according to the protection level they provide. Figure 4.3 depicts an example lattice defined over two sensitivity domains, *privacy* (*p*) and *integrity* (*i*). The example shows four ordered sets of policies, two that controls the *read* and *write* actions for the privacy domain and two that controls the *read* and *write* actions for the integrity domain. How ordered policies can be specified and a lattice created will be explained in the following section.

As in Asbestos, data classification labels representing sensitivity classes are defined as sets of $(tag, level)$ pairs. Each $(tag, level)$ pair is mapped to at most one policy for each action (Figure 4.3) so that the ordering of levels corresponds to the ordering of policies. In other words, each sensitivity level for each tag defines the policies that must control the usage of the so-classified objects and every lattice of data classification labels implies an isomorphic lattice of policy sets (policy lattice in short). Therefore, data is classified under a single



Fully ordered sets of policies:

$P_{read}^{p,0}$	= everyone can read private content with sensitivity level 0
$P_{read}^{p,1}$	= only medics and police constables can read private content with sensitivity level 1
$P_{write}^{p,0}$	= everyone can write private content with sensitivity level 0
$P_{write}^{p,1}$	= only medics and police constables can write private content with sensitivity level 1
$P_{read}^{i,0}$	= everyone can read high-integrity content with sensitivity level 0
$P_{read}^{i,1}$	= everyone can read high-integrity content with sensitivity level 1
$P_{write}^{i,0}$	= everyone can write high-integrity content with sensitivity level 0
$P_{write}^{i,1}$	= only police constables can write high-integrity content with sensitivity level 1

Figure 4.3: Mapping from lattice of data labels to lattice of policy sets

node of the lattice at any one time depending on its data classification label. Consider two types of documents being disseminated amongst rescuers at the scene of an accident. One contains personal information of a victim found in the area and the other contains information on the rescue operations. Figure 4.3 depicts the policy lattice for such documents. Initially, the documents' labels are $\{(p,0) (i,0)\}$ for documents about victims and $\{(p,0) (i,1)\}$ for documents about the operations. In other words, everyone can read documents of both types, but only police constables can write information on the state of the rescue operations. According to the usages data is subjected to, its classification (i.e. its data classification label) can then either increase or decrease in the lattice (how this is realised will be shown later). In other words, the set of policies applied to the data keeps changing depending on the transformations applied to the data. For example, let us consider the case where someone writes information about her own medical history in an initially empty document about victims. The document's classification is immediately increased to $\{(p,1) (i,0)\}$ so that only medics and police constables can read and write it. This approach also allows data originators to protect data integrity as policies are not limited to control read accesses but can also control different operations that can be performed on the data to modify it (e.g. "write").

In the following we describe how well-formed policy lattices, i.e. lattices where an increase in data classification actually corresponds to the application of a set of stricter usage control policies, can be constructed.

4.4 Policy Lattices

4.4.1 Policy Specification

Our policy lattice construction is based on the notion of semantics given in Section 4.2. The protection level provided by a policy can only be measured in terms of the requests that are permitted or denied by the policy. This is measured by the policy's *provided security level (PSL)*:

Definition 4.8 *The provided security level of a policy $P \in PS$ is a relative measure of the access requests P authorises, obtained with the following rule:*

$$\forall P_i, P_j \in PS : PSL_{P_i} \leq PSL_{P_j} \iff R_Y^{P_i} \subseteq R_Y^{P_j} \wedge R_N^{P_j} \subseteq R_N^{P_i}$$

In other words, a policy's provided security level PSL_{P_i} is greater or equal to PSL_{P_j} if the set of requests authorised by P_i is a subset of the set of requests authorised by P_j and the set of requests denied by P_j is a subset of the set of requests denied by P_i . *Not applicable (NA)* results do not affect this relationship.

Given a set of sensitivity tags TS and the lattice of data classification labels that their discrete sensitivity levels generate, we associate each $(tag, level)$ pair with at most one policy for each possible action that can be performed on data (multiple policies for the same action can be easily combined into one with a conjunction or disjunction of their individual constraints). Given a data item with a certain classification label, we can therefore determine the applicable policies for all sensitivity domains. In the example of Figure 4.3, the data label $\{(p,0) (i,1)\}$ indicates that the data is associated with policies $P_{read}^{p,0}$ and $P_{write}^{p,0}$ for the privacy domain and with policies $P_{read}^{i,1}$ and $P_{write}^{i,1}$ for the integrity domain. However, we must guarantee that the policy lattice implied by the lattice of data labels is well-formed, i.e. that higher sensitivity levels correspond to policies with an higher PSL. In other words, the higher an object classification, the fewer access requests are granted. To construct such a lattice, we constrain the way policies are specified by users. Rather than directly writing policies that will later be included in a lattice, we let users specify the individual constraints that must hold at each sensitivity level, for each tag. The set of policies associated with a given $(tag, level)$ pair is then constructed as a combination of the individual constraints of all levels up to that one. This restriction ensures a monotonic growth of the constraints that must be satisfied to obtain access at higher sensitivity levels, thus guaranteeing

that the so-created policy lattice is well-formed (see Section 4.4.3 for a more detailed proof). Constraints are specified by defining each tag t as a chain of sensitivity levels:

$$t : * \rightarrow (\{false\}, \{true\}) \rightarrow (D^{t,1}, A^{t,1}) \rightarrow \dots \rightarrow (D^{t,n}, A^{t,n})$$

where $D^{t,j}$ and $A^{t,j}$ are sets of compound Boolean expressions ce_{act} (at most one for each possible action act) and n is the maximum sensitivity level for tag t . $D^{t,j}$ and $A^{t,j}$ represent the individual constraints that must hold at every sensitivity level and that will be used to build up the deny and allow constraint sets for policies associated with each (tag, level) pair. The sensitivity levels of a given tag correspond to the PSLs policies must provide to protect data sufficiently. The $*$ level indicates that a security domain is not applicable to the data item, i.e. that the data must never be protected under that domain, whereas sensitivity level 0 indicates that the data is not currently protected. In both cases any action is allowed with no obligations. To include obligations in this model, we can simply redefine a chain of sensitivity levels as:

$$t : * \rightarrow (\emptyset, \{false\}, \emptyset, \{true\}) \rightarrow (O_N^{t,1}, D^{t,1}, O_Y^{t,1}, A^{t,1}) \rightarrow \dots \rightarrow (O_N^{t,n}, D^{t,n}, O_Y^{t,n}, A^{t,n})$$

where each level is also associated with the supersets $O_N^{t,i} = \{O_{N,act_1}^{t,i} \dots O_{N,act_j}^{t,i}\}$ and $O_Y^{t,i} = \{O_{Y,act_1}^{t,i} \dots O_{Y,act_j}^{t,i}\}$ of obligatory actions valid at that level. Therefore, each action is associated with two sets of obligations, one for negative and one for positive evaluations.

Given a chain of sensitivity levels for each tag, a set of policies for each action can be determined that are totally ordered according to their PSL, as described in definition 4.8. We define the policy applicable to an action request $r = (q, read) \in R$ for a data item with sensitivity level l for tag t as:

$$P_{act}^{t,l}(r) = \begin{cases} \text{if } r.act \neq act & \langle NA, \emptyset \rangle \\ \text{else if } r \models \bigvee_{\substack{ce_{act} \in D^{t,i} \\ i \leq l}} (ce_{act}) & \langle N, \bigcup_{i \leq l} O_{N,act}^{t,i} \rangle \\ \text{else if } r \models \bigwedge_{\substack{ce_{act} \in A^{t,i} \\ i \leq l}} (ce_{act}) & \langle Y, \bigcup_{i \leq l} O_{Y,act}^{t,i} \rangle \\ \text{otherwise} & \langle NA, \emptyset \rangle \end{cases} \quad (4.1)$$

In other words the deny constraint set for a policy associated with level l and action act , DS_{act}^l , comprises all the compound Boolean expressions $ce_{act} \in D^{t,i}$ for the protection levels lower than l . Similarly, the allow

constraint set for a policy associated with level l and action act , AS_{act}^l , comprises all the compound Boolean expressions $ce_{act} \in A^{t,i}$ for the protection levels lower than l . Note that the equation does not contain any reference to policy subjects or objects. This is because both subjects and objects are identified by the set of constraints that can include conditions on subjects' and objects' attributes.

Consider an example *privacy* domain to protect the personal information of the victims of an accident. Information should be accessible by all police or ambulance rescuers in the emergency area or in a hospital (*privacy* sensitivity is equal to 1). However, access to data containing medical information (*privacy* sensitivity is equal to 2) can be granted only to someone who possesses a first-aid expertise certification or is a paramedic or medic. Also consider the *read* action as the only possible one. We represent this case by defining two sensitivity levels for privacy:

$$p : * \rightarrow (\emptyset, \{false\}, \emptyset, \{true\}) \rightarrow (\emptyset, D^{p,1}, \emptyset, A^{p,1}) \rightarrow (\emptyset, D^{p,2}, \emptyset, A^{p,2})$$

associated with the following compound Boolean expressions:

$$\begin{aligned} D^{p,1} &= (ce_{read} = \{zone \neq hospital \vee zone \neq emr. area\}) \\ A^{p,1} &= (ce_{read} = \{(role \geq Volunteer \vee role \geq Constable)\}) \\ D^{p,2} &= \emptyset \\ A^{p,2} &= (ce_{read} = \{aid exp. = true \vee role \geq Paramedic\}) \end{aligned}$$

where the inequality operators applied on roles are based on an existing role hierarchy (or partial order) where medics are higher-level rescuers than paramedics and volunteers. Hence, the corresponding encoding for the two policies applicable at privacy level 1 and privacy level 2 is:

$$\begin{aligned} P_{read}^{p,1}(r) &= \begin{cases} \text{if } r.act \neq read & \langle NA, \emptyset \rangle \\ \text{else if } r \models zone \neq hospital \vee zone \neq emr. area & \langle N, \emptyset \rangle \\ \text{else if } r \models role \geq Volunteer \vee role \geq Constable & \langle Y, \emptyset \rangle \\ \text{otherwise} & \langle NA, \emptyset \rangle \end{cases} \\ P_{read}^{p,2}(r) &= \begin{cases} \text{if } r.act \neq read & \langle NA, \emptyset \rangle \\ \text{else if } r \models (zone \neq hospital \vee zone \neq emr. area) & \langle N, \emptyset \rangle \\ \text{else if } r \models (role \geq Volunteer \vee role \geq Constable) & \langle Y, \emptyset \rangle \\ \wedge (aid exp. = true \vee role \geq Paramedic) & \langle Y, \emptyset \rangle \\ \text{otherwise} & \langle NA, \emptyset \rangle \end{cases} \end{aligned}$$

The proof that policies corresponding to different levels of the same tag and to the same action are totally ordered on the basis of their provided security level is given in Section 4.4.3.

4.4.2 Policy Ratification

Given the above allow and deny constraint set combination rule, it is possible that redundant policies are created. Consider a compound Boolean expression at level i requiring that a user has at least a *paramedic* role and an expression at level j with $i < j$ requiring that a user has at least role *medic*, where *medic* dominates *paramedic* in the considered role hierarchy. In the policy obtained for level j the paramedic condition is redundant and could be eliminated.

In the general case, using a chain of compound Boolean expressions can lead to redundant policies. To avoid a waste of computational resources, it might be useful to reduce a policy to its minimal form, i.e. a form where no conditions can be removed without changing the output of the policy evaluation for any given request. However, finding the minimal form of a constrained Boolean expression, for example using binary decision diagrams (BDDs), is a well-known NP-complete problem [BW96]. It is therefore only possible to apply heuristics for fast reductions. Given a specific Boolean expression representing a policy, we could for example transform it in Conjunctive Normal Form or Disjunctive Normal Form and perform a dominance check between atomic conditions contained in the same conjunct or disjunct, and then a coverage check between distinct conjuncts or disjuncts [AGLL05]. However, this solution has two main drawbacks. First, given our definition of constraint, dominance checks cannot cause the deletion of a constraint that is the only one specified over a specific attribute. This means that the number of credentials that must be gathered for a policy evaluation would remain the same after a reduction operation. Second, a reduction step might cost more than evaluating redundant constraints. This is why we do not take any action to reduce policy redundancy. Similarly, a redundant or even conflicting set of obligations (e.g. closing and opening a connection) could be defined. However, conflicting obligations are application-specific in the sense that conflicts depend on the semantic of the involved obligations. This kind of conflicts can be detected and thus solved with existing techniques [CLM⁺09].

Besides redundant policies, the allow and deny constraint set combination rule we described also allows the specification of inconsistent constraints at different sensitivity levels and thus the creation of policies that deny all requests. However, we do not consider such policies as errors, since it should always be possible to specify situations where a data object must be sealed forever. If this is not the case and data is put out of reach by mistake, recovery mechanisms must be put in place to take the data classification back to a consistent state. As already stated, recovery mechanisms are left for future work.

4.4.3 Lattice Derivation

Once a set of tags has been specified as described above, their totally ordered sensitivity levels automatically generate a lattice where sensitivity classes are actually policy sets. In the following we demonstrate that:

- policy sets are partially ordered according to the PSLs of their individual policies, i.e. the policy lattice is isomorphic to a lattice of data classification labels;
- the set of sensitivity classes, together with the ordering relationship amongst policies (that is equivalent to the flow relation of traditional information flow systems), constitute a well-formed lattice.

Our information flow model is defined by:

$$PM = \langle SC, \rightarrow, \oplus \rangle$$

where SC is the set of all sensitivity classes (i.e. of sets of $P_{act}^{t,i}$ policies) and $\rightarrow$ is a flow relation defined on pairs of sensitivity classes that indicates the permitted information flows. In particular, $A \rightarrow B$ indicates that any object protected by the policies contained in A can also be protected by the policies contained in B (i.e. the policies in B are at least as restrictive as the policies contained in A). The third component, $\oplus$, is a class-combining operator on SC defining how to label information obtained by combining information from other sensitivity classes and will be described later. Given our construction, we define $\rightarrow$ as:

$$A \rightarrow B \Leftrightarrow \forall P_{act}^{t,j} \in A \exists P_{act}^{t,i} \in B \mid PSL_{P_{act}^{t,j}} \leq PSL_{P_{act}^{t,i}}$$

In other words, B dominates A if and only if for each policy $P_{act}^{t,j}$ in A there exists at least one policy $P_{act}^{t,i}$ in B defined for the same tag and action and that provides a higher security level. We must now demonstrate that this rule holds for all flow relations created when specifying a set of tags and their chains of sensitivity levels. For simplicity in the notation we also assume only one action is available.

Demonstration 4.1 *Let us consider two policies $P_{act}^{t,i}$ and $P_{act}^{t,j}$ such that $i \geq j$. We must prove that $PSL_{P_{act}^{t,i}} \geq PSL_{P_{act}^{t,j}}$. This follows immediately from the construction of the policies. Consider the policy corresponding to the first sensitivity level for tag t :*

$$P_{act}^{t,0}(r) = \begin{cases} \text{if } r.act \neq act & \langle NA, \emptyset \rangle \\ \text{else if false} & \langle N, \emptyset \rangle \\ \text{else if true} & \langle Y, \emptyset \rangle \end{cases}$$

The policy allows every request (and returns an empty set of obligations) and denies none, i.e. $R_Y^{P_{act}^{t,0}} = R$ and $R_N^{P_{act}^{t,0}} = \emptyset$. We call this policy the zero policy. Each policy at successive levels $P_{t,i} \forall i > 0$ adds a disjunction of Boolean conditions to the deny rule and a conjunction of Boolean conditions to the allow rule. This increases the number of denied requests and restricts the number of allowed ones, i.e. $R_Y^{P_{act}^{t,i}} \subseteq R_Y^{P_{act}^{t,j}}$ and $R_N^{P_{act}^{t,i}} \subseteq R_N^{P_{act}^{t,j}} \forall i > j$. From the definition of PSL we can therefore derive that $PSL_{P_{act}^{t,j}} \leq PSL_{P_{act}^{t,i}}$.

Note that the solution we are proposing is not the only possible one and different approaches can be taken to order policies according to the definition of PSL we have given.

To show that $\langle SC, \rightarrow \rangle$ forms a lattice in our model we must establish that:

- SC is finite;
- $\langle SC, \rightarrow \rangle$ is a partially ordered set;
- SC has a lower bound $\perp$ such that $\perp \rightarrow A \forall A \in SC$;
- There exists a least upper bound operator on SC, $\sqcup$.

SC is finite as an obvious consequence of the limited set of sensitivity tags and levels (and thus policies) that it is possible to specify. To demonstrate that $\langle SC, \rightarrow \rangle$ is a partially ordered set we must demonstrate that the flow relation $\rightarrow$ is reflexive, transitive and antisymmetric, that is:

- $A \rightarrow A \forall A \in SC$ (reflexivity);
- $A \rightarrow B \wedge B \rightarrow C \Rightarrow A \rightarrow C \forall A, B, C \in SC$ (transitivity);
- $A \rightarrow B \wedge B \rightarrow A \Rightarrow A = B$ (antisymmetry).

Demonstration 4.2 The three properties can be easily proven by proving that they hold when comparing individual policies rather than sensitivity classes. Reflexivity is immediately proven as given a policy $P_{act}^{t,i}$, $PSL_{P_{act}^{t,i}} = PSL_{P_{act}^{t,i}}$. Reflexivity then holds for sensitivity classes if it holds for all of their policies. Transitivity can be proven similarly. Given three policies $P_{act}^{t,i}$, $P_{act}^{t,j}$ and $P_{act}^{t,k}$ such that $PSL_{P_{act}^{t,i}} \leq PSL_{P_{act}^{t,j}}$ and $PSL_{P_{act}^{t,j}} \leq PSL_{P_{act}^{t,k}}$, the total ordering given by PSLs requires that $PSL_{P_{act}^{t,i}} \leq PSL_{P_{act}^{t,k}}$ for consistency. Finally, given two policies $P_{act}^{t,i}$ and $P_{act}^{t,j}$ such that $PSL_{P_{act}^{t,i}} \leq PSL_{P_{act}^{t,j}}$ and $PSL_{P_{act}^{t,j}} \leq PSL_{P_{act}^{t,i}}$, consistency requires that $PSL_{P_{act}^{t,i}} = PSL_{P_{act}^{t,j}}$, proving antisymmetry.

The sensitivity class containing zero policies for all tags acts as the lower bound $\perp$ for SC. In fact, for each policy $P_{act}^{t,i}$, $PSL_{P_{act}^{t,i}} \geq PSL_{P_{act}^{t,0}}$. The least-upper bound operator $\sqcup$ can also be easily obtained as the operator

that given two sensitivity classes returns a sensitivity class containing for each tag the highest level in the input classes. In other words the operator returns a sensitivity class associated with the policies with the highest PSL in the input classes, for each tag and action:

$$L_a \sqcup L_b(t) = \begin{cases} L_a(t) & \text{if } L_a(t) \geq L_b(t) \\ L_b(t) & \text{otherwise} \end{cases} \quad (4.2)$$

where L_a indicates a label, and thus a tuple of (tag, level) pairs in the lattice isomorphic to the policy lattice and $L_a(t)$ indicates the numeric value of the label's sensitivity level for tag t . We can then show that:

- $A \rightarrow A \sqcup B$ and $B \rightarrow A \sqcup B$;
- $A \rightarrow C \wedge B \rightarrow C \Rightarrow A \sqcup B \rightarrow C$.

Demonstration 4.3 Both properties are derived from the very definition of $\sqcup$ and $\rightarrow$. For every policy $P_{act}^{t,i}$ associated with A , $A \sqcup B$ is associated with either the same policy, or a policy $P_{act}^{t,j}$ from B such that $PSL_{P_{act}^{t,i}} \leq PSL_{P_{act}^{t,j}}$, thus proving the first property. If for every policy $P_{act}^{t,i}$ associated with A and $P_{act}^{t,j}$ associated with B there exists a policy $P_{act}^{t,k}$ associated with C such that $PSL_{P_{act}^{t,i}} \leq PSL_{P_{act}^{t,k}}$ and $PSL_{P_{act}^{t,j}} \leq PSL_{P_{act}^{t,k}}$, then $A \sqcup B$ cannot be associated with a policy with PSL higher than the corresponding policy in C , thus proving the second property.

Therefore, defining a set of tags and their chains of sensitivity levels leads to the construction of a lattice of policy sets. The existence of a *greatest-lower bound operator* $\sqcap$ and of a highest bound $\top$ (the sensitivity class containing the policies with the highest possible PSL for all tags) is implied by the above properties. We can thus define $\sqcap$ as:

$$L_a \sqcap L_b(t) = \begin{cases} L_a(t) & \text{if } L_a(t) \leq L_b(t) \\ L_b(t) & \text{otherwise} \end{cases} \quad (4.3)$$

4.4.4 Policy Decision

Policies defined for each tag are fully ordered according to their PSLs. Introducing new tags allows us to specify more policies that are relevant for different sensitivity domains and to generate a *policy lattice* whose minimal element is $\perp$ and maximal element is $\top$. Therefore, data can be associated at any moment with a sensitivity class L_d and thus with a set of currently applicable policies $P_{act}^{L_d} = \{P_{act}^{t_1, l_1}, P_{act}^{t_2, l_2}, \dots, P_{act}^{t_n, l_n}\}$ for each action, such that an access request $r = (q, act)$ is evaluated as follows:

$$P_{act}^{L_d}(r) = \begin{cases} \langle N, O_N \rangle \iff \exists P_{act}^{t,l_t} \mid P_{act}^{t,l_t}(r) = N \\ \langle Y, O_Y \rangle \iff \exists P_{act}^{t,l_t} \mid P_{act}^{t,l_t}(r) = Y \wedge \\ \quad \neg \exists P_{act}^{t,l_t} \mid P_{act}^{t,l_t}(r) = N \\ \langle NA, \emptyset \rangle \iff \forall P_{act}^{t,l_t} \mid P_{act}^{t,l_t}(r) = NA \end{cases} \quad (4.4)$$

where O_N and O_Y are the sets of all obligations returned by all policies returning a negative and positive evaluation, respectively. In other words, an access request is not authorised if at least one policy specified for the requested action and associated with the current sensitivity class returns N . Otherwise, if at least one policy returns Y the access request is authorised. This implies that a *deny overrides* rule is applied whenever different policies return conflicting results. *Not applicable* results (obtained if all policies from all tags return NA) can be treated differently according to the specific application scenario. A default permission or denial could be returned or decision might be deferred to specific non-conflicting *decision policies* [JSSS01]. In the remainder of this thesis we will use a default deny policy so that *not applicable* results correspond to denials.

4.5 Data Labelling

4.5.1 Content Verification Procedures

When it is first created, data must be immediately associated with the most suitable policies in the policy lattice. Associating new data with level 0 is not a viable solution, as it assumes data is initially empty or not sensitive. To determine the initial sensitivity of data we introduce the concept of *content verification procedure* (CVP), similar in spirit to the *integrity verification procedures* (IVP) used in [CW87]. An IVP is defined as a procedure meant to verify the validity of the state of constrained data items after transformation procedures are executed on them. In our case, CVPs are used on newly created data to determine their sensitivity based on their content. A similar concept was also used to classify database elements in multilevel relational databases [LDS⁺90, JS91] where elements in relational tables were classified on the basis of their actual value rather than according to the relational column they were members of. Content-dependent conditions for discretionary policies [SdV00] are also similar in concept to CVPs.

A content verification procedure is a Boolean function that verifies specific conditions or properties on the data content. A good example of CVP is a well-known function that is widely used nowadays, for example by Google Maps: the face recognition function. Consider a video being taken by a CCTV camera installed in a building. The video must be kept private as it shows people entering and leaving the area. However, this is not true if no one appears in the video. Once the recording stops or even while video is streaming from the

camera into a nearby workstation, the new data must be correctly labelled to protect people's privacy. Rather than relying on a trusted user to label the data as it is created, we propose to verify automatically whether the data contains sensitive information (in this case human faces) and to label it accordingly. Other examples are functions that analyse the structure of XML data or the values of spreadsheets [GZS09].

Whenever new data is created, the initial level for each tag in its classification label is decided by the results of the CVPs specified in a DSA by the cooperating organisations. We consider the granularity of DSAs to be fine enough so that the number of CVPs to execute is not overwhelming. In other words, given several organisations sharing data at different levels, we assume several different DSAs to be put in place for each level (such as “working groups” constituted by cooperating sub-departments), so that not all possible CVPs for the whole partnership must be executed but only those related to the specific level/group within which the data was created. When defining tags and their chains of sensitivity levels, a CVP is associated with each level:

$$t : * \rightarrow (\emptyset, \{false\}, \emptyset, \{true\}, CVP^{t,0}) \rightarrow (O_N^{t,1}, D^{t,1}, O_Y^{t,1}, A^{t,1}, CVP^{t,1}) \rightarrow \dots \rightarrow (O_N^{t,n}, D^{t,n}, O_Y^{t,n}, A^{t,n}, CVP^{t,n})$$

where the value returned by $CVP^{t,i}$ indicates whether the data should be labelled with level i for tag t . Note that the $*$ value is applied whenever no CVP returns true, otherwise the highest level whose CVP returns true is applied. CVPs are particularly useful whenever the initial sensitivity level for a given tag should not be decided manually by the data originator but on the basis of higher-level corporate policies. Otherwise, a simple CVP can be used that asks the data originator to input a specific initial sensitivity level for the data (in the following we will indicate this CVP as *Req*).

We can now re-define the example privacy tag introduced in Section 4.4.1 to include content verification procedures:

$$\begin{aligned} p : * \rightarrow (\emptyset, \{false\}, \emptyset, \{true\}, isText()) &\rightarrow (\emptyset, D^{p,1}, \emptyset, A^{p,1}, NamesOrAddresses()) \rightarrow \\ &\rightarrow (\emptyset, D^{p,2}, \emptyset, A^{p,2}, MedInfo()) \end{aligned}$$

where *NamesOrAddresses()* and *MedInfo()* are two CVPs verifying whether the data contains personal names or addresses and medical information, respectively. Note that we automatically protect textual data with privacy level 0, implying that all textual information can potentially be private.

4.5.2 Transformation Labelling

After data has been associated with an initial classification label and disseminated amongst unknown recipients, we must ensure that its content always conforms to one of the sensitivity classes contained in the lattice,

independently of the usages it is subjected to. Relying on content verification procedures to relabel data every time it is modified is not a viable solution, as some data properties cannot be verified by any CVP. Clark and Wilson [CW87] already showed how protecting data integrity requires a strict control over the set of valid transformations that can be performed. They defined *transformation procedures* as the only procedures that are allowed to modify or create new constrained data items. Control over the set of transformations allowed the authors to enforce the *well-formed transaction requirement* so that only legitimate (i.e. integrity preserving) actions could be executed by users. Similarly, we aim to guarantee that after each transformation data can still be classified under the policy lattice provided initially and that such classification can be found easily. To do so we pre-define a set of transformation functions through which users can modify or create new data, and define a *class-combining operator* $\oplus$ that specifies the class to which the output of a specified transformation belongs. Our definition of $\oplus$ differs from the one given by Denning [DD77] in that the result returned by the operator *depends* on the particular transformation used to transform the inputs.

To determine the result of the $\oplus$ operator we characterise transformation functions by associating them with different *transformation labels*. First, each transformation is associated with a **classification label** L^c containing a sensitivity level for each existing tag. Each (tag,level) pair in L^c represents the minimum sensitivity level of the output data for that tag. The default level for tags in a classification label, if not specified explicitly, is considered to be 0, meaning that the transformation's output is at most as sensitive as the most sensitive input data. The range of sensitivity levels for classification labels does not include the $*$ value. Intuitively, derived data returned by a transformation may not be sensitive and thus be associated with sensitivity level 0, but a security requirement specified at creation time for the input data cannot be completely removed from the derived one. Therefore derived data cannot be associated with the $*$ level, unless the original data is as well.

Besides the greatest lower bound and least upper bound operators introduced in Section 4.4.3 to combine data classification labels, we must also consider that when a specific sensitivity domain is not applicable for certain data (i.e. the level for the specific tag is equal to $*$), whatever transformation is applied to the data the $*$ value of the tag cannot be modified. We therefore introduce a $*$ -preserving operator whose second operand is always a classification label:

$$L_a \sqcup^* L^c(t) = \begin{cases} * & \text{if } L_a(t) = * \\ L_a \sqcup L^c(t) & \text{otherwise} \end{cases} \quad (4.5)$$

Whenever one or more data inputs with data labels $L_{d_1} \dots L_{d_n}$ are transformed through a transformation with classification label L^c , the output is assigned label:

$$\oplus(L_{d_1} \dots L_{d_n}, L^c) = (L_{d_1} \sqcup L_{d_2} \sqcup \dots L_{d_n}) \sqcup^* L^c \quad (4.6)$$

For simplicity, in the following we will use the notation:

$$\oplus(L_{d_1} \dots L_{d_n}, L^c) = \bigsqcup L_{d_i} \sqcup^* L^c \quad (4.7)$$

After the execution of a transformation function, the sensitivity level for each applicable tag is increased to be the least upper bound of the input data labels and classification label. The $*$ value is overridden if the classification labels of different input data objects have values higher than $*$ for the same tag. This happens because while the $\sqcup$ operator combines two data classification labels, and thus the highest sensitivity level must be considered (also overriding $*$ levels), the $\sqcup^*$ operator combines a data classification label with a transformation's classification label, in which case $*$ levels in the former label override higher levels in the latter label.

As it is now, the $\oplus$ operator only considers transformations that return output data more sensitive than the input data. However, transformations can also declassify data. To address this case, we associate transformations with two **declassification labels**, namely the **general declassification label** L^g and the **relative declassification label** L^- . Intuitively, while the classification label represents the value added by the transformation to the input data, the general declassification label represents the amount of sensitive information that is lost in the transformation process. For example, a summary transformation may remove all details deemed not useful from a document. If these details included personal information about the document's author, then the output summary's sensitivity for an hypothetical privacy domain should be reduced. Each (tag, value) pair in L^g represents the maximum sensitivity level of the output data for that tag. If not specified explicitly, the default value for tags in general declassification labels is the highest possible level for that tag, meaning that no information is lost. With the general declassification label L^g , the $\oplus$ operator becomes:

$$\oplus(L_{d_1} \dots L_{d_n}, L^c, L^g) = \bigsqcup (L_{d_i} \sqcap L^g) \sqcup^* L^c \quad (4.8)$$

The rule first considers the loss of information in the input data due to the declassification, and then increases the value of the declassified inputs according to the applied transformation. When for some tags' levels the input data does not contain enough sensitive information to be further declassified (i.e. $L_d(t) \leq L^g(t)$), the general declassification has no effect on such tags.

The relative declassification label represents the loss of information relative to the current sensitivity of the data. Each (tag, value) pair in L^- (where values are real numbers in the range $[0 \dots 1]$) represents the reduced sensitivity for that tag as a proportion of the current sensitivity. If not specified explicitly, the default value for

tags in L^- is 1, meaning that no information is lost. L^- is also associated with a threshold value th indicating the minimum non-integer sensitivity value under which the data is classified at level 0. Consider for example an image data and a transformation that reduces its quality or resolution. The loss of information depends on the current resolution of the data, thus the security level required by the output data cannot be universally defined in a label. To apply relative declassification labels we introduce a further operator:

$$L_a - L^-(t) = \begin{cases} * & \text{if } L_a(t) = * \\ 0 & \text{if } L_a(t) \times L^-(t) < threshold \\ \lceil L_a(t) \times L^-(t) \rceil & \text{otherwise} \end{cases} \quad (4.9)$$

The relative declassification label allows any discrete level to be decreased by a specific percentage and then rounded up to the next discrete level. With the relative declassification label L^- , the $\oplus$ operator becomes:

$$\oplus(L_{d_1} \dots L_{d_n}, L^c, L^g, L^-) = \bigsqcup((L_{d_i} - L^-) \sqcap L^g) \sqcup^* L^c \quad (4.10)$$

The $\oplus$ operator described so far can only be applied to transformations whose output, or at least its characteristics, are well-known. However, in many cases it is not possible to know in advance what the output of a transformation will look like. Examples are transformations performed by human users, such as text editing. A consequence of this impossibility is that a transformation may change one or more tags such that a wrong sensitivity level is applied to the output data. To address this problem we introduce **decisional labels** L^d . The decisional label associates each tag with a Boolean value indicating whether a content verification procedure is required (as for newly created data) to determine the security level of the output data. The default value for a tag in a decisional label is *false* (i.e. CVPs are not used by default). With the decisional label L^d , the $\oplus$ operator finally becomes:

$$\oplus(L_{d_1} \dots L_{d_n}, L^c, L^g, L^-, L^d) \begin{cases} \max(i) \mid CVP_{i,t} = true & \text{if } L^d(t) = true \\ \bigsqcup((L_{d_i} - L^-) \sqcap L^g) \sqcup^* L^c & \text{otherwise} \end{cases} \quad (4.11)$$

In other words, if the decisional label is set to true for a given tag, the highest sensitivity level for that tag whose CVP is verified is applied to the data. If the decisional label is set to *true* for a tag but no CVP is satisfied after a transformation, then the data modifications are considered not valid and must be rolled back. Similarly, the absence of a required CVP is considered an error and also leads to a rollback. Note that if the specified CVP for a given level is a *Req* procedure that requires a user to enter a sensitivity level, this may correspond to a request to the data originators of the input objects (and not to the user who is modifying the data). This means that 1) the data originator's identifier must always be attached to the data and 2) that a consensus amongst the

originators of several input objects must be found to authorise a transformation. Finding a consensus amongst originators that might be unreachable is however an expensive and time consuming operation and not one we will try to accomplish in this thesis, even if only one originator needs to be contacted. We instead decide to assume that users authorised to transform the data are also authorised to decide the classification of the derived information (if a *Req* CVP has been specified).

4.6 Secure transitions

Bell and LaPadula designed their model around the *basic security theorem*, which states that an MLS system is secure if i) its initial state v_{init} is BLP-secure (i.e. it satisfies the simple security and *- properties) and ii) the state transition function T preserves security, i.e. it moves the system from a BLP-secure state to another BLP-secure state. As pointed out by McLean with his example called "System Z" [McL90], the failure to control transitions can cause several security problems in a BLP-secure system. The example describes a system with only one transition that downgrades all subjects and objects in the system to the lowest possible classification. The Bell and LaPadula model deals with these examples by assuming with its *tranquillity principle* that object classifications should not be changed during normal operations. However, with such an assumption the model becomes too restrictive for real-world systems. McLean proposes a new definition of security based on the notion of *secure transition*, i.e. a transition where object and subject classification are changed only by subjects trusted to do so. For McLean, a system is secure if all its reachable states are BLP-secure (i.e. simple secure and *-secure) and its transition function is transition secure.

The first difference between our model and a traditional MLS model is that subjects are not associated with a clearance label. Let S be the set of all subjects. We can classify subjects on the basis of the strictest policies they satisfy for a given action act , i.e. the maximum PSL for each domain t such that they satisfy $P_{act}^{t,PSL}(r)$. Let us call such PSL *satisfied security level* for action act (SSL_{act}). Note that this classification varies over time as policy evaluations change with the request contexts users provide. We can therefore define a subject clearance label for action act as a set of (tag, SSL_{act}) pairs and define a revised *simple security property*:

Definition 4.9 *A state is simple secure if and only if for every subject that has read access to an object in the state, the clearance label for action read of the subject dominates (in the policy lattice) the data classification label of the object.*

The second difference between our model and a traditional MAC model is that write access is controlled with the same mechanism as read access, where we consider all calls to transformation functions as write requests. Again, we define a revised *-property:

Definition 4.10 A state is **-secure* if and only if for every subject that is permitted to perform a transformation f on an object in the state, the clearance label for action f of the subject dominates (in the policy lattice) the data label of the object.

This is reasonable since both the read action and transformation functions are actions that can be associated with policies, so that stricter policies allow less users to perform them. Our transition function is a function $F : Q \times A \times V \rightarrow V$ where $q \in Q$ is a request context containing the subject, object and environmental attributes, A is the set of actions that can be performed on data and V is the set of states of the system. In particular, let TR be the set of all transformations and D be the set of all data objects in state v . Then, $A = TR \cup \{read, createNew\}$. A transition function is *transition secure* if and only if each transition $F(q, act, v) = v'$ is such that for all objects $d \in D$ and all tags $t \in TS$, if $L_d(t) \neq L_d(t)'$ then:

- $act \in TR$;
- act took d as input;
- at least one of the transformation labels associated with act has a non-default value for tag t or act took as input at least another object d_i besides d with a different classification label;
- the issued request $r = (q, act)$ satisfies $P_{act}^{L_d}$ (i.e. $P_{act}^{L_d}(q, act) = Y$). In other words a transition can modify the classification of an object only if the subject executing it was authorised to do so. This is an obvious consequence of the fact that access to transformations is controlled by policies associated with the same lattice used to classify data and transformations.

Since data classification can only be changed by the operators defined in Section 4.5.2 and since the set of transformations is pre-defined, secure transitions will always move the system into a state where data labels are defined over the existing lattice, i.e. $\forall d \in D : \perp \sqsubseteq L_d \sqsubseteq \top$ so that further transitions would also be secure. This is why it is important that policy sets are ordered in a lattice. If there existed a pair of sensitivity classes without a unique least upper bound or unique greatest lower bound, we could not determine deterministically the sensitivity class of the output of certain transformations.

We underline that traditional MLS solutions can be implemented using the proposed model by having request contexts that only contain the requesting user's clearance. The *simple security property* can then be verified by policies that only check the dominance relationship between user clearance and data classification, while the **-property* can be verified by policies that check the opposite relationship. Also, traditional access control and policy systems can be implemented by reducing the policy lattice to just one sensitivity level for each domain.

4.7 Derived Data Protection Architecture

In the following sections we show how the label model described in the previous section can be applied to protect XML documents and how it can be integrated in the high-level architecture we described in Chapter 3. In particular, our crisis management scenario is based on the assumption that rescuers disseminate data in the form of *tactical situation objects (TSO)*, i.e. highly structured and pre-defined documents whose format is understood by rescuer organisations. The implementation of the **derived data manager** component (Figure 3.2) uses the XQuery and XPath languages as means to specify data transformations and CVPs. XQuery is a query and functional programming language that can be used to query XML documents. The queries' outputs can be saved as new XML documents or overwrite already existing ones. In our implementation we assume the file name of documents correspond to the name of their root XML element. This is how we determine when a transformation overwrites (i.e. modifies) an already existing document or creates a new one. Queries can be complex enough to eliminate, rearrange and merge XML elements, attributes and data from several documents at the same time and can also call Java functions to perform more complex operations. For example, an XQuery transformation may take as input a documents containing personal information about an organisation's employees, a document containing information about projects run by another organisation and merge them by matching employees with projects they could participate to (e.g. on the base of their competencies described by XML attributes). XQuery allows us to provide a flexible and general implementation of our solution.

XPath is a query language for selecting specific elements from XML documents. XPath queries can be used to explore the XML tree of a document and select nodes on the basis of their path, attributes and content. We define CVPs as XPath expressions taking as input XML documents and returning a list of elements that satisfy the procedure. For example, an XPath CVP may take in input a document and return all the subelements containing personal information (names, dates of birth etc.). This slightly differs from the model described in Section 4.5.1 where CVPs only return Boolean values. This is due to the fact that we are now dealing with structured data rather than atomic data objects. We can thus think of XPath CVPs as returning *true* for all XML elements they select. The type and complexity of CVPs depend on the application scenario. However, we assume that the use of XML documents and XPath facilitates the CVP specification process at DSA definition time. If the disseminated data contains instead free text or images, the complexity of CVPs may increase and include complex algorithms to analyse the data content. However, both CVPs and transformations could be easily reused whenever documents have similar or standardised structures or when they contain similarly formatted data (such as images, videos etc). As previously mentioned we assume that the set of transformations that can be applied to data is pre-defined. Users cannot specify their own transformations (XQueries) and are

limited to a specific set of predefined ones. For example, given two documents a DSA may state that users can only transform documents by summarising them (with an XQuery removing specific XML elements) or merging them (with an XQuery merging information from one document with information from an other one).

4.7.1 Data Sharing Agreement

Using our labelling model, protecting derived data means deciding the classification labels associated with it. Users and applications manipulate data through a series of *transformation functions* that need to be known in advance by all partner organisations for two reasons. First, organisations must be able to specify policies controlling how such transformations are used on the data they disseminate, even after the data crosses the organisational boundaries. Second, to specify how derived data must be protected, data originators must know how transformations actually process the original data. To this end, we use DSAs as the main means for partners to agree on a common set of transformations.

As defined in Chapter 3, a DSA is a 7-tuple $(ID, S, A, EA, CA, UP, V)$ where UP represents the set of usage control policies. We can now expand this definition:

Definition 4.11 *The set of usage control policies UP specified in a data sharing agreement $(ID, S, A, EA, CA, UP, V)$ is a 3-tuple (TAG, CVP, TR) where TAG is the set of all tags (and their chains of sensitivity levels) associated with the agreed sensitivity domains, CVP is a set of XPath functions representing the content verification procedures used in the tags' definition and TR is a set of XQuery functions representing the available data transformations.*

Therefore, partner organisations agree at DSA level on the structure of the policy lattice controlling data usage and on the set of transformations recipients will be authorised to use. For example, a DSA may contain a CVP looking for elements containing personal information and associating them with a given sensitivity level for the privacy domain and a transformation removing all such elements from the document, thus declassifying the element's privacy sensitivity level. For the rest of this chapter we will assume the term EA of the DSA contains a static list of references to trusted evaluation authorities.

In our scenario the use of a DSA is ideal. In general, large libraries of transformations and CVPs may be constructed on the basis of well-known regulations (e.g. the european regulations on data protection [eud12]) and be freely available for everyone to use.

4.7.2 Data Publication

The data publication phase we propose is shown in Figure 4.4. Note that the process stems from the one described in Section 3.4.2 with the addition of the derived data controller component. When new data is created and a publication request is issued, the PEP first queries the communication manager for the state of the network. If connectivity is available, the client first tries to update the policies (i.e. tags, CVPs and transformations) it currently stores. Then, the PEP asks the requesting application to provide a set of metadata to be published with the data. This phase is application-specific and the metadata used may widely vary. In general, the PEP should require the application to provide at least the metadata needed for the evaluation of the available policies. The PEP then asks the derived data controller to run all the available CVPs and correspondingly label the data the user wants to publish. Once the data has been labelled, the PEP encrypts it and encloses all the information needed for decryption into a *publishing license* that is also encrypted for a chosen policy evaluation authority from the set *EA*. Finally, the licence is attached to the encrypted data for dissemination. We will describe how data labelling and encryption are performed and the format of the publishing licence in the following.

Consider the XML structure depicted in Figure 4.5 containing medical records of victims of an accident and two example sensitivity tags, *privacy* and *confidentiality*. Every element can be associated with a data classification label and can be modified by a transformation. Data classification labels are associated with elements as standard XML attributes. In the example we omit all labels with value 0 for all tags for simplicity. CVPs are defined as labelled XPath expressions. Figure 4.6 shows an example implementation for the *MedInfo()* and *NamesOrAddresses()* CVPs that verify whether the data contains personal names or addresses and medical information, respectively. The first and second lines in both listings indicate the CVP's ID and partial label it generates if verified, respectively. For example, an XML element returned by the *NamesOrAddresses* CVP would be associated with sensitivity level 1 for the *privacy* tag, unless it were also returned by the *MedInfo* CVP, in which case it would be associated with sensitivity level 2. Figure 4.6 does not show a CVP for level 0 as it simply returns the root of the XML document (in other words, all documents are not protected by default for the *privacy* domain). Therefore, these three CVPs, together with the specification of the deny and allow sets shown in section 4.4.1, completely define the *privacy* tag (since no obligations are specified). We implement particular CVPs such as a facial recognition CVP for images and the *Req* CVP as Java functions and group them in an API that can be accessed via XPath through the *custom* namespace.

Algorithm 1 and Figure 4.7 show the data labelling process via CVPs. First, all CVPs are run on the newly created data. If any CVP associated with level *l* for tag *t* returns one or more elements, then the sensitivity level for tag *t* in the data classification label of those elements is set to *l* (procedure **setInitialLabel**). If more than

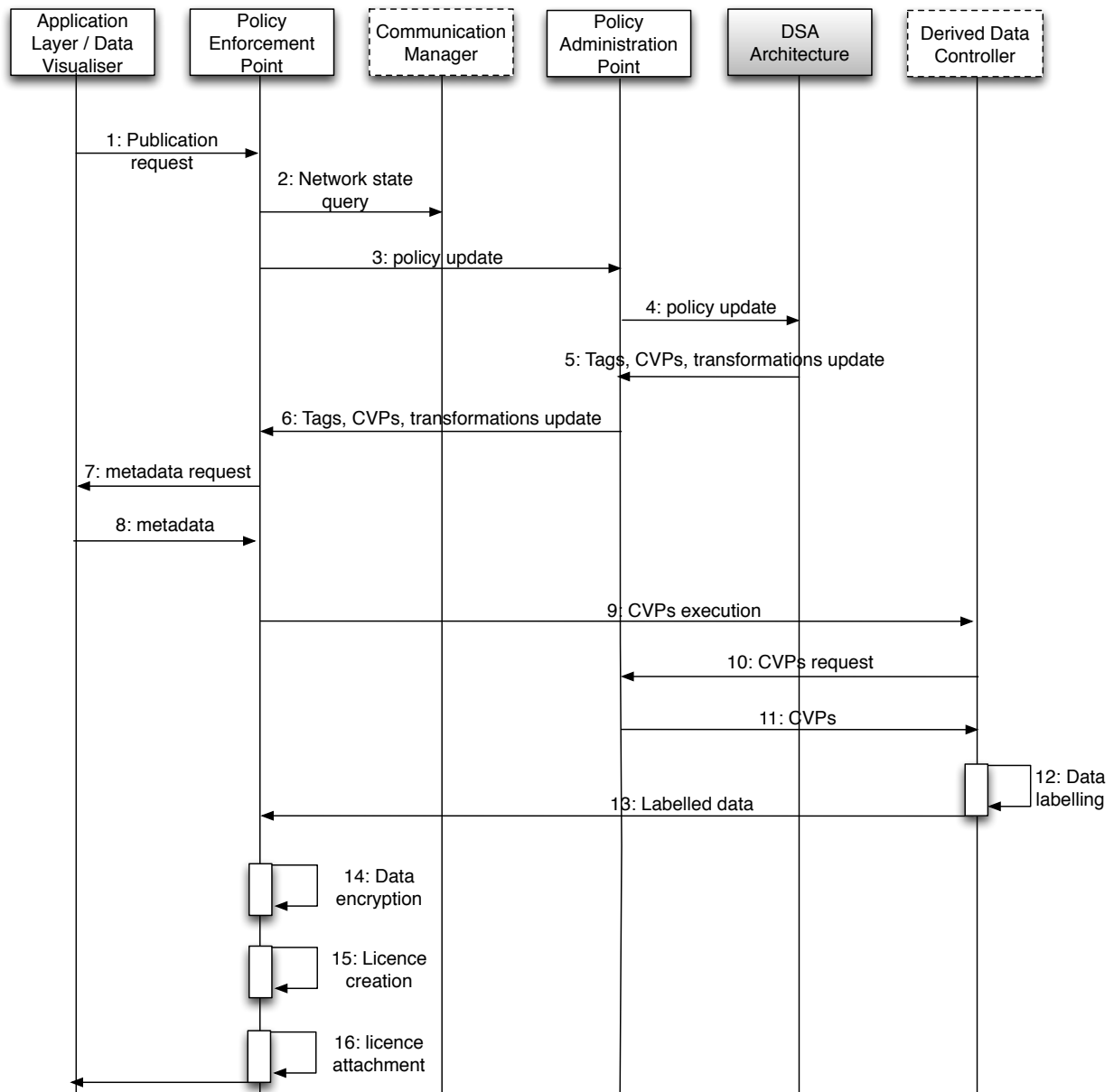


Figure 4.4: Interaction of components for data publication

one CVP have been specified for the same sensitivity level in a tag, then the result is the disjunction of the two CVPs. In other words, just one of them needs to return an element for it to be labelled with the corresponding level. Whenever more than one CVP for the same tag but different level return the same element, the procedure selects the highest sensitivity level. Finally, sensitivity levels can be propagated down the XML tree, from parent nodes to their children (procedure **propagateLevels**). While we let all sensitivity levels propagate down the XML tree, several other approaches [DdVPS02] have been proposed to determine how policies associated with XML elements can propagate and how conflicts between propagated and not-propagated policies can be solved (see Section 2.4.2 for examples). The same solutions can be easily applied to sensitivity domains

```

<VICTIMS>
  <VICTIM status="alive" label="(confidentiality 0),(privacy 1)">
    <NAME>Jane</NAME>
    <SURNAME>Austen</SURNAME>
    <GENDER>F</GENDER>
    <AGE>23</AGE>
    <NATIONALITY>British</NATIONALITY>
    <DOB>1980-04-07</DOB>
    <PHONENUMBER>07877578834</PHONENUMBER>
    <ADDRESS>
      <ADDRESSLINE>390 High Road</ADDRESSLINE>
      <ADDRESSLINE>South Kensington, London</ADDRESSLINE>
      <POSTCODE>SW7 2AZ</POSTCODE>
    </ADDRESS>
    <CONDITIONS label="(confidentiality 0),(privacy 2)">
      <SYMPTOMS>
        <SYMPTOM active="yes">
          <DESCRIPTION>head bleeding</DESCRIPTION>
          <FIRST-AID date="2000-01-12T12:33:00Z">bandaging</FIRST-AID>
          <SUGGESTEDTHERAPY>apply stitches</SUGGESTEDTHERAPY>
        </SYMPTOM>
      </SYMPTOMS>
    </CONDITIONS>
  </VICTIM>
</VICTIMS>

```

Figure 4.5: An example XML structure labelled by CVPs prior to level propagation

<pre> <CVP id="MedInfo"> <LABEL label="privacy,2"/> <XPATH>//CONDITIONS //MEDICAL_HISTORY</XPATH> </CVP> </pre>
<pre> <CVP id="NamesOrAddresses"> <LABEL label="privacy,1"/> <XPATH>//VICTIM[NAME or SURNAME or ADDRESS]</XPATH> </CVP> </pre>

Figure 4.6: Examples of content verification procedure

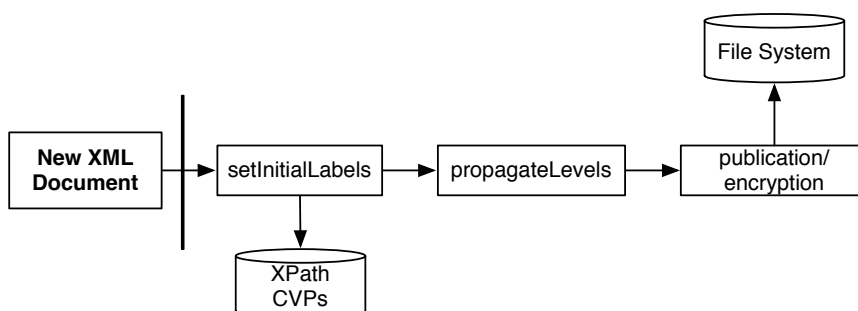


Figure 4.7: Data labelling procedure

and levels. We could for example specify a *propagation bound* for each tag, so that sensitivity levels above the specified bound would propagate down the tree. We could otherwise define tags as *recursive* or *local*, indicating whether the corresponding levels should propagate or not. However, deciding *a priori* which tags propagate and which ones do not may lead to errors as it is extremely difficult to foresee all information that

Algorithm 1: XML labelling procedure.

Input: An XML tree T , a set of CVPs P , a set of domains TS

Output: The labelled XML tree T

setInitialLabel(T, P);

propagateLevels(root(T), TS);

Procedure setInitialLabel(T, P):

forall CVPs $cvp \in P$ **do**

$t = cvp.tag$;

$l = cvp.level$;

$E = \{ \text{Element } e \mid e \in T, \text{ cvp returns } e \}$;

forall elements $e \in E$ **do**

if $l \geq \text{currentLevel}(e, t)$ **then**

$\text{setLevel}(e, l, t)$;

end

end

end

Procedure propagateLevels(e, TS):

forall elements $c \in \text{children}(e)$ **do**

forall tags $t \in TS$ **do**

$\text{parentLevel} = \text{currentLevel}(e, t)$;

$\text{childLevel} = \text{currentLevel}(c, t)$;

if $\text{childLevel} \leq 0 \wedge \text{parentLevel} > 0$ **then**

$\text{setLevel}(c, \text{parentLevel}, t)$;

end

end

 propagateLevels(c, TS);

end

will be protected under a specific tag, and thus to decide whether its content (i.e. subelements) will need to be protected as well or not. Different XML elements protected under the same tag may even have different requirements for their subelements, and creating a different tag for each would cause the DSA specification to be over-complicated. We therefore take the conservative decision of letting all sensitivity levels propagate down the XML tree. This implies that different sensitivity levels for the same tag could be propagated down to the same element. For example, consider a node classified with level 1 for a given tag, and its parent node classified with level 2 for the same tag. The parent's classification would be inherited by the child node that would then be classified with both level 1 and 2 for the same tag. A conflict resolution strategy must therefore be applied to decide which level elements are actually associated with. Our strategy overrides default values (* and 0) with the closest non-default ancestor's levels for the same tags. Non-default levels are instead not overridden by propagated ones, as the *most specific policy applies* strategy dictates. This is reasonable since information that is considered public for a given tag and is thus labelled with level 0 or * can become sensitive when included

```

<VICTIMS>
  <VICTIM status="alive" label="(confidentiality 0),(privacy 1)">
    <NAME label="(confidentiality 0),(privacy 1)">Jane</NAME>
    <SURNAME label="(confidentiality 0),(privacy 1)">Austen</SURNAME>
    <GENDER label="(confidentiality 0),(privacy 1)">F</GENDER>
    <AGE label="(confidentiality 0),(privacy 1)">23</AGE>
    <NATIONALITY label="(confidentiality 0),(privacy 1)">British</NATIONALITY>
    <DOB label="(confidentiality 0),(privacy 1)">1980-04-07</DOB>
    <PHONENUMBER label="(confidentiality 0),(privacy 1)">07877578834</PHONENUMBER>
    <ADDRESS label="(confidentiality 0),(privacy 1)">
      <ADDRESSLINE label="(confidentiality 0),(privacy 1)">390 High Road</ADDRESSLINE>
      <ADDRESSLINE label="(confidentiality 0),(privacy 1)">South Kensington, London</ADDRESSLINE>
      <POSTCODE label="(confidentiality 0),(privacy 1)">SW7 2AZ</POSTCODE>
    </ADDRESS>
    <CONDITIONS label="(confidentiality 0),(privacy 2)">
      <SYMPTOMS label="(confidentiality 0),(privacy 2)">
        <SYMPTOM active="yes" label="(confidentiality 0),(privacy 2)">
          <DESCRIPTION label="(confidentiality 0),(privacy 2)">head bleeding</DESCRIPTION>
          <FIRST-AID date="2000-01-12T12:33:00Z" label="(confidentiality 0),(privacy 2)">bandaging</FIRST-AID>
          <SUGGESTEDTHERAPY label="(confidentiality 0),(privacy 2)">apply stitches</SUGGESTEDTHERAPY>
        </SYMPTOM>
      </SYMPTOMS>
    </CONDITIONS>
  </VICTIM>
</VICTIMS>

```

Figure 4.8: An example XML structure labelled by CVPs after level propagation

into a sensitive XML container (e.g. an address included in a person's profile). Usage of information that was instead already associated with a non-default usage policy should not be further restricted to allow the original intended uses. Procedure **propagateLevels** enforces this behaviour during the data labelling process. In the example depicted in Figure 4.5 all XML elements but the `<VICTIM>` and `<CONDITIONS>` ones are labelled, according to the CVPs in Figure 4.6, with the default value 0 for all domains, i.e. they are public. However, after levels from the `<VICTIM>` and `<CONDITIONS>` nodes are propagated, all elements contained in the `<VICTIM>` node are associated with privacy level 1 and all elements contained in the `<CONDITIONS>` node with privacy level 2. This means that all privacy-sensitive information is labelled with the correct sensitivity level. This situation is depicted in Figure 4.8. Conflicts between sensitivity levels must not be confused with conflicts between policies corresponding to different tags. These are in fact solved when authorisation requests are issued and treated with a *deny overrides* conflict resolution strategy, as dictated by formula 4.4.

With this mechanism, once a new XML document has been created, the derived data controller associates each of its elements with a data label L_i . The PEP then encrypts elements labelled with the same label L_i with the same symmetric key k_i . For each element, its name, attributes (excluding the data label) and data content (if any) are encrypted, but the overall XML structure is maintained. A symmetric key k_m is also generated and used to encrypt the document's metadata. The set of keys $\{k_1 \dots k_n\}$ and k_m , encrypted with the public key of the chosen evaluation authority, are then included into a publishing licence, together with a reference to the authority itself. The license, the encrypted metadata and the encrypted data are then included in the same

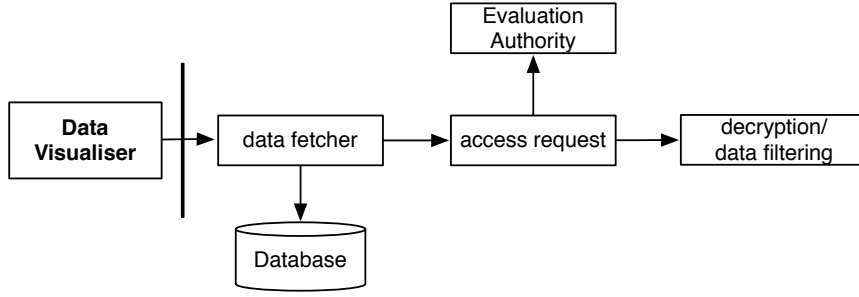


Figure 4.10: Data access procedure

sensitivity level for the tag, as described by formula 4.1. For example, given the label (privacy 1,integrity 0) for the lattice depicted in Figure 4.3, the process reconstructs the policies $P_{read}^{p,1}$ and $P_{read}^{i,0}$. The client's PEP queries its local PIP to retrieve the required credentials and sends them to the requesting authority. The set of keys corresponding to the data classification labels whose policies are satisfied is then sent back to the client's PEP that can decrypt the elements of the document associated with them. The client's PEP also removes from the document view all XML elements that cannot be decrypted and whose contained elements are also not accessible. If at least one of the contained elements is instead accessible, the PEP includes the container element in the document view hiding both its (encrypted) tag name and attributes. This process assumes that if the original document had to comply with a specific template or format (e.g. an XML schema or DTD), this is ignored in the filtered document. Figure 4.10 depicts a simplified description of the data access procedure.

4.7.4 Data Transformation

In our implementation, transformations are specified as a combination of an XQuery and a set of *labelling rules* of the form $(L^c, L^g, L^-, L^+, \text{input docs}, \text{XPath expression})$. Since XQueries can produce complex XML data structures, simply associating a whole XQuery with a classification label, a general declassification label, a relative declassification label and a decisional label implies that only the root element of the output can be labelled, or that all elements are equally labelled. This is indeed a possibility, but we aim to provide a more flexible solution that allows each element contained in the output to have a different label. To do so we associate each transformation with a set of *labelling rules* whose XPath expressions identify elements in the output document that must be labelled according to the corresponding transformation labels. Each labelling rule also contains a set of references to input documents that are used by the transformation to create the output identified by the XPath expression. This is done because not all data accessed by the XQuery engine to produce the output is used for each of the output's subelements. For example, consider a document containing only one XML element labelled as $\{(p,1) (i,0)\}$ and another document also containing only one XML element,

```

<TOXIC_ANALYSIS date="{fn:current-dateTime()}">
  {
    if (every $victim in fn:doc("victims.xml")//VICTIM satisfies
        (some $symptom in $victim/CONDITIONS/SYMPTOMS/SYMPTOM satisfies
            $symptom/DESCRIPTION/text() = "breathing problems"
        or $symptom/DESCRIPTION/text() = "red eyes"))
        then "toxic contamination in the area!"
        else "no contamination!"
    }
  }
</TOXIC_ANALYSIS>
<RULES>
  <RULE clabel="(confidentiality 1)" glabel="(privacy 0)">
    <XPATH>//TOXIC_ANALYSIS</XPATH>
    <INPUTDATA file="victims" />
  </RULE>
</RULES>

```

Figure 4.11: Example XQuery transformation

this time labelled as $\{(p,0) (i,1)\}$. Now consider a transformation appending one element to the other. If we considered both input documents when calculating the labels for the two elements in the output document, we would label both with label $\{(p,1) (i,1)\}$, which is not a desirable result. Instead, we specify this very simple example transformation as a combination of two labelling rules. The first rule contains an XPath expression identifying the first element of the output document and a reference to the first input document. The second rule contains instead an XPath expression identifying the second element of the output document and a reference to the second input document. When calculating the labels for the two elements in the output document, we can now distinguish between the first element to which only the first input document contributed, and the second element to which only the second input document contributed. Figure 4.11 depicts a very simple example of a transformation that uses victims' information to evaluate the possibility of a toxic contamination in the area of the accident depending on whether all victims have experienced either *breathing problems* or *red eyes*. The estimate is inserted into an XML document containing only one *toxic analysis* element with a date attribute. The only labelling rule points to the root element of the output and contains a classification label and a general declassification label so that the element is confidential (the minimum sensitivity for the confidentiality domain is set to 1) but not private (the maximum sensitivity level for the privacy domain is set to 0). The output in fact does not contain any private information but is considered confidential as it contains information on a toxic threat which might cause panic if publicly available. The relative declassification label L^- and decisional label L^d are not specified and are therefore set to default values for all tags (1 and *false*, respectively).

The interaction of components for data transformation is shown in Figure 4.12. When a user requests to run a transformation, first a policy evaluation is required to verify whether the user is authorised to access the transformation's input data and to execute the transformation itself. Input documents are decrypted (if they are not already accessible) and a policy evaluation is triggered as for normal data accesses. If the transformation is

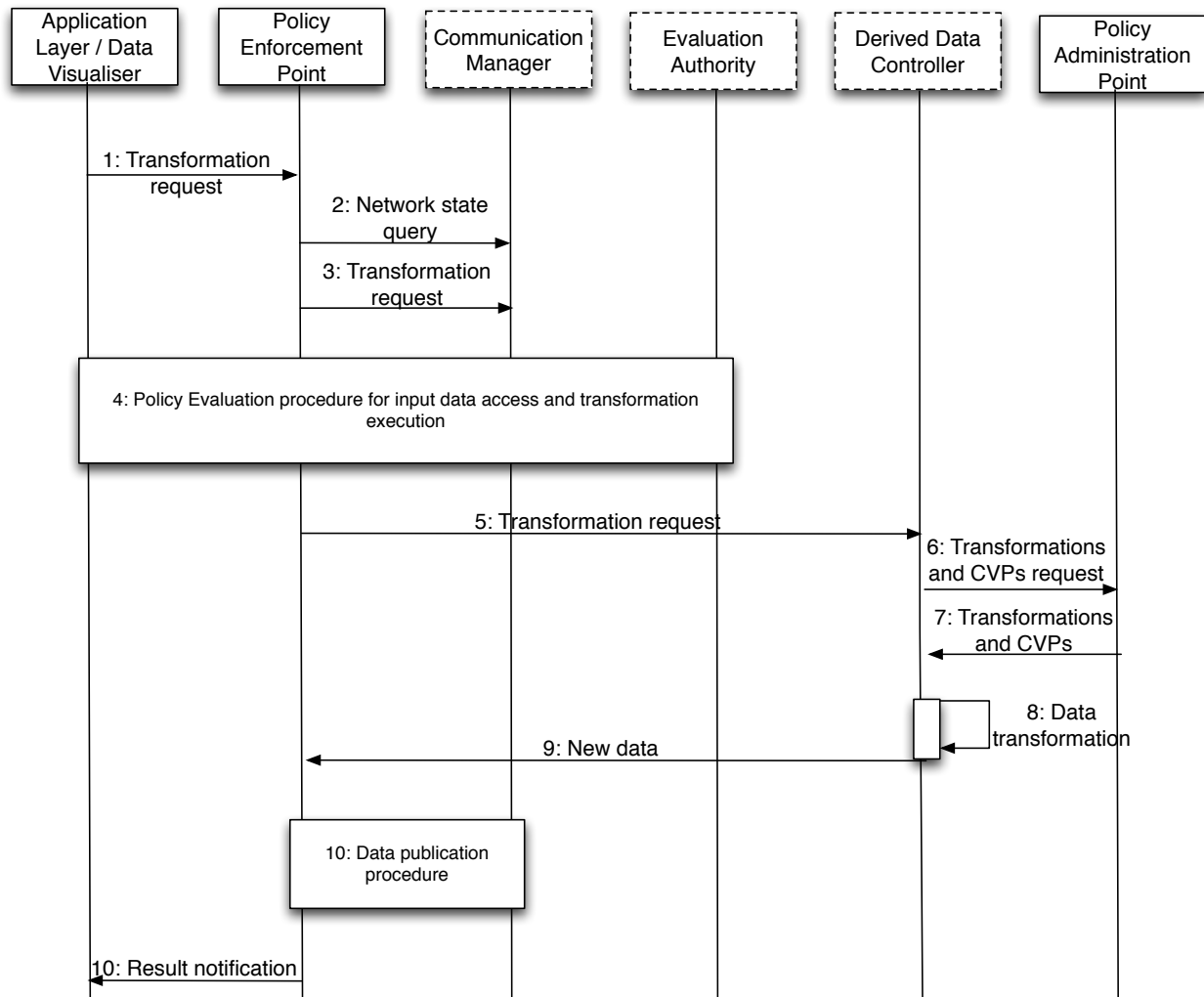


Figure 4.12: Interaction of components for data transformation

authorised, the derived data controller performs it (using an XQuery engine) on the pruned input data resulting from the access request. If the transformation fails because of the pruning, an authorisation denial notification is returned to the user. Consider the toxic analysis transformation shown in Figure 4.11. If the user running the transformation could not access one or more victims’ records showing intoxication symptoms, the transformation might return a “no contamination” result and false information could be disseminated. Therefore, since the transformation may succeed anyway but return an incorrect output, an alternative solution is to deny the transformation request altogether if any of the input documents cannot be accessed completely. Again, the choice of which behaviour to apply depends on the specific application scenario and transformation. For example, in the crisis management scenario we consider, rescuers may favour data availability over concerns on data correctness, or viceversa. Once the transformation has been performed, the output document is published as if it were a completely new document.

Algorithm 2: Label derivation procedure.

Input: A transformation *act*, a set of input XML documents *I*, a set *LR* of labelling rules, a set of domains *TS*

Output: The transformation's labelled XML output *O*

```
O = executeQuery(act,I);
transformLabels(O, LR);
propagateLevels(root(O),TS);
forall XML elements  $e \in O$  do
  dl = deriveDataLabel(getTransformationLabels(e), getDataLabels(I));
  setLabel(e, dl);
end
```

Procedure transformLabels(O, LR)

```
forall Labelling rule  $lr \in LR$  do
  ( $L^c, L^g, L^-, L^t$ ) = lr.transformationLabels;
  D = lr.inputData;
  E = {Element  $e \mid e \in O, lr.XPath \text{ returns } e$ };
  forall elements  $e \in E$  do
    if  $e$  is not contained in I then
       $L_e^c = L_e^c \sqcup L^c$ ;           element  $e$ 's temporary  $L^c$ 
       $L_e^g = L_e^g \sqcup L^g$ ;           element  $e$ 's temporary  $L^g$ 
       $L_e^- = \max(L_e^-, L^-)$ ;       element  $e$ 's temporary  $L^-$ 
       $L_e^t = TrueValues(L_e^t, L^t)$ ; element  $e$ 's temporary  $L^t$ 
       $D_e = D_e \cup D$ ;             element  $e$ 's temporary input data
      labelDerivation( $e, L_e^c, L_e^g, L_e^-, L_e^t, D_e$ );
    end
    else
      copyLabelFromInput( $e, I$ );
    end
  end
end
```

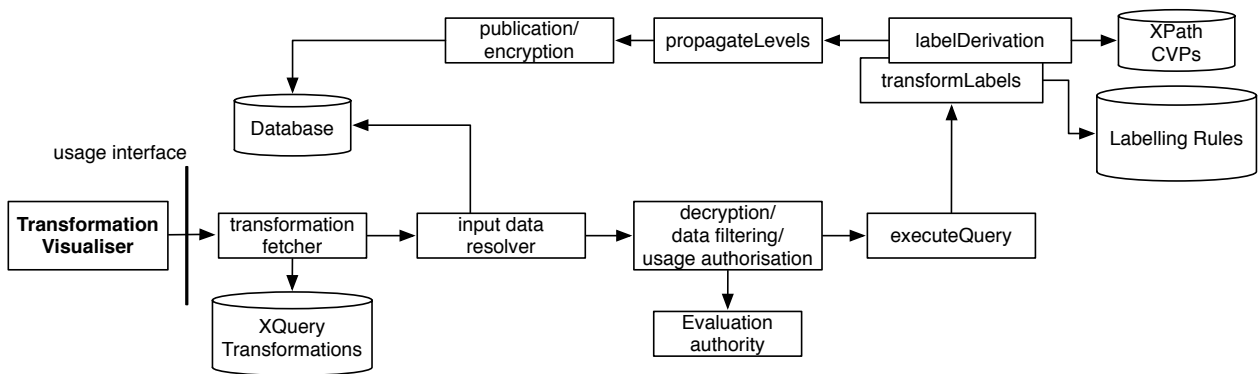


Figure 4.13: Data transformation procedure

Algorithm 2 and Figure 4.13 show the transformation and label derivation procedure. First, the derived data controller executes all the XPath expressions of the labelling rules on the output document (procedure **transformLabels**). For each returned element, the procedure should calculate its derived label according to formula 4.11. However, the same element could be returned by more than one XPath expression associated with different labelling rules. To cater for this case, returned elements are associated with temporary transformation labels ($L_e^c, L_e^g, L_e^-, L_e^t$) and with the set of all input documents associated with the rules that returned them (D_e). This is necessary so that the final data label for each element is created considering all labelling rules whose XPath expression returned the element. The temporary classification label L_e^c is obtained by calculating the upper bound of all classification labels of the labelling rules whose XPath expressions returned the element e . The temporary general declassification label L_e^g is obtained by calculating the upper bound of all general declassification labels of the labelling rules whose XPath expressions returned the element e . The temporary relative declassification label L_e^- is obtained by keeping the highest values amongst all relative declassification labels of the labelling rules whose XPath expressions returned the element e . Finally the temporary decisional label L_e^t is obtained by keeping all true values of all decisional labels of the labelling rules whose XPath expressions returned the element e . The rationale behind the way temporary labels are calculated is that the output's minimum sensitivity is defined by the highest value in all classification labels, while we consider only the least effective declassification has been performed to guarantee the highest protection possible. Once all elements have been associated with a set of temporary transformation labels and a set of input documents, the label derivation procedure **labelDerivation** performs for each element the operation described by formula 4.11, using the temporary transformation labels and input documents in D_e . In particular, the procedure considers all data labels associated with different elements of the input documents as separate inputs. In other words, if one of the input documents contains several XML elements, all of their labels are used as if they came from different documents. For example, a transformation performed on the XML document pictured in Figure 4.5 would consider two separated inputs, the *<VICTIM>* element associated with label $\{(confidentiality\ 0), (privacy\ 1)\}$ and the *<CONDITIONS>* element associated with label $\{(confidentiality\ 0), (privacy\ 2)\}$. Also, if any temporary decisional label associated with a specific element e contains a true value for a tag t , the corresponding CVPs are run. If any CVP returns the element e or any of its subelements e' , then the CVP's sensitivity level is associated with e or e' , respectively. If this conflicts with another applied sensitivity level, the highest one is chosen. Instead, when the transformation simply copies a whole element from one of the inputs, then the corresponding data label remains unchanged. This is obtained by marking every XML element with a unique ID at publication time. Finally, labels are propagated in the output document by the **propagateLevels** procedure.

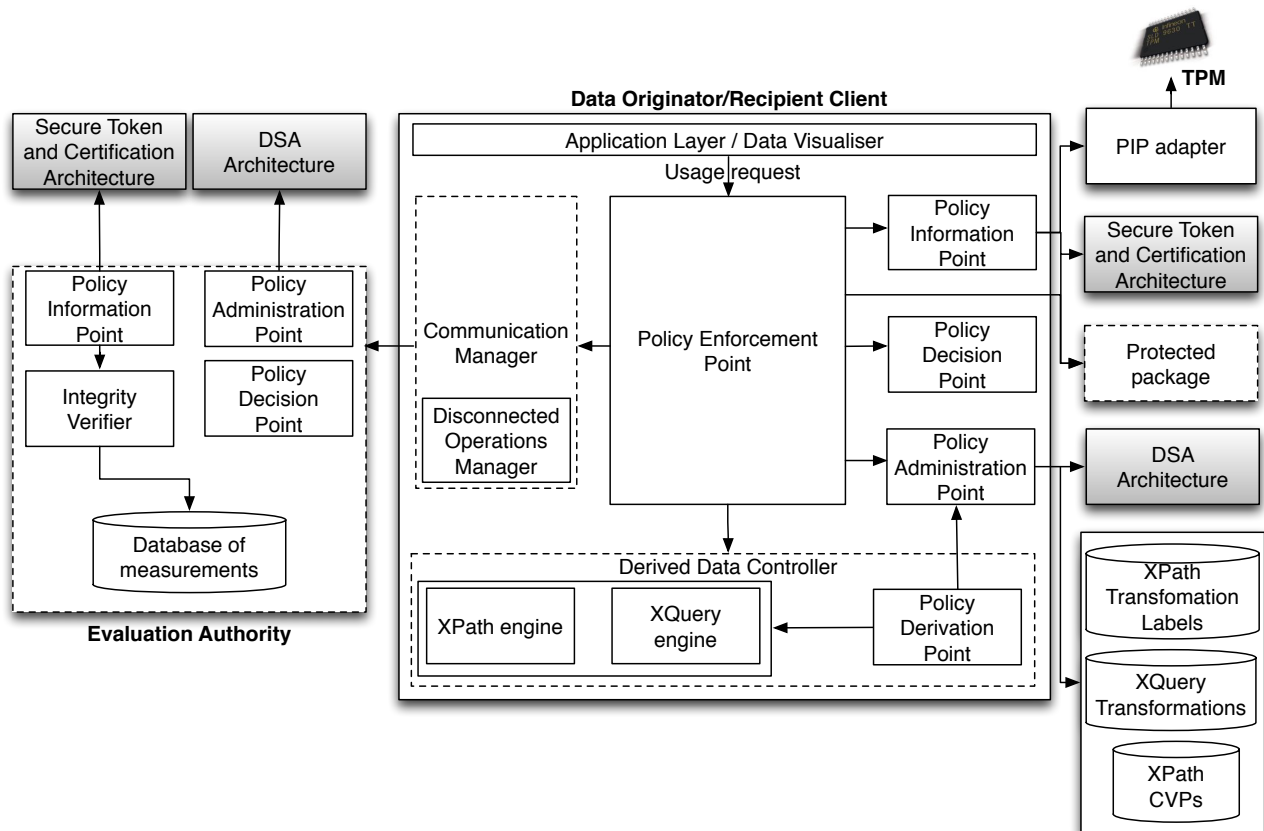


Figure 4.14: High level architecture for derived data control

4.7.5 Components

Figure 4.14 shows the updated version of our high-level architecture including the realised derived data controller. It differs from existing architectures in the role of the application used to access data. User applications usually allow users to read data and modify it. Any performed action, starting from data disclosure to data saving, triggers a policy evaluation to obtain permissions. The policy enforcement point (PEP) receives action requests, coordinates the policy evaluation and returns a response to the application (and possibly the decrypted data). We instead propose a solution where user applications only provide an interface of the list of available transformations (including the possibility to create new data and to access existing data). This list is controlled by the enforcement layer via the PAP that stores all XQuery and XPath functions. The **policy derivation point (PDevP)** is the component responsible for labelling newly created data and transformed data. We emphasise that the integrity of transformations is fundamental to the solution we propose. Transformations that have been tampered with, may in fact lead to incorrect data labelling and thus data leaks. We have therefore integrated trusted computing platforms in our architecture to ascertain the integrity of the components managing the transformations.

4.8 Evaluation

The protection provided by our approach depends on the heuristic criteria represented by both content verification procedures and labelling rules. As such, we can only evaluate the approach by using extensive and real examples to show that it is possible to satisfy complex requirements. In particular, we consider the actual practices regulating data sharing and protection during crisis management operations carried out by public safety organisations in the United Kingdom and studied by *BAE systems plc* [GBLD]. The examples proposed here are thus meant to be realistic as they are drawn from real case studies and practices. The specific scenario we use is the same one described in Section 1.3.

Responders to a crisis are broadly divided into two categories. Category 1 includes emergency services, local authorities, health bodies and environment agencies, while category 2 includes utility and transport companies, strategic health authorities and the Health and Safety Executive (HSE). Each agency involved maintains and deploys three different types of command centres from and towards which data is disseminated: gold command centres that manage the strategic policy in the long term (weeks and months), silver command centres that manage the tactical deployment of responders at the incident site and bronze command centres that operate within the incident site. Figure 4.15 depicts the internal role hierarchies for the police force, ambulance services and fire brigade. Bronze command centres in particular manage all the responders that are deployed directly in the incident area. The figure does not show gold command centres as their operational role in the short term is limited and not interesting for our study. The role of command centres as *data hubs* is particularly important since they are the nodes where data will be transformed and re-distributed. Data is disseminated as *tactical situation objects (TSO)*, i.e. as XML documents containing information relevant to the responders [pc, GBLD], thus making this scenario perfect for demonstrating our approach.

4.8.1 Disseminated Data

The information that category 1 responders share can be divided into three categories: background information (e.g. road maps, important locations, weather), coordination information (e.g. status of each service's response, evacuation and rendezvous details) and incident information (e.g. the type of incident, casualties, possible threats).

Table 4.1 shows the type of sensitive information gathered and disseminated during the crisis and the responders who need to access it. Individual responders receive information from their bronze command centres and are therefore not included in the table. Similarly, we do not include gold command centres due to their non-operational short-term role. Non-sensitive information is also disseminated (e.g. street maps and closures,

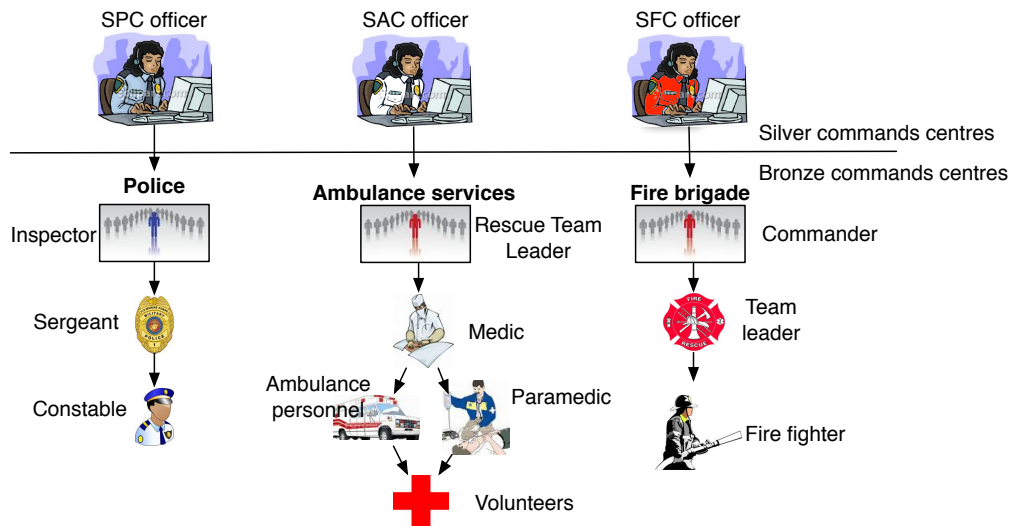


Figure 4.15: Example role hierarchies

RESPONDERS	ACCESSED INFORMATION
<i>bronze ambulance command (BAC)</i>	Patient's personal information and conditions, dangerous locations, hazardous materials involved
<i>bronze police command (BPC)</i>	Patients' personal information, dangerous locations, type of violence on scene, weapons involved, witness information, hazardous materials involved
<i>bronze fire brigade command (BFC)</i>	Patients' personal information, dangerous locations, hazardous materials involved
<i>silver ambulance command (SAC)</i>	Nearby hospitals and care centres
<i>silver police command (SPC)</i>	Chemical-related information, resident information, CCTV images
<i>silver fire brigade command (SFC)</i>	Resident information, nearby hospitals and care centres

Table 4.1: Gathered and disseminated information

weather forecasts, rendezvous details etc.) but we do not explicitly include it in the table.

Table 4.2 shows instead how command centres transform the gathered information into new data to disseminate. The bronze ambulance command centre (BAC) uses information on the conditions of victims to generate a document containing an estimate of the risk of toxic contamination of the area. This estimate is sent to the silver police command centre (SPC) that combines it with a document detailing the contents of the vehicles involved in the accident (obtained from the owner organisations) and from the responders' reports on chemical

CREATOR	INPUT	OUTPUT	ACCESSED BY
BAC	patients' conditions	toxic contamination estimate	SPC
SPC	hazardous materials involved chemical-related information <i>toxic contamination estimate</i>	toxic contamination assessment (short term)	All bronze commands
		toxic contamination assessment (long term)	All silver commands
	resident information type of violence on scene weapons involved <i>toxic contamination assessment</i>	severity and type of the incident	BPC
	CCTV images	blurred CCTV images	Media, BFC , BPC
		plans for dangerous situations	BPC
	<i>casualty document</i> commander's statement	official statement	Media, SPC
SAC	patients' conditions <i>toxic contamination assessment</i> nearby hospitals and care centres	patients' evacuation plans	BAC
	patients' conditions	toll of casualties	SPC

Table 4.2: Transformed and disseminated information

spillage in the area to produce an assessment of the short and long term chemical threats and risks. Warnings on short-term threats must then be disseminated to all bronze command centres so that they can take precautions during the rescue operations, while information on long-term threats must be kept reserved for use after the current situation has been settled. The silver police command also uses the information on the unfolding events and the involved people to determine the severity and type of the accident. This information is then disseminated amongst police responders to better organise their efforts. Images obtained from a CCTV system active in the area are blurred by the police so that faces and details such as vehicles' plates are removed and the images can be disseminated to the media. Finally, the police disseminates the plan to be followed in case of dangerous situations. While the creation of this document is also based on the information gathered by responders, it is most probably the result of human editing rather than an automatic process. This is why we do not consider any input for its creation.

The silver ambulance command centre (SAC) combines information on patients' conditions and nearby hospitals and care centres to generate an evacuation plan for the incident's victims. It also accounts for the casualty tolls as new information is received. The toll is then used by the police to output an official statement to be released to the media.

4.8.2 Sensitivity Domains

To protect the tactical situation objects containing the data described above, the three involved agencies stipulate a DSA containing the following sensitivity domains:

- a *privacy* domain (P) to protect the patients' personal and medical information;
- an *imageprotection* domain (IP) to protect confidential images;
- a *confidentiality* domain (C) to protect information that may cause panic and that must be carefully managed by the police;
- a *media* domain (M) to control the release of information to the media;
- a *commercial* domain (CO) to protect commercially sensitive information;
- an *utility* domain (U) to protect information for the involved utility companies (e.g. a problem with a railway caused by the incident).

Table 4.3 shows the compound Boolean expressions for the six defined tags from which the policy lattice can be derived. The policies they represent are a more elaborated and sophisticated version of the policies described in [GBLD]. For the sake of simplicity we again consider the *read* action as the only possible one. Policies controlling the use of transformations simply authorise command centres that need them to create the data of Table 4.2. Also note that in the constraint set definitions we use a dot notation only to solve ambiguities in attributes' names when they refer to the TSO's metadata. The considered tags are:

- **Privacy:** this tag is associated with two non-default sensitivity levels: level 1 protects personal information so that all rescuers operating in the area or in a hospital can access it, while level 2 protects information on medical conditions so that only the ambulance services or responders with a first-aid qualification can access it.
- **Confidentiality:** this tag is associated with three non-default sensitivity levels: level 1 protects information about the immediate dangers in the area (e.g. the dangerous locations) so that access can be granted to all rescuers acting in the specific emergency situation. Level 2 protects more sensitive information such as information about witnesses in the emergency area or plans for dangerous situations, so that access can be granted only to responders directly authorised by the bronze police command centre, to higher-ranked police officers and to fire brigade commanders. Level 3 protects high-confidentiality information that is not immediately useful for responders but needs first to be evaluated at a silver command centre or by police inspectors.

- **Imageprotection:** this tag is associated with three non-default sensitivity levels: level 1 protects images where gruesome details are still visible so that only rescuers can access them (fire fighters and ambulance services are also required to be operating in the area). Level 2 protects images that show individuals' faces so that only police responders can access them. The location of the recipient is not important for policemen since, for example, any police officer might recognise individuals in the video with a criminal record. Finally, level 3 protects images that have not been sanitised yet (and thus all gathered images), since besides faces they might show sensitive details (such as the names of the possibly military companies whose trucks have been involved in the accident) that should be removed before releasing the images even to low-ranked police officers.
- **Media:** this tag is associated with only one non-default sensitivity level that protects the official police statements that cannot be released to the public yet. This is done because the statement is publicly released only when certain conditions are met and should be kept confidential while it is being updated.
- **Commercial:** this tag is associated with only one non-default sensitivity level that protects information coming from companies involved in the accident. The companies owning the involved vehicles might for example release information on the chemicals contained in the lorries. This information can be accessed only by the company itself and the silver police command centre (that receives it from the company and publishes it).
- **Utility:** this tag is associated with only one sensitivity level that protects information of interest for the involved utility companies. In our example we consider British Gas as the only company involved.

In the specific scenario we are considering no obligations are specified for responders accessing protected information. The chains of sensitivity levels for the specified tags are thus the following:

$$\begin{aligned}
P &: (\{false\}, \{(true)\}, true) \rightarrow (D_{p,1}, A_{p,1}, NamesOrAddresses()) \rightarrow (D_{p,2}, A_{p,2}, MedInfo()) \\
C &: (\{false\}, \{(true)\}, true) \rightarrow (D_{c,1}, A_{c,1}, ConfCheck1()) \rightarrow (D_{c,2}, A_{c,2}, ConfCheck2()) \rightarrow (D_{c,3}, A_{c,3}, Req(3, C)) \\
IP &: (\{false\}, \{(true)\}, false) \rightarrow (D_{ip,1}, A_{ip,1}, false) \rightarrow (D_{ip,2}, A_{ip,2}, false) \rightarrow (D_{ip,3}, A_{c,3}, isImage()) \\
M &: (\{false\}, \{(true)\}, true) \rightarrow (D_{m,1}, A_{m,1}, VictimsCount()) \\
CO &: (\{false\}, \{(true)\}, true) \rightarrow (D_{co,1}, A_{co,1}, Req(1, CO)) \\
U &: (\{false\}, \{(true)\}, true) \rightarrow (D_{u,1}, A_{u,1}, Req(1, U))
\end{aligned}$$

where the *NamesOrAddresses()* and *MedInfo()* CVPs have already been described in Section 4.7.2, and Figure 4.16 depicts the others. In particular, the *ConfCheck1()* CVP checks whether the reports created by

		Sensitivity Levels			
		0	1	2	3
P	$D_{p,i}$	<i>false</i>	$zone \neq hospital \wedge$ $zone \neq TSO.emrArea$	-	-
	$A_{p,i}$	<i>true</i>	$role \geq BACVolunteer \vee$ $role \geq BPCConstable \vee$ $role \geq BFCFirefighter$	$role \geq BACVolunteer \vee$ $first - aid = true$	-
C	$D_{c,i}$	<i>false</i>	$mission \neq TSO.emrCode$	-	-
	$A_{c,i}$	<i>true</i>	$role \geq BACVolunteer \vee$ $role \geq BPCConstable \vee$ $role \geq BFCFirefighter$	$role \geq BPCSergeant \vee$ $BPCAuthDocs = TSO.type \vee$ $role \geq BFCFireCommander$	$role \geq BPCInspector$
IP	$D_{ip,i}$	<i>false</i>	$zone \neq TSO.emrArea \wedge$ $(role \geq BACVolunteer \vee$ $role \geq BFCFirefighter)$	-	-
	$A_{ip,i}$	<i>true</i>	$role \geq BACVolunteer \vee$ $role \geq BPCConstable \vee$ $role \geq BFCFirefighter$	$role \geq BPCConstable$	$role \geq SPCOfficer$
M	$D_{m,i}$	<i>false</i>	-	-	-
	$A_{m,i}$	<i>true</i>	$role \geq SPCOfficer$	-	-
CO	$D_{co,i}$	<i>false</i>	-	-	-
	$A_{co,i}$	<i>true</i>	$id = TSO.owner \vee$ $role \geq SPCOfficer$	-	-
U	$D_{u,i}$	<i>false</i>	-	-	-
	$A_{u,i}$	<i>true</i>	$role \geq SPCOfficer \vee$ $id = BritishGas$	-	-

Table 4.3: Compound Boolean expressions for the crisis management scenario

responders contain information on the dangerous locations of the area and hazardous materials involved in the emergency. The *ConfCheck2()* CVP is instead used to protect information regarding possible witnesses of the incident and the plans for dangerous situations. The *isImage* CVP verifies whether the document contains any image, while the *VictimsCount* CVP verifies if the number of victims in the official police statement is greater than one, in which case the statement must not be released before further approval. Finally, the *Req* CVPs are used to ask the data originator whether she wants to protect the data with a specific value for a specific tag (passed as parameters). CVPs have not been specified for all levels of all tags. When missing, they are considered to be XPath expression not returning any element.

Note that the defined CVPs assume that responders only edit document templates, filling them with the information they gather while moving in the incident area. This makes it possible for the XPath expressions to search the documents properly and find the relevant information.

<CVP id="ConfCheck1"> <LABEL label="confidentiality,1"/> <XPATH>//report/dangerouslocations //report/materials</XPATH> </CVP>
<CVP id="ConfCheck2"> <LABEL label="confidentiality,2"/> <XPATH>//report/plans //witnesses</XPATH> </CVP>
<CVP id="Req(3,C)"> <LABEL label="confidentiality,3"/> <XPATH>/child::*[custom:Req("3","C")]</XPATH> </CVP>
<CVP id="isImage"> <LABEL label="imageprotection,3"/> <XPATH>//child::*[custom:isImage(/child::*)]</XPATH> </CVP>
<CVP id="VictimsCount"> <LABEL label="media,1"/> <XPATH>//statement[VICTIMS_COUNT > 0]</XPATH> </CVP>
<CVP id="Req(1,CO)"> <LABEL label="commercial,1"/> <XPATH>/child::*[custom:Req("1","CO")]</XPATH> </CVP>
<CVP id="Req(1,U)"> <LABEL label="utility,1"/> <XPATH>/child::*[custom:Req("1","U")]</XPATH> </CVP>

Figure 4.16: Example Content Verification Procedures

4.8.3 Data Transformations

The so-defined chains of sensitivity levels satisfy the access and protection requirements for gathered data specified in Table 4.1. Once protected data has reached the bronze and silver ambulance command centres and the silver police command centre, it is used as input for several data transformations whose output must also be protected. Let us consider the transformations described in Table 4.2. The transformation run by the bronze ambulance command centre that generates an estimate of toxic contamination of victims is equivalent to the *toxic analysis* transformation depicted in Figure 4.11 and described in Section 4.7.4. We have redefined it in Figure 4.17 so that its output is protected for the highest-ranked police responders, thus including the SPC. We have therefore labelled the transformation with only one labelling rule targeting the root of the output document with a classification label $L^c = \{(C, 3)\}$ and a general declassification label $L^s = \{(P, 0)\}$. The output document may indeed include further details on the contamination, depending on the kind of controls responders can perform on victims.

The toxic contamination estimate is used by the silver police command together with the reports from the responders on the hazardous materials present in the incident area and the information on the chemicals contained in the involved lorries (that it can access) to run the *toxic contamination assessment* transformation depicted

```

<TOXIC_ESTIMATE date="{fn:current-dateTime()}">
{
  if (every $TOXIC_ESTIMATE in fn:doc("victims.xml")/VICTIM satisfies
    (
      some $symptom in $victim/CONDITIONS/SYMTOMS/SYMTOM satisfies
      (
        $symptom/DESCRIPTION/text() = "breathing problems"
        or $symptom/DESCRIPTION/text() = "red eyes"
      )
    )
  )
  then "positive toxic contamination"
  else "negative toxic contamination!"
}
</TOXIC_ESTIMATE>
<RULES>
<RULE clabel="(confidentiality 3)" glabel="(privacy 0)">
  <XPATH>/TOXIC_ESTIMATE</XPATH>
  <INPUTDATA file="victims" />
</RULE>
</RULES>

```

Figure 4.17: Toxic contamination estimate transformation

```

<TOXIC_ASSESSMENT date="{fn:current-dateTime()}">
{
  <SHORT>custom:assessment(true,fn:doc("TOXIC_ESTIMATE.xml"),fn:doc("chemicals.xml"),fn:doc("reports.xml"))</SHORT>
  <LONG>custom:assessment(false,fn:doc("TOXIC_ESTIMATE.xml"),fn:doc("chemicals.xml"),fn:doc("reports.xml"))</LONG>
}
</TOXIC_ASSESSMENT>
<RULES>
<RULE clabel="(confidentiality 1)" glabel="(privacy 0, commercial 0)">
  <XPATH>/TOXIC_ASSESSMENT/SHORT</XPATH>
  <INPUTDATA file="TOXIC_ESTIMATE" />
  <INPUTDATA file="CHEMICALS" />
  <INPUTDATA file="REPORTS" />
</RULE>
<RULE clabel="(confidentiality 3)" glabel="(privacy 0, commercial 0)">
  <XPATH>/TOXIC_ASSESSMENT/LONG</XPATH>
  <INPUTDATA file="TOXIC_ESTIMATE" />
  <INPUTDATA file="chemicals" />
  <INPUTDATA file="reports" />
</RULE>
</RULES>

```

Figure 4.18: Toxic contamination assessment transformation

in Figure 4.18. We do not delve into the details of how such an assessment can be created and simply use a Java function to obtain a short-term and a long-term assessment. The two corresponding sub-elements are assigned confidentiality level 1 and 3, respectively, while all private information (that could be contained in the responders' reports) and commercial information is removed. We have thus labelled the transformation with two labelling rules targeting the *<SHORT>* and *<LONG>* sub-elements of the output with a classification label $L^c = \{(C, 1)\}$ and $L^c = \{(C, 3)\}$, respectively. Both rules also include a general declassification label $L^s = \{(P, 0)(CO, 0)\}$ which means the output document does not contain any private or commercially sensitive information.

The silver police command centre also uses the information contained in the responders' reports, the information on the residents of the incident area and the toxic contamination assessment to generate a general

```

<EMERGENCY date="{fn:current-dateTime()}">
{
  if (some $description in fn:doc("reports.xml")/report/dangerouslocations satisfies
  (
    $description/violence/weapons/@weaponsinvolved = "true"
  ) or some $resident in fn:doc("residents.xml")/residents satisfies
  (
    $resident/@dangeroussubject = "true"
  )
  then return {
    <DANGER type="violence">dangerous subjects in the area</DANGER
    <PROCEDURE> look for weapons </PROCEDURE>
  }
  if (fn:doc("TOXIC_ASSESSMENT.xml")/SHORT/@toxicitylevel > 3)
  then return {
    <DANGER type = "toxicity">potential toxic threat</DANGER>
    <PROCEDURE> wear protective masks </PROCEDURE>
  }
}
</EMERGENCY>
<RULES>
<RULE clabel="(confidentiality 1)" glabel="(confidentiality 1, privacy 0)">
  <XPATH>/EMERGENCY</XPATH>
  <INPUTDATA file="reports" />
  <INPUTDATA file="TOXIC_ASSESSMENT" />
  <INPUTDATA file="residents" />
</RULE>
</RULES>

```

Figure 4.19: Emergency document transformation

document containing the type and severity (or *threat level*) of the incident. A simple version of the transformation is shown in Figure 4.19. The transformation contains warnings about the possible dangers that could be present in the incident area, including suggestions of how to act for example if any resident is considered dangerous or if weapons have been used, or if there is a contamination with a release of toxic chemicals thought to be above a certain threshold. The transformation removes all private content that might be contained in the document containing information about the residents of the area. Confidentiality must also be lowered with respect to the confidentiality level of the toxic assessment document because all rescuers in the area should be aware of the current risk, but must also be kept to a minimum level so that only rescuers can access it. We have thus labelled the transformation with only one labelling rule targeting the root of the output document with a classification label $L^c = \{(C, 1)\}$ and a general declassification label $L^s = \{(C, 1)(P, 0)\}$. Note that the transformation can be arbitrarily more complex and return a document with different sensitivity levels for different sub-elements, so that only specific responders can access them.

According to the content verification procedures associated with the *imageprotection* tag, all images are initially labelled with an *imageprotection* sensitivity level equal to 3. The image blurring transformation (Figure 4.20) used by the media notification service invokes a Java function to decrease image resolution. Several runs of the transformation progressively reduce the amount of sensitive information contained in the images,

```

<IMAGES>
  {for $image in fn:doc("images.xml")/IMAGES/IMAGE
  return
  <IMAGE>custom:ResolutionDecrease($image)</IMAGE>}
</IMAGES>
<RULES>
  <RULE rlabel="(imageprotection 0.5)">
    <THRESHOLD>0.5</THRESHOLD>
    <XPATH>/IMAGES</XPATH>
    <INPUTDATA file="images"/>
  </RULE>
</RULES>

```

Figure 4.20: Images blurring transformation.

```

<VICTIMS_COUNT date="{fn:current-dateTime()}">
  {
    fn:count(fn:doc("victims.xml")/VICTIM[@status="deceased"])
  }
</VICTIMS_COUNT>
<RULES>
  <RULE clabel="(confidentiality 3)" glabel="(privacy 0)">
    <XPATH>/VICTIMS_COUNT</XPATH>
    <INPUTDATA file="victims"/>
  </RULE>
</RULES>

```

Figure 4.21: Victims count transformation

up to the point that the document can be publicly disseminated. The blurring transformation is therefore associated with a labelling rule targeting the root of the output document with a relative declassification label $L^- = \{(IP, 0.5)\}$ with threshold parameter 0.5 that ensures that every time an image is blurred, its *imageprotection* level decreases. Officers of the silver police command, who are the only entities who can access the data, decide how many times the transformation must be run, depending on what information the images contain and on whom they think should receive them. After the first iteration of the transformation, that removes details such as small logos and writings, the document containing the images will thus be labelled with an *imageprotection* level equal to 2, so that only police officers will be able to access it. After the second iteration, that completely deletes faces, the document will be labelled with an *imageprotection* level equal to 1 so that all rescuers will be able to access it. The third iteration will remove instead all details from the scene (e.g. gruesome scenes and plate numbers) and make the document public.

The silver ambulance command centre uses the information on victims' conditions to calculate the number of casualties caused by the incident. On the one hand private content is deleted, while on the other hand the output document's sensitivity is increased for the *confidentiality* tag so that only the silver police command centre can access the information about the number of casualties until its public release. The count transformation (Figure 4.21) is therefore associated with a labelling rule targeting the output document's root with a general declassification label $L^s = \{(P, 0)\}$ that caters for the removal of private content and a classification label

```

<statement date="{fn:current-dateTime()}">
  {</VICTIMS_COUNT count= "{data(fn:doc(VICTIMS_COUNT.xml))}">
  <TEXT>fn:doc("statement.xml")/TEXT</TEXT>}
</statement>
<RULES>
  <RULE glabel="(confidentiality 0)" dlabel="(media true)">
    <XPATH>/statement</XPATH>
    <INPUTDATA file="VICTIMS_COUNT"/>
    <INPUTDATA file="statement"/>
  </RULE>
</RULES>

```

Figure 4.22: Statement update transformation

```

<ASSIGNMENTS date="{fn:current-dateTime()}">
  {for $v in fn:doc("victims.xml")/VICTIMS/VICTIM
  return
    <ASSIGNMENT> {$v,
      for $h in fn:doc("hospitals.xml")/HOSPITALS/HOSPITAL
      where every $d in $v/CONDITIONS/SYMPTOMS/SYMPТОМ/
      SUGGESTEDTHERAPY satisfies
        (some $s in $h/STRUCTURE/DEPARTMENT satisfies
          ($s/@name = $d/@department and $s/@freebeds>0))
      return <HOSPITAL name="{ $h/@name }"/>}
    }
</ASSIGNMENTS>
<RULES>
  <RULE flabel="(confidentiality 1)" glabel="(privacy 0)">
    <XPATH>/ASSIGNMENTS/ASSIGNMENT</XPATH>
    <INPUTDATA file="victims" />
    <INPUTDATA file="hospitals" />
  </RULE>
</RULES>

```

Figure 4.23: Hospital assignment transformation.

$$L^c = \{(C, 3)\}.$$

The transformation to update media statements (Figure 4.22) used by the silver police command centre takes as input the victims count document and inserts it into an official statement for the media. The document is considered sensitive only if the number of casualties is greater than zero, which can be checked using one of the defined CVPs. The transformation is therefore associated with a labelling rule targeting the root of the output document with a general declassification label $L^g = \{(C, 0)\}$ that caters for the removal of the confidentiality protecting the statement and a decisional label $L^t = \{(M, true)\}$ that ensures the number of casualties is checked before deciding the data's label.

Finally, the transformation to assign victims to hospitals (Figure 4.23) used by the silver ambulance command centre generates a document associating victims with hospitals that have free beds in departments suitable to treat their wounds. Victims' information is copied unchanged into the new document and will therefore keep its original labels. The general declassification label for privacy is used to represent the absence of private information in the $\langle HOSPITAL \rangle$ sub-elements of the assignments, while the transformation is also associated with a classification label $L^c = \{(C, 1)\}$ that ensures the output document's $\langle ASSIGNMENT \rangle$ elements are protected so that only rescuers can access them. Figure 4.24 shows a fragment of the output document where

```

<ASSIGNMENTS>
  <ASSIGNMENT label="(confidentiality 1),(privacy 0)">
    <VICTIM status="alive" label="(confidentiality 1),(privacy 1)">
      <NAME label="(confidentiality 1),(privacy 1)">Jane</NAME>
      ....
      <CONDITIONS label="(confidentiality 1),(privacy 2)">
        <SYMPTOMS label="(confidentiality 1),(privacy 2)">
          ....
        </SYMPTOMS>
      </CONDITIONS>
    </VICTIM>
    <HOSPITAL name = "Sacred Heart" label="(confidentiality 1),(privacy 0)">
  </ASSIGNMENT> ....

```

Figure 4.24: The hospital assignment document

the confidentiality level of the `<ASSIGNMENT>` element has been propagated down the tree to `<VICTIM>` and `<HOSPITAL>` elements.

4.8.4 A Comparison with Related Work

Existing ERM systems either ignore the problem of derived data or offer only partial solutions. Most often users need to republish manually the data with new policies of their choice. Atluri and Gal [AG02] provide a definition for *derived authorisations* as authorisations for data derived through a reversible transformation on some original information. They also propose a criterion (but not an actual mechanism) to verify whether a set of derived authorisations is *safe* with respect to the authorisations applied to the original data, i.e. when a user is not authorised to access the original data and cannot derive it back by applying a reversed transformation on the derived data that she can access. However, the authors only propose a simplistic derivation method for authorisations based on the union of the sets of all original authorisations. In [ZJMW05], the authors classify functions to create new data objects on the basis of whether the output might be used to disclose information on the input objects. Depending on whether the function is a *disclosure function* or a *non-disclosure function*, the derived policy (i.e. the set of authorised access requests) is defined either as the intersection or union of the input objects' policies, respectively. The system is the first to directly consider and classify transformations on the basis of how the outputs' and inputs' sensitivities are related. However, the solution completely entrusts policy writers or system administrators with the responsibility to specify correct additional policies for each transformation, which corresponds to directly specifying policies for the transformation's output.

In information flow systems, non-secure information flows are prevented by forbidding processes from receiving data whose label is not dominated by their clearance label. However, in data dissemination scenarios no check over the recipient clearance or rights can be performed before the dissemination as recipients are often not known in advance. Moreover, data senders other than the data originator may have little interest in checking the recipient's clearance before sending the data. Controls need not be introduced at programming language

level either by associating labels with variables and I/O channels, as in the Decentralized Label Model [ML00]. Instead data can be freely disseminated through any available channel since it is previously protected by encryption. An ERM system then guarantees that only authorised recipients will be able to decrypt the data. Our solution is based on the observation that as security labels float from data to processes and from one process to the other, confidential content flows from data to its derivations through the transformations performed. Our solution also allows data declassification. However, while in existing works declassification is performed directly by data owners [ML00, SS05] or at programming language level [LM09], we let declassification depend on the transformations applied to the data. Similarly, we let the amount of sensitive information that flows depend on the transformation used. Finally, while MLS and information flow systems rely on discrete sensitivity levels and the simple security property to control data access, our solution allows the use of expressive policy languages so that we can cater for usage controls that other systems cannot handle.

4.9 Implementation Aspects

All the components in the architecture have been implemented in Java, excluding the daemon interfacing with the trusted computing chip that was implemented in C. The policy language and policy decision point component we used were created in the context of the Consequence project [Con]. The derived data controller component integrated into our proposed architecture uses the standard Java DOM libraries to evaluate XPath expressions. The API allows the specification of custom XPath function resolvers that can be used to call Java functions contained in external libraries. We instead used the Saxon XQuery API to evaluate XQueries. The API allows the specification of different URI resolvers for the query processing so that the in-memory decrypted and pruned XML documents can be passed to the XQuery engine. Encryption has been realised using the standard Java crypto libraries. Communications between the client components and external services such as DSA repositories and certification authorities have been realised with standard TCP/IP connections. The implementation caters natively for cross-organisational boundaries, as the presence of a data sharing agreement guarantees all organisations follow the same policies and decryption keys are attached, together with policies, to the disseminated data.

4.10 Summary and Conclusions

In this chapter we presented the first major contribution of this thesis, i.e. our approach to the problem of derived data protection. We first showed how lattices of policy sets can be built by defining different security domains and specifying for each of them a chain of sensitivity levels associated with policies. We also introduced

content verification procedures as the main means to classify newly created data into a defined policy lattice, and transformations as the only way recipients have in order to modify protected data. Our solution offers a mechanism to control data that is derived through predefined transformations and whose content can only vary according to a predefined scheme. In this sense, the solution limits the number of ways data can be used as it can only be transformed with strictly controlled predefined transformations. Note that “strictly controlled transformations” does not mean simplistic, naive or limited transformations. It rather means that possibly complex transformations are decided at organisational level. The set of defined transformations might thus be large enough to satisfy all possible users’ requirements, depending on the application scenario. Moreover, the set of transformations applicable to a problem domain can be defined ahead of time at organisational level, thus liberating users from the burden of specifying exactly how their actions relate to the security policies.

The implementation of the proposed solution and its integration with the general architecture described in Chapter 3 enables usage control on derived XML documents. The realised mechanism uses the XPath and XQuery languages as means to analyse and modify XML documents, while the derived data controller verifies that data is properly labelled and classified into a policy lattice for the whole duration of its lifetime. Both data confidentiality and integrity can be achieved. The first with selective data encryption, the second by specifying authorisations to run transformations. We emphasise that this implementation is by no means the only possible one, and different mechanisms could be used to specify CVPs and transformations, to resolve policy conflicts and to propagate sensitivity levels and policy decisions in the XML tree. Similarly, different policy languages could be used to build policy lattices, as long as policies can be ordered according to their PSL.

Finally, we evaluated our approach by applying it to an extensive example scenario that was inspired by real data protection practices followed by emergency-relief organisations in the UK. Our qualitative analysis showed that the approach taken is flexible enough to satisfy a large variety of different requirements, and if necessary, also to replicate traditional policy specification. If existing ERM systems had been applied to the scenario provided, one of two outcomes would be possible: 1) no protection would have been applied to sensitive derived documents that are supposed to be accessed only by rescuers or 2) rescuers would be entrusted to directly specify protection requirements for all newly created and gathered data, operation that is not only prone to human errors but would also slow down the rescue operations.

5 Policy-based Authority Evaluation Scheme (PAES)

5.1 Chapter Overview

We have so far described a data protection architecture integrated with trusted computing technologies that allows data to be derived from existing information and protected accordingly. Protection is realised by means of policies associated with the data that specifies what credentials users must present and what conditions these must satisfy for the access to be granted. However, the described solution still presents two of the main shortcomings that are also present in most data protection systems. First, users must trust remote and unknown policy evaluation authorities with their possibly private credentials. In existing ERM systems, all users trust the same centralised authority provided by their organisation. Second, policy evaluation requires users to be connected to a network to contact remote services and access to data is prevented whenever no connectivity is available. In this chapter we present our solution to the aforementioned problems called the Policy-based Authority Evaluation Scheme (PAES).

As we described in previous chapters, the term “policy evaluation” indicates the process through which an entity verifies whether a set of credentials and context tokens satisfies a set of policy constraints. In Chapters 3 and 4 policy evaluation has been used to determine whether data recipients had the right to use received data in specific ways. However, rights to use and access data are not the only type of rights that can be determined with a policy evaluation. For example, the possibility to evaluate policies is a *right* on its own. PAES is a policy evaluation scheme whose design is motivated by the idea that authority to evaluate policies can be granted by other policies associated with them in the same manner as access to data is granted by policies associated with the data. Therefore, in addition to specifying the criteria recipients must meet to access the data, PAES also permits data originators to specify the criteria that authorities must meet to be trusted to evaluate policies. The set of trusted authorities is therefore a dynamic set defined by characterisation (i.e. by the set of constraints defined by a policy) and this confers increased flexibility in finding authorities mutually trusted by both data originators and recipients. This permits the devolution of authority to evaluate policies from a central TA to other parties in a controlled manner. This use of the scheme represents the second contribution

of this thesis. Besides its use in connected environments where recipients can choose a suitable and trusted evaluation authority amongst those that satisfy the criteria defined by data originators, the scheme can also be adapted and used in partially disconnected environments. Consider the crisis management scenario described by *BAE systems plc* [GBLD] as an example of realistic rescue and data exchange practices for public safety organisations in the United Kingdom and presented in Section 1.3. As the context of the crisis evolves, entities moving in the area of the accident can gain new rights such as the right to evaluate policies for other entities, i.e. the right to become trusted policy evaluation authorities. As rescuers meet and communicate by means of *ad hoc* connections, they can exchange decryption keys, evaluate each other with respect to specific policies and grant authorisations, in particular the right to act as policy evaluation authorities or to access data. This use of the scheme represents the third contribution of this thesis.

The remainder of this Chapter is organised as follows. Section 5.2 gives a general overview of our approach while Section 5.3 describes how the data protection architecture described so far can be integrated with PAES. In particular, Section 5.3.2 and 5.3.3 describe the protocol and the data protection mechanism we implemented for connected and disconnected environments, respectively; finally, Section 5.4 describes the mobility model we consider for rescuers in the emergency area and the results of the simulations showing the validity of our approach.

5.2 The Policy-based Authority Evaluation Scheme

We have previously defined a usage control policy P_{act} as the set of requirements recipients must satisfy to perform a given action act on some data and $R_Y^{P_{act}}$ as the set of requests satisfying it. In the data protection architecture described so far, policies are evaluated by evaluation authorities that are trusted by data originators. Rather than specifying a static set of evaluation authorities at publication time, we propose a solution where any entity that satisfies some originator's requirements can act as a policy evaluation authority. Such requirements can again be specified as policies that must be evaluated by other authorities, iteratively. We define an *authority policy* as:

Definition 5.1 *An authority policy $P_{evaluate}$ is a policy describing the requirements that must be satisfied for an entity to be authorised to evaluate other policies.*

In other words, while an evaluation authority can be explicitly chosen if the policy writer directly knows and trusts it, it can also be defined by characterisation if the policy writer only knows which characteristics an entity must present to be trusted. Let us define G_P as the group of entities issuing requests in R_Y^P , i.e. the group of entities that satisfy policy P . For an entity to be part of G_P , an *evaluation authority* trusted by

the data originator must vouch that the conditions specified in P are satisfied. The core idea of PAES lies in the distinction between directly trusted evaluation authorities (i.e. authorities whose identity is well known by the policy writer) and authorities defined by characterisation, i.e. defined by a policy that must in turn be evaluated by other authorities. Traditional ERM systems only use evaluation authorities directly trusted by the data originator, i.e. data recipients are forced to request access to a statically defined set of authorities (usually one). In the following we use DEA_P and EA_P to designate the set of directly trusted evaluation authorities and the whole set of evaluation authorities for policy P (therefore, $DEA_P \subseteq EA_P$).

Each policy is connected with the authority policies specifying which conditions must be met to evaluate it. Therefore, the set of all policies and their connections form a *policy graph* (not to be confused with a policy lattice), which we define more formally below. For example, the sequence of policies:

$$\text{usage control policy } P_0 \leftarrow \text{authority policy } P_1 \leftarrow \text{authority policy } P_2 \leftarrow \text{directly trusted authority}$$

indicates that users who want to access data protected by policy P_0 must request a policy evaluation for P_0 to an entity that satisfies policy P_1 , as evaluated by an entity that satisfies policy P_2 as evaluated by a directly trusted authority. In other words, it represents an authority delegation chain.

Policy writers can specify policy graphs by adding authority policies besides traditional usage control policies, using the same policy language or model. In the following, we will use the terms $in(n)$ to refer to the sets of nodes (i.e. policies) in a policy graph with an outgoing arc towards node n and $out(n)$ to refer to those with an incoming node from node n :

Definition 5.2 *A policy graph is a directed acyclic graph where:*

- *each node n_i for which $out(n_i) \neq \emptyset$ (i.e. a non-leaf node) is a tuple $(P_{evaluate,i}, DEA_i)$ where $P_{evaluate,i}$ is an authority policy specifying conditions to evaluate the (possibly authority) policies P_j contained in all nodes $n_j \in out(n_i)$ and DEA_i is the set of authorities directly trusted to evaluate $P_{evaluate,i}$;*
- *each node n_i for which $in(n_i) = \emptyset$ (i.e. a root node) is a tuple (P_i, DEA_i) where $DEA_i \neq \emptyset$ (i.e. root nodes always have at least one directly trusted evaluation authority);*
- *each node n_j for which $out(n_j) = \emptyset$ (i.e. a leaf node) is a tuple (P_j, DEA_j) where P_j is a usage control policy (i.e. leaf nodes always contain a usage control policy).*

In other words, nodes with no incoming edges represent policies that must be evaluated by directly trusted authorities (i.e. $EA_{P_i} = DEA_i$) and are the root of trust of the rest of the graph. Nodes with no outgoing edges

a directly trusted authority named *entity*₁. Authority policy P_2 can be evaluated by an entity satisfying both authority policy P_4 and P_5 . Authority policy P_5 can be evaluated by an entity satisfying authority policy P_7 that in turn can only be evaluated by a directly trusted authority named *entity*₂. More realistic examples will be shown later.

When applying policy graphs to existing data protection systems, the set of authorities is no longer static but automatically changes when entities no longer satisfy the specified requirements or when new ones that do appear. Recipients (or evaluation authorities) that must be evaluated under a policy can choose their trusted evaluator among those satisfying the corresponding authority policy and those being explicitly listed by the policy writer. They can for example choose the authority they trust the most with their confidential credentials. Moreover, evaluation chains can be built over different paths (i.e. with different authorities). If an authority goes offline or cannot be reached it can be easily replaced by another one satisfying the same requirements. This aspect forms the basis of our solution for data protection in partially disconnected networks.

PAES was directly inspired by trust management systems such as those described in Section 2.4.3. However, when used in the context of data protection systems, these systems have a number of shortcomings. First, their evaluation model relies on a central evaluator to which all the credentials must be presented. Second, the policies used after the first level in the chain are often not decided by the data originator who thus loses control over the delegation sequence. In contrast, PAES entities are only delegated the right to evaluate pre-defined policies and authority over a policy evaluation cannot be further delegated. Also, credentials are only sent from an evaluated entity to its trusted evaluator. Finally, traditional policy evaluation mechanisms can be easily represented by a PAES policy graph where usage control policies can only be evaluated by directly trusted authorities.

5.3 Architecture

In the following we show how the PAES model described in the previous section can be integrated in the high-level architecture we described in Chapter 3 and 4 for data protection in both connected and partially disconnected environments. We emphasise that the description of the **communication manager** and **disconnected operations manager** components we propose abstracts from the specific low-level communication protocols and technologies that can be used in partially disconnected environments. Also note that integration of PAES with the previously described derived data system requires that we modify the data publication and policy evaluation phases described in Chapter 4. To do so, we consider policy graphs whose leaf nodes contain the specification of the set of policies corresponding to specific (tag, level) pairs in a policy lattice. In other words,

the PAES policy graph is used to determine which entities are authorised to evaluate the policies specified in a policy lattice for derived data protection. However, note that PAES and our derived data protection solution are independent of each other and can be used separately.

5.3.1 Data Sharing Agreement

When specifying and signing a DSA, organisations can include information about what evaluation authorities are trusted according to a PAES policy graph. As defined in Chapter 3, a DSA is a 7-tuple $(id, S, A, EA, CA, UP, V)$ where EA represents the set of trusted evaluation authorities. In Chapter 4 we assumed that this field contained the description of a static set of directly trusted authorities that could be chosen when publishing data. We can now instead consider EA to contain the specification of a PAES policy graph. The actual language used in the specification is not important, as long as it allows policies to be linked to each other. We can however use the policy notation described previously, so that an authority policy can be specified as:

$$P_{evaluate}^{ID}(r) = \begin{cases} \text{if } r.act \neq evaluate & \langle NA, \emptyset \rangle \\ \text{else if } r \models (targetPolicy = id_1 \vee \dots \vee targetPolicy = id_n) \wedge (deny\ conditions..) & \langle N, \emptyset \rangle \\ \text{else if } r \models (targetPolicy = id_1 \vee \dots \vee targetPolicy = id_n) \wedge (allow\ conditions..) & \langle Y, \emptyset \rangle \\ \text{otherwise} & \langle NA, \emptyset \rangle \end{cases}$$

where, differently from the policies described in Chapter 4, each policy is associated with a unique ID used to refer to it as the target of another authority policy. For the sake of simplicity, we will not include obligations for authority policies. The authority policy specifying who can evaluate a set of usage control policies corresponding to a specific (tag,level) pair in a policy lattice can be specified as:

$$P_{evaluate}^{ID}(r) = \begin{cases} \text{if } r.act \neq evaluate & \langle NA, \emptyset \rangle \\ \text{else if } r \models tag = t \wedge level = l \wedge (deny\ conditions..) & \langle N, \emptyset \rangle \\ \text{else if } r \models tag = t \wedge level = l \wedge (allow\ conditions..) & \langle Y, \emptyset \rangle \\ \text{otherwise} & \langle NA, \emptyset \rangle \end{cases}$$

where “tag” and “level” are keywords for the tags and levels in a policy lattice. For the sake of clarity, we remind that a (tag, level) pair is associated with a set of policies with *at most* one policy for each action. In the example of Figure 4.3, the (privacy,1) pair is associated with the policies $P_{write}^{p,1}$ and $P_{read}^{p,1}$. In other words, the (tag, level) pair is used as an ID for policy sets contained in a policy lattice. A disjunction of such conditions can be used to specify more than one policy set in the same authority policy. In other words, the devolution of

authority described by a policy graph is applied per (tag, level) pair. For the sake of simplicity, we will assume all (tag, level) pairs to be contained into only one node in the graph, so that evaluation of the corresponding policies is controlled by only one authority policy. Directly trusted evaluation authorities can be specified with equality conditions on their identity in the authority policy specification. In the following, given a leaf node n_i in a policy graph, we will refer to P_i as the set of (tag, level) pairs associated it and from which policies can be reconstructed.

The specification of several authority policies like the two shown above actually corresponds to the specification of a PAES policy graph. Generally, the creation of a DSA is no easy task, especially when organisations with different requirements decide to share data and when people's lives may depend on the agreed policies. Creating both a policy graph and a policy lattice is complex as it may prove difficult to foresee all the situations where the data access may be required and the manner policies combine and propagate once several data objects have been merged and transformed. This is why a policy analysis tool that could determine whether the specified policies satisfy specific criteria would be an important tool for the organisations writing the DSA. Significant work has been done on policy analysis including within our own group [CLM⁺09]. However, the integration of such techniques with a data dissemination framework remains for future work.

5.3.2 Connected Environments

Data Publication in Connected Environments

Whenever connectivity is available, the data publication phase follows the same steps described in Section 4.7.2. However, the licence is generated differently, as described in the following.

Consider a scenario where a fictional Medical Research Centre (MRC) receives data about patients from the Sacred Heart Hospital (SHH) to run studies on some medical products its employees are currently working on. Figure 5.2 shows a possible SHH corporate policy graph governing access to patient records. SHH lets each patient verify the policy lattice (whose details are not shown) and choose a set of entities she trusts to evaluate all the corresponding policies. The patient considered in the example chooses Bob, one of her closest friends, to determine whether recipients satisfy the policies in the lattice. If Bob is a journalist or competitor most researchers would not want him to know what they are working on and will not send their credentials to him. However, the policy graph also allows researchers to be evaluated by any research facility working on stem cells research ranked by USNews as one of the best hospitals. Evaluation of whether a research facility satisfies this policy is left to SHH. SHH and the research facility may however be competitors, and the facility may not trust SHH to know what kind of research projects it is carrying out. Therefore the policy also allows research facilities to be evaluated by any publicly recognised organisation giving grants for medical research. Whether

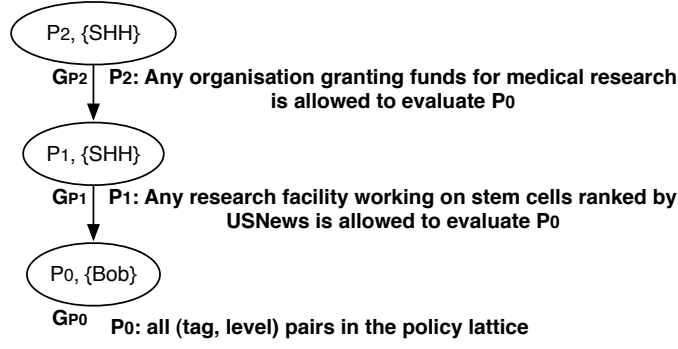


Figure 5.2: Example policy graph for a connected scenario

Algorithm 3: Symmetric key generation for encryption of XML data

Input: A set of data labels $DL = \{L_1 \dots L_n\}$ associated with an XML document's elements

forall distinct (tag,level) pairs (t,l) in labels in DL **do**

 | generate random symmetric key $k_{t,l}$;

end

forall label $L_i \in DL$ **do**

$k_i = 0$;

forall (tag,level) pair $(t,l) \in L_i$ **do**

 | $k_i = k_i \oplus k_{t,l}$;

end

end

a company or organisation satisfies this requirement is left again to be verified by SHH.

The protected patient's data package must therefore contain information related to the specified policy graph. To do so, we modify the mechanism to generate the symmetric keys used to encrypt the XML document described in Section 4.7.2. Instead of generating a symmetric key for each distinct data classification label associated with the document's elements, we generate a symmetric key $k_{t,l}$ for each (tag, level) pair (t,l) in the policy lattice. The symmetric key used to encrypt an element associated with a specific data label is generated as the xor of all keys $k_{t,l}$ for which the (tag, level) pair (t,l) appears in the label. The process is described by Algorithm 3.

All elements associated with the same data label L_j can now be encrypted with the same key k_j . Once the PAP has received all policy updates (including PAES graphs), the derived data manager has labelled the data and the PEP has encrypted it with the generated keys, the PEP also generates n symmetric keys k_{P_i} (one for each authority policy P_i , or non-leaf node n_i in the graph). The keys are used to generate a publishing licence that is a concatenation of message parts $m_0 \dots m_n$ generated according to Algorithm 4. The algorithm takes in input a policy graph, the set of keys $k_{t,l}$ associated with (tag, level) pairs and a symmetric key k_m used to encrypt the metadata. Intuitively, each message m_i is associated with a node n_i in the graph and contains (possibly part

Algorithm 4: Publishing licence generation

Input: A policy graph, the list of (tag,level) pairs and corresponding keys $(t,l) : k_{t,l}$ and a key k_m

forall nodes $n_i = (P_i, DEA_i)$ **do**

if $out(n_i) \neq \emptyset$ **then**

 generate random symmetric key k_{P_i} ;

end

end

forall nodes $n_i = (P_i, DEA_i)$ **do**

 generate message $m_i = P_i$;

if $out(n_i) = \emptyset$ **then**

$info_i = k_m | list\ of\ (tag, level)\ pairs \in P_i\ and\ corresponding\ keys\ k_{t,l}$;

end

else if $out(n_i) \neq \emptyset$ **then**

$info_i = H(P_i), k_{P_i}$;

end

forall $ea \in DEA_i$ **do**

$m_i = m_i | \{info_i\}_{PK_{ea}} | ID_{ea}$;

end

end

forall node n_i **do**

if Node Type(n_i) $\neq$ "strict" **then**

forall node $n_j \in in(n_i)$ **do**

$m_i = m_i | \{info_i\}_{k_{P_j}}$;

end

end

else if Node Type(n_i) = "strict" **then**

 generate random symmetric key k'_{P_i} ;

$m_i = m_i | \{info_i\}_{k'_{P_i}}$;

$num = \#in(n_i)$;

 /* break the key in num fragments

*/

$\{frag_{i,1} \dots frag_{i,num}\} = break(k'_{P_i}, num)$;

$l=1$;

forall node $n_j \in in(n_i)$ **do**

$m_i = m_i | \{frag_{i,l}\}_{k_{P_j}}$;

$l++$;

end

end

end

of) the information to unlock the information associated with subsequent nodes in the policy graph. Therefore, each message contains the policy of node n_i (or set of (tag, level) pairs for leaf nodes), the set of directly trusted evaluation authorities that can be contacted (with their identity certificates ID_{ea}) and the decryption key (or fragment of decryption key) to decrypt the subsequent node's information (or the data for leaf nodes). Keys associated with strict nodes are fragmented using a simple xor function so that a key fragment is generated for each parent node. For leaf nodes, the policy to be satisfied is the list of (tag, level) pairs associated with the node. Messages for leaf nodes also contain the key k_m used to encrypt the metadata that must be used by

evaluation authorities to evaluate the usage control policies and the list of keys $k_{t,l}$ associated with the (tag, level) pairs. The message parts for a labelled document published with the example policy graph of Figure 5.2 would thus look like:

$$\begin{aligned}
m_0 & P_0 \text{ (list of all (t,l) in the leaf node)} \mid \{k_m, \text{list of all (t,l) : } k_{t,l}\}_{k_{P_1}} \mid \\
& \{k_m, \text{list of all (t,l) : } k_{t,l}\}_{PK_{Bob}} \mid ID_{Bob} \\
m_1 & P_1 \text{ (Stem cells facility with USnews rank } \leq 10) \mid \{H(P_1), k_{P_1}\}_{k_{P_2}} \mid \\
& \{H(P_1), k_{P_1}\}_{PK_{SHH}} \mid ID_{SHH} \\
m_2 & P_2 \text{ (Recognised org. granting funds)} \mid \{H(P_2), k_{P_2}\}_{PK_{SHH}} \mid ID_{SHH}
\end{aligned}$$

Figure 5.3 depicts the new protected data package containing the message parts as a publishing licence. Each key k_{P_i} is used to protect the keys corresponding to subsequent nodes in the graph. A copy of each symmetric key k_{P_i} is also protected with the public keys of the directly trusted evaluation authorities for P_i . Therefore, to obtain key k_{P_i} an entity belonging to group G_{P_i} must be authorised by an authority $ea \in G_{P_j}$ where $n_i \in out(n_j)$ or by a directly trusted authority for P_i , i.e. by entities that can decrypt key k_{P_i} . If a node is a strict evaluation node, a new key k'_{P_i} is generated to protect k_{P_i} . The new key is then fragmented and each fragment is protected with the key k_{P_j} corresponding to a node $n_j \in in(n_i)$. Therefore, to obtain key k_{P_i} an entity belonging to group G_{P_i} must be authorised by an entity that is in all authority groups G_{P_j} for each node n_j in the set $in(n_i)$ (i.e. an entity that can obtain all fragments of k'_{P_i} and reconstruct it). As for the data package described in Section 4.7.2, the packages integrity is also protected. An attacker trying to modify one of the messages m_i could either modify the unprotected policy, the encrypted parts of the message or the directly trusted authorities' identity certificates. In the first and second case, the attack would be detected by an authority when accessing the hashes (or list of (tag, level) pairs and corresponding keys in message m_0) in the message's encrypted part. In the third case, the package would become useless as only the originally intended authorities can decrypt the rest of the package. In other words, modification of any part of the package would result in the impossibility of accessing the data, but not in an incorrect policy evaluation.

Policy Evaluation and Enforcement in Connected Environments

The policy evaluation process for a PAES policy graph can be described as a recursion of the process described in Chapter 3. It comprises an initial forward process (*policy evaluation*) and a final backward process (*key disclosure*). The policy evaluation process finds a chain of entities satisfying the policies in the policy graph. The initiator is the data recipient e that receives the data with the messages $m_0, m_1 \dots m_n$ and tries to access it. Let P_0 be a leaf node containing the set of (tag,level) pairs contained in the data label associated with the data e is trying to access. The data recipient first looks for an entity $ea \in DEA_{P_0}$ she trusts. If the search

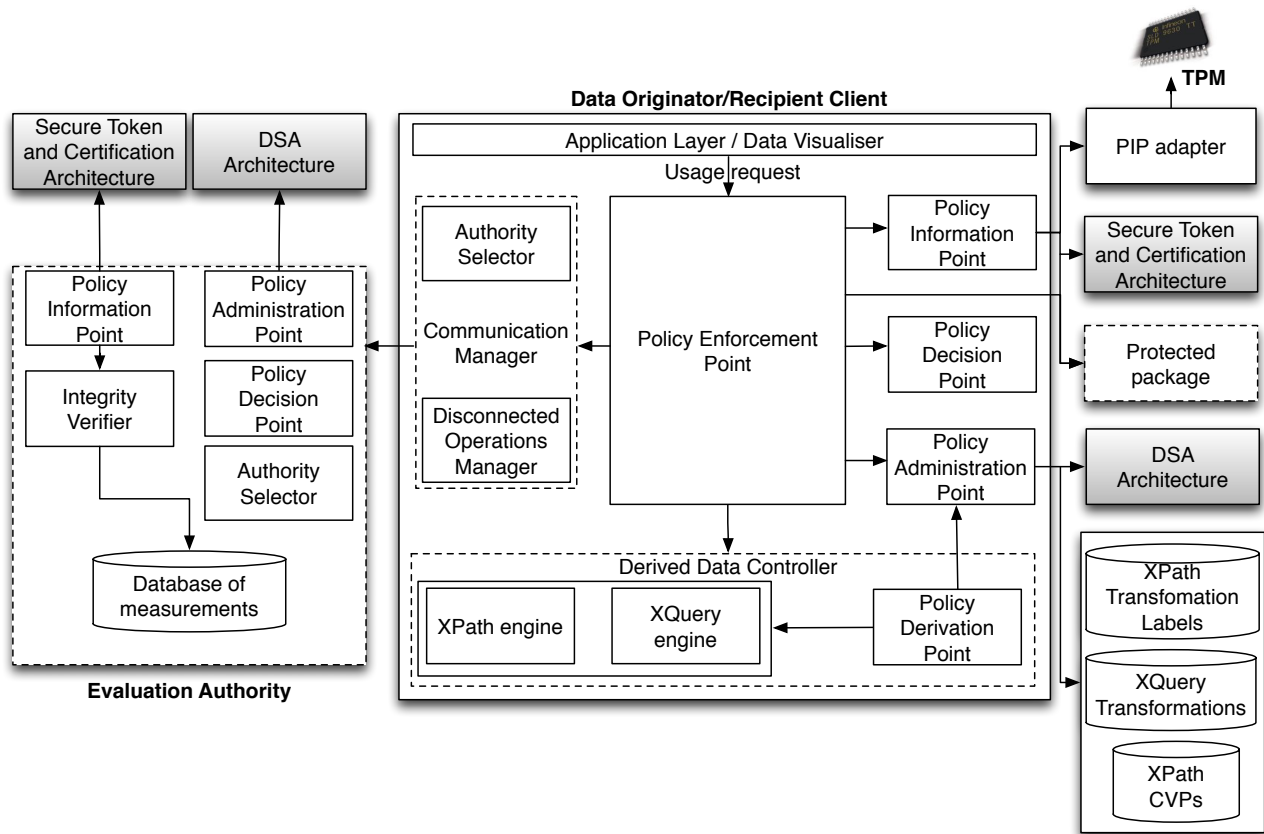


Figure 5.4: High level architecture integrated with PAES for use in connected environments

Figure 5.4 depicts the updated version of the proposed architecture. The **authority selector** component is used by the PEP to select trusted entities. This is realised with a simple predefined list to compare with the entities contained in the *DEA* sets. If no entity in the set is trusted, we let the user select an entity from the predefined list. Other more complex approaches are indeed possible. Note that evaluation authorities must also be provided with an authority selector as they also must be evaluated. Figure 5.5 shows the interaction of components for policy evaluation. The figure does not show the whole evaluation process since after a suitable evaluation authority has been selected, the process continues as depicted in Figure 3.7.

Figure 5.6 shows the policy evaluation process in the example described at the beginning of the previous section. The recipient of a patient's record sends the message parts, along with her credentials (certifying that she is a biochemist working for MRC and specialised in stem cells) to a research facility. Since authority policies are public, she should look for a facility running a project on stem cells and ranked by USNews, as specified in the requirements. We call this entity e_0 . e_0 verifies that the recipient is a MRC biochemist specialised in stem cells, removes message m_0 and sends the data, along with its credentials, to an organisation that funds research projects. We call this entity e_1 . e_1 verifies that e_0 is a facility working on stem cells and ranked by USNews, removes message m_1 and sends the data, along with its credentials, to SHH (e_2). SHH

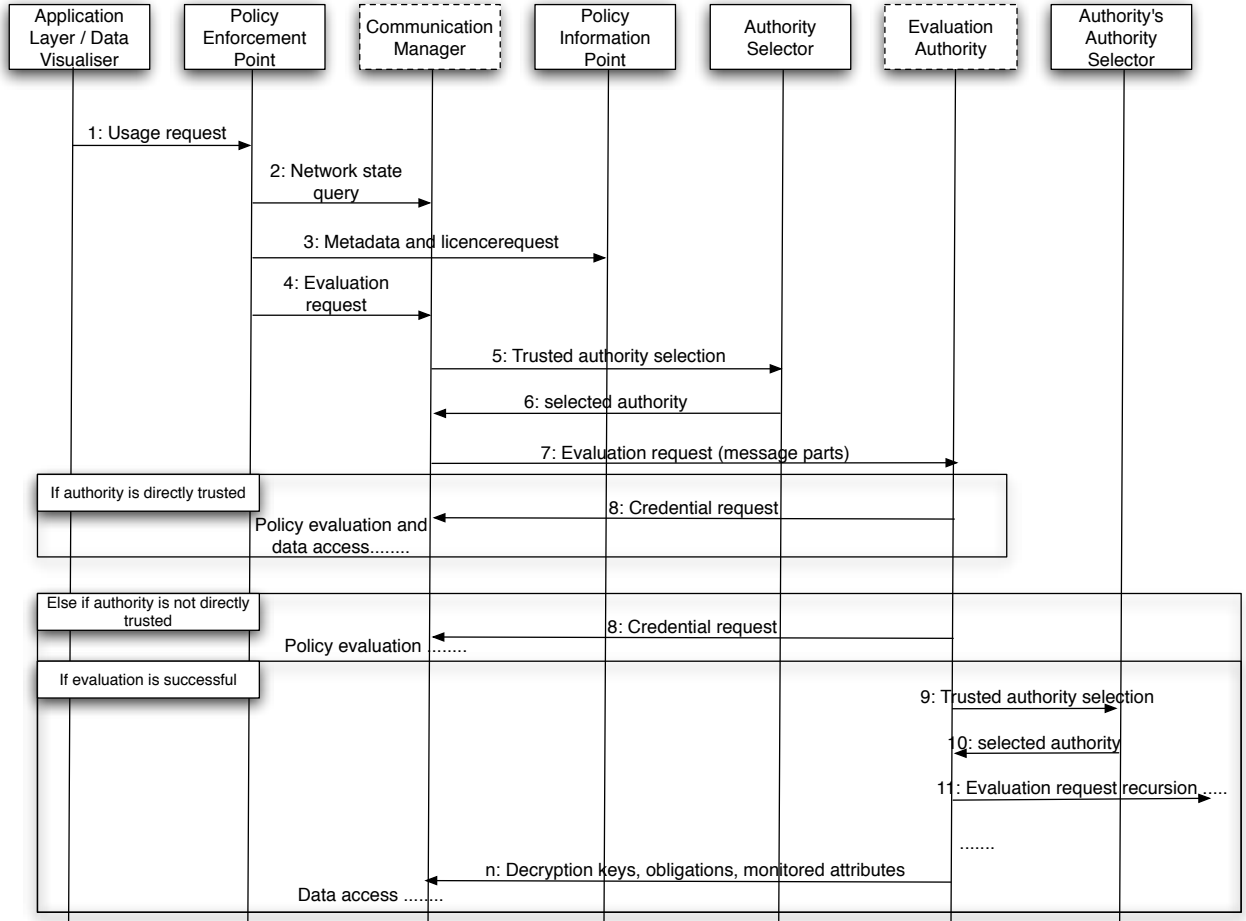


Figure 5.5: Interaction of components for policy evaluation using PAES in connected environments

can directly access $h(P_2)$ and k_2 and verify that e_1 is a company whose research grants are publicly recognised. This concludes the forward process.

During the backward process, i.e. when keys are disclosed, e_2 sends k_2 to e_1 , who can now access and verify the integrity of P_1 . If the policy was not tampered with, it can send k_1 back to e_0 , who can now access the (tag, level) pairs list with the corresponding keys and the metadata. It can evaluate the policies corresponding to the (tag,level) pairs and send the keys back to the data recipient who may now access the document. Note that the integrity of the metadata and the (tag, level) list is guaranteed by the fact that both are encrypted like the keys encrypting the document. Maliciously altering any part of the publishing license would result in the impossibility for the authorities to decrypt the metadata, encryption keys or document.

Each entity trusts the one at the next level to see its credentials and to verify correctly whether it satisfies the requirements specified in the policy. However, trust is not transitive since it is strictly related to the evaluation of a policy which involves only two nodes. Credentials sent from an entity to an authority to be evaluated are in fact not further disseminated and confidential information is accessed only by authorities trusted to do

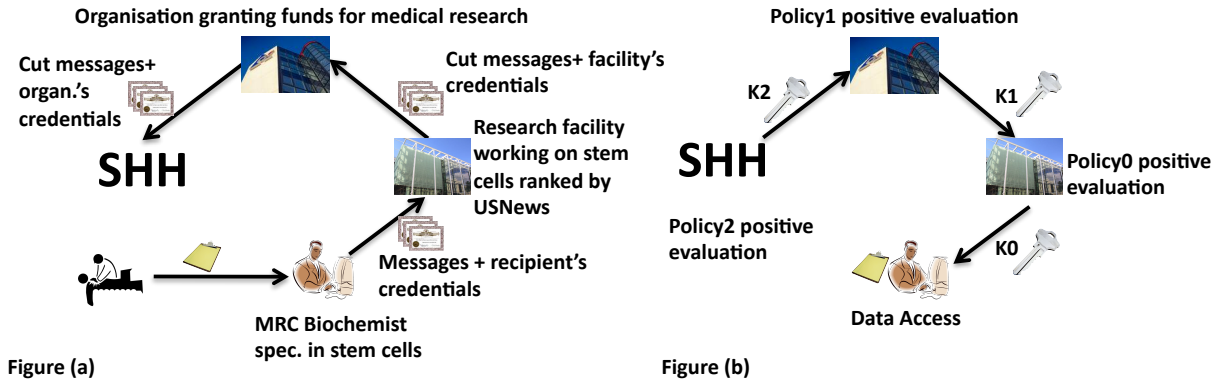


Figure 5.6: Policy evaluation recursion

so. Finally, since at each step a part of the data package is removed, entities at each level can only access the information they have been authorised for. With respect to trust management systems, recipients no longer need to gather credentials from every entity involved in the chain. At each step two different entities (evaluator and evaluated entity) perform a part of the protocol and evaluate part of the policy chain. At the end, the data recipient only has to present her own credentials to the chosen authority and does not worry about the other entities being part of the chain.

5.3.3 Partially Disconnected Environments

Let us now consider the crisis management scenario described in Section 1.3. When network connectivity is absent, the movements of first responders at the scene of the accident can be exploited to carry the information to whoever needs it. The multi-hop *ad hoc* network paradigm [CG07, RHB⁺07] was introduced to address situations where no network connectivity is available. Intuitively, peers exploit the temporary *ad hoc* connections they establish with each other when moving to deliver messages. Vahdat *et al.* [VB00] introduced the concept of *epidemic routing*, i.e. the broadcast of messages to all nearby peers to guarantee their final delivery to an intended recipient. Lilien *et al.* [LGY07] proposed the *opportunistic networks* (oppnets) paradigm, where a small set of initial peers (*seeds*) iteratively ask others to participate in the network and become *helper nodes*. By including new peers the oppnet grows, thus increasing the likelihood that messages are delivered. Li *et al.* [LR00] proposed an algorithm to modify peers' movement trajectories to increase message delivery. The *Haggle* project [SHCD06, SSH⁺07] integrates the various approaches by introducing an architectural layer that automatically selects the best connection mode available when applications require network access.

Oppnets are ideal for first responders that need to exchange information in the area of an incident and several proposals have applied *ad hoc* networking to crisis management. Aschenbruck *et al.* [AGPG⁺07, AFMT04]

present a mobility model for rescuers based on the established procedures that public agencies follow when facing a crisis. The simulation results show that *ad hoc* networks benefit from an higher density of the peers and their constrained mobility. Lilien *et al.* [LKB⁺06] highlight the general security requirements for *ad hoc* networks in terms of message privacy and integrity and of common attacks that can hinder the network operation (e.g. denial of service). However, the authors suggest that because of the impossibility of authenticating peers when they enter the network, it is necessary to rely on intrusion detection or trust management systems to discover malicious nodes only after they have already misbehaved. Seth *et al.* [SK05] propose the use of hierarchical identity-based encryption (HIBE) to encrypt messages disseminated in an *ad hoc* network. However, this solution is suitable only when the identity of the intended recipient is known in advance and when all the participants on the network know the central private key generator (PKG). Moreover, as the scale of the incident grows larger, several agencies are called to intervene and share information. Although DSAs between the agencies would have been typically set up, it is not possible for them to agree about which responders will be authorised to access which data (and pre-emptively give them access keys created from access hierarchies or using attribute-based encryption), as crisis situations often evolve in sudden and unpredictable ways that do not conform to the rehearsed scenarios. The context in which the events unfold must thus be considered at each access request. Agencies with no pre-existing DSA and whose rescuers have not been given any pre-defined key may also be called upon to intervene. In general, data protection schemes for *ad hoc* networks are mostly intended for protecting communications between a specific sender-recipient pair, i.e. to protect data packets while they are stored and forwarded by intermediate nodes. This conflicts with the need to disseminate information widely and in a timely manner. Information must be shared among several responders and sending separate copies of the same data to each intended recipient is not a viable solution in such a low bandwidth network. Similarly, the centralised architecture of traditional rights management solutions makes ERM approaches unusable in opportunistic and *ad hoc* networks.

Opportunistic Network Overview

Let us assume that responders are equipped with mobile devices such as PDAs, smartphones and netbooks capable of running an instance of our data protection architecture and capable of short-range wireless communication so that they can establish temporary *ad hoc* connections when they are close to each other. The devices can also store small amounts of information and can deal with encryption keys and algorithms. Modern smartphones with a Java virtual machine and equipped with an SD card or other memory are typical examples of the devices we consider. Let us also assume that responders manually activate a *disconnected operations mode* on their devices, so that they can actively search for nearby peers to communicate with. This is mainly

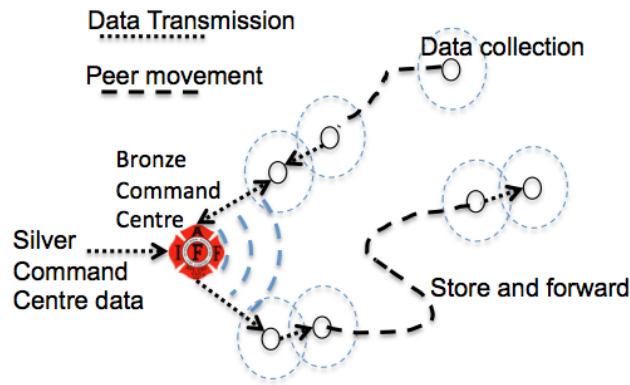


Figure 5.7: Data exchange in a crisis scenario

done to save battery when connectivity is available. Alternatively, the disconnected operations mode could be automatically turned on when no network is detected. However, this solution would again result in a waste of battery in the case no connectivity was available but there were no need for data exchange. The emergency opportunistic network established in our scenario (Figure 5.7) comprises three types of peers: rescuers (the same depicted in Figure 4.15), bronze command centres (BCC) and *roots of authority*.

Rescuers move in the disaster area, gather information and store and forward messages while moving. Note that they may carry messages that they cannot themselves access, which are protected as described in the following sections. BCCs are usually located on emergency vehicles on the perimeter of the disaster area. Although it is not a strict requirement, we consider BCCs to have special connection capabilities that allow them to send/receive information to/from silver command centres (SCC) located in the organisations' headquarters. They thus receive information both from rescuers and SCCs. As described in Chapter 4, both BCCs and SCCs aggregate information received from both outside and inside the disaster area and re-disseminate derived information possibly of limited size. Roots of authorities are the peers entrusted by the respective organisations to start a sequence of policy evaluations, i.e. the entities placed as directly trusted evaluation authorities in the PAES policy graph. They keep the access and usage control policies and deploy them in the crisis area. Each individual rescuer is assumed to store in her device a copy of the identity (public key) certificate of the roots of authority and of the BCCs belonging to her agency and of others well-known trusted third parties (e.g. certification authorities). In our scenario, we consider BCCs to work as both roots of authorities and data aggregators.

Using PAES in Opportunistic Networks

The difficulty in applying traditional ERM protection schemes to oppnets is the lack of reachable trusted evaluation authorities (EA) that can be used for data publication and policy evaluation. When publishing data that

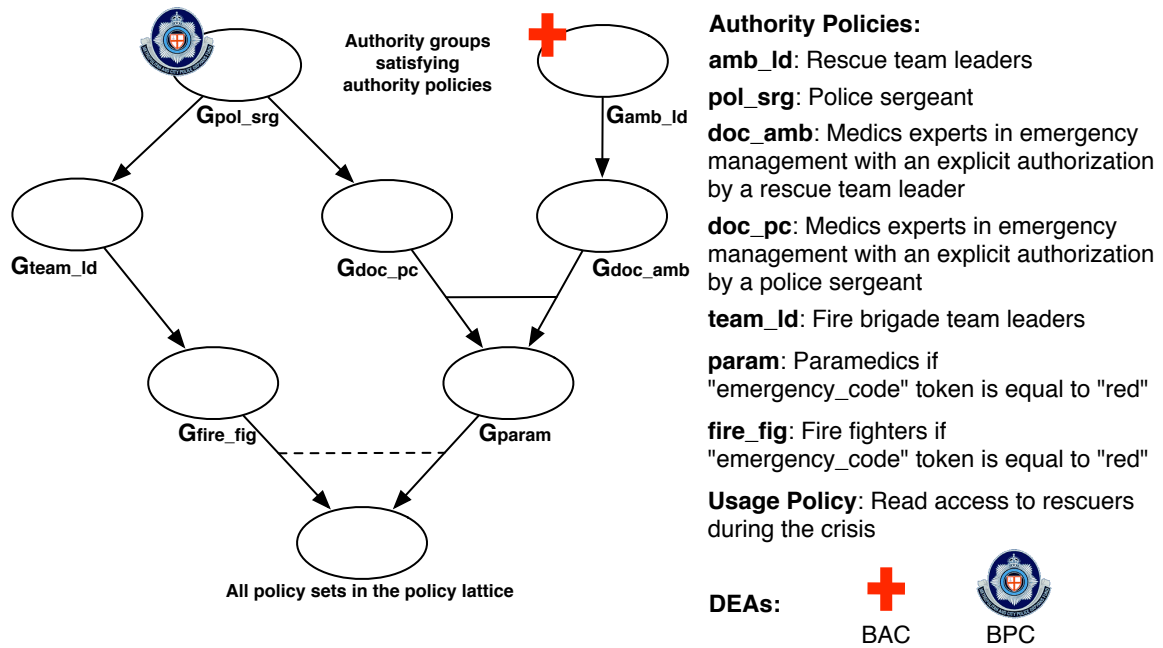


Figure 5.8: An example PAES hierarchy for crisis management

will be disseminated in an environment with no connectivity, the only entities that data recipients will be able to contact are those that will move in the disconnected area. In other words, the only entities that will be able to evaluate whether a rescuer can access some data or not are other rescuers moving in the area and that the data recipient can meet and interact with.

Rescue organisations typically have pre-defined intervention strategies that their personnel follow during crisis management. Similarly, pre-defined strategies for policy evaluation can be specified as PAES graphs. Figure 5.8 shows an example PAES graph for crisis management. Consider the scenario where a police BCC receives from the company owning one of the tankers involved in the accident a short document with details about a liquid leaking into a sewer. The information is immediately disseminated in the area via opportunistic forwarding and with an attached label whose confidentiality level states that all rescuers can access it while the emergency operations are still ongoing. The DSA stipulated by the police, fire brigade and ambulance services specifies that any data recipient requesting to access the data must be authorised at least by a fire fighter (satisfying the *fire_fig* policy) or a paramedic (satisfying the *param* policy), provided the current emergency-code level described by a security token disseminated by the police is “red”. In other words, if the situation is severe enough that data must be released more quickly, fire fighters and paramedics can act as evaluation authorities trusted by data originators (or, more specifically, by their organisations). They are located close to potential recipients and are thus in the best position to evaluate their access to data. However, before they can do so, their

role and the value of the current emergency code must be verified. Fire brigade team leaders (satisfying the *team_ld* policy) and expert medics (satisfying both the *doc_pc* and *doc_amb* policies) are considered authorities to evaluate these conditions, as they are the most suitable individuals for evaluating their subordinates. However, even team leaders and medics must be evaluated to verify that they can be trusted for policy evaluation, i.e. that they satisfy the specified criteria. Since the organisation originating the data is the police, it is necessary that police identifies an authority to act as a “bridge” with the fire brigade and ambulance services. In other words, an entity that can pass authority to members of the two partner organisations during the emergency. It is thus decided that team leaders must be evaluated by police sergeants (satisfying the *pol_srg* policy). Since medics must take care of much more delicate tasks, it is also decided that doctors should receive an explicit authorisation by both an ambulance services’ rescue team leader (satisfying the *amb_ld* policy) and a police sergeant. These new authorities can be directly evaluated by the respective BCCs that act as roots of authority. To maximise the probability of meeting a trusted evaluation authority, recipients can also contact authorities at higher levels than their direct parents in the policy graph. For example, when the current emergency code is not equal to “red” and paramedics and fire fighters cannot act as authorities, recipients (including fire fighters and paramedics) can still contact fire brigade team leaders and medics to access the data.

The solution we propose to allow policy evaluation in *ad hoc* networks is to evaluate policies preemptively and authorise (even unknown) peers when they are met in the crisis area, possibly before they receive any data. In our example, rescuers may have no contact with the police BCC. However, the police BCC gives to police sergeants the right to evaluate fire brigade team leaders. Both sergeants and team leaders are likely to move in the area and meet more peers. They can then evaluate them and give them either access rights or evaluation rights, according to the defined policy graph. Police sergeants may meet team leaders and give them evaluation rights over fire fighters. Team leaders are then likely to move and meet fire fighters, to whom they can assign the corresponding evaluation rights. Finally, fire fighters may meet other rescuers and give them access rights. Evaluation or access rights are therefore passed from an authority to a peer whenever they meet for the first time, opportunistically.

One of the most important aspects of crisis management is the timeliness of the information. Delays in accessing data caused by the absence of connectivity with an evaluation authority may cause severe problems during the operations. The preemptive policy evaluation we propose is therefore a suitable solution.

Authority Specification and Deployment

As root of authorities, BCCs initiate the dissemination of authority rights in the area of the accident in the form of disseminated group keys. First, they generate an *authority table* on the base of the PAES policy graph

ID	Policy (Y conditions)	EAGs	DEAs	Node Type	Keydata
fire_fig	role="fire fighter" $\wedge$ emergency_code="red"	team_ld			APK_{fire_fig}
param	role="paramedic" $\wedge$ emergency_code="red"	doc_pc doc_amb		strict	APK_{param}
team_ld	role="team leader"	pol_srg			APK_{team_ld}
doc_pc	role="doctor" $\wedge$ expertise="emergency" $\wedge$ explicitAuthorisation="true"	pol_srg			APK_{doc_pc}
doc_amb	role="doctor" $\wedge$ expertise="emergency" $\wedge$ explicitAuthorisation="true"	amb_ld			APK_{doc_amb}
pol_srg	role="sergeant"		BPC		APK_{pol_srg}
amb_ld	role="rescue team leader"		BAC		APK_{amb_ld}
policy lattice		fire_fig param			

Table 5.1: Example authority table

described in the DSA. Table 5.1 shows an authority table for our example. Each row associates the identifier of a group of authorities (in the ID column, corresponding to the ID of an authority policy) with the authority policy entities must satisfy to be assigned to it (in the **Policy** column), and with the authority group(s) authorised to evaluate whether the policy is satisfied (in the **evaluation authority groups, EAGs** column). For simplicity, the policy specification in the example table contains only the positive conditions (contrary to the policies described in Chapter 4 which also contain a negative and a *not applicable* set of conditions), but a full specification is also possible. The **DEAs** column is instead used to specify directly trusted authorities. Their ID in this case must correspond to the ID contained in their identity certificates that must be stored in the devices. The **Node Type** column specifies whether the policies are contained either in a loose-evaluation node or in a strict-evaluation node. Intuitively, each row represents a node of the policy graph. As for traditional policy evaluations, once a peer becomes a member of an authority group, it is trusted to verify the satisfaction of the policy conditions (that must be monitored) periodically. Thus, while a condition for a group G is temporarily not satisfied, the peer is trusted not to use the corresponding authority, i.e. it does not evaluate peers for groups G_i for which G is an evaluation authority group. The last line in the table indicates which authorities can evaluate usage control policies associated with (tag, level) pairs.

Each authority group G is also associated with a pair of public and private keys (APK_G and APK_G^{-1}) contained in the **Keydata** column. Possession of a group private key is synonymous with membership in that group. Keys are periodically refreshed by roots of authority before any new crisis event, to avoid reuse of keys distributed during previous events that may have been compromised. Initially, rescuers do not possess any private keys as their membership in a specific authority group must be assigned by a trusted evaluation authority. No one is part of any group *a priori*, as keys are distributed during the crisis. This is done for two main reasons: first,

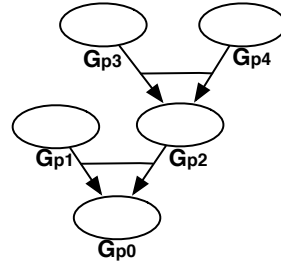


Figure 5.9: Policy graph with multiple strict evaluation nodes

context conditions must be considered when evaluating policies; second, rescuers belonging to organisations who are not part of a DSA might also need to access data and may thus receive the keys during the crisis if they satisfy the corresponding policies.

Group private and public keys are initially generated and kept only by roots of authority in special data structures described in the next section. While public keys are freely distributed when peers meet, distribution of private keys is subject to policy evaluation.

Key Generation

The process generating the authority private key pairs is a function of the policy graph defined in a DSA. It can be run either by roots of authority or higher-level entities managing them (e.g. SCCs). For each row in the authority table a new key pair is generated and the corresponding public key is stored in the **Keydata** field. Consider an authority ea and a peer e being evaluated by ea to be assigned to group G_P . If the evaluation is positive, ea must be able to give $APK_{G_P}^{-1}$ to e . Therefore, every evaluation authority must be given the group private keys of all the authority groups it can evaluate. However, if policy P is part of a strict evaluation node, then ea must be a member of different authority groups to evaluate P , e.g. G_{P_1} and G_{P_2} . Therefore ea should be able to obtain $APK_{G_P}^{-1}$ and give it to e only if authorised by two possibly *different* authorities. This is why for strict evaluation nodes private keys are split in several shares with an xor function and given to different evaluation authorities specified in the table. The use of the xor function for secure key fragmentation is well-known in cryptography [key03] and assures that no knowledge about the original key can be gained from knowing only a subset of its fragments. However, not only $APK_{G_P}^{-1}$ must be fragmented, but also all the keys or fragments of keys of groups for which G_P is an evaluation authority group.

Consider the example depicted in figure 5.9. To be authorised to evaluate P_0 (i.e. to have $APK_{G_{P_0}}^{-1}$), a peer must satisfy P_1 and P_2 . Satisfying each policy should provide a key fragment of $APK_{G_{P_0}}^{-1}$ ($frag_1$ and $frag_2$ respectively) to the peer. To be authorised to evaluate P_2 and provide $frag_2$, a peer must satisfy P_3 and P_4 . Also $frag_2$ should then be fragmented, so that each evaluation provides a fragment of $frag_2$, i.e. $frag_3$ and

$frag_4$ respectively. Key $APK_{G_{P_0}}^{-1}$ should thus be actually split in three fragments $frag_1$, $frag_3$ and $frag_4$. We assume peers do not collude to obtain fragments they do not possess to reconstruct whole keys and impersonate authorities they are not authorised for. We also assume they do not evaluate strict-evaluation policies before being members of all the groups they need (e.g. a member of G_{P_1} does not give $frag_1$ if not also possessing $frag_2$). These assumption are reasonable since each fragment is issued only to peers that satisfy specific trust requirements (i.e. policies). Private keys and their fragments are stored in special *key data structures* of the form $(keyFragment, evalChain)$, where *keyFragment* is a key or fragment of key and *evalChain* is the sequence of authorities that can have the key and pass it to others after a positive policy evaluation. Keeping information on such sequences is necessary as authorities store keys for their direct children, but also for all their descendants. Algorithm 5 describes the recursive procedure to generate the key pairs and key data structures initially stored by roots of authority. $keyset_{ID}$ is a set of key data structures belonging to the authority ID . Note that despite the similarities with Algorithm 4, the two algorithms produce very different results. Algorithm 5 generates in fact a set of *keyset* data structures and requires a recursive procedure to fragment keys according to the policy graph. Algorithm 4 instead generates a set of messages that constitute a license ready to be disseminated with the data and does not require any recursion since fragments of keys must not be further fragmented. The result obtained by Algorithm 5 for the policy graph of our crisis example (Figure 5.8) would then be:

$$\begin{aligned} keyset_{pol_srg} = \{ & (KP_{pol_srg}^{-1}, pol_srg), \\ & (KP_{fire_fig}^{-1}, fire_fig/team_ld/pol_srg), \\ & (KP_{team_ld}^{-1}, team_ld/pol_srg), \\ & (KP_{doc_pc}^{-1}, doc_pc/pol_srg), \\ & (frag_{param,1}, param/doc_pc/pol_srg) \} \end{aligned}$$

$$\begin{aligned} keyset_{amb_ld} = \{ & (KP_{amb_ld}^{-1}, amb_ld), \\ & (KP_{doc_amb}^{-1}, doc_amb/amb_ld), \\ & (frag_{param,2}, param/doc_amb/amb_ld) \} \end{aligned}$$

Note that only the key sets for root nodes are generated, since any other information would be redundant. Keys and key fragments belonging to each group can be easily recognised as the group's ID is included at the

first position in the evaluation chain.

Algorithm 5: Authority key generation procedure

Input: An authority table

forall TAG elements ea **do**

 generate random key pair $(APK_{ea}, APK_{ea}^{-1})$;
 AddKey($ea, (APK_{ea}^{-1}, ea)$);

end

Procedure AddKey:

Input: An authority ea , a data structure $KD = (frag, chain)$

if $EAGs(ea) = \emptyset$ **then**

$keyset_{ea} = keyset_{ea} \cup \{KD\}$;

end

else if Node Type(ea) $\neq$ "strict" **then**

forall $a \in EAGs(ea)$ **do**

 AddKey($a, (frag, chain/a)$);

end

end

else if Node Type(ea) = "strict" **then**

$num = \#EAGs(ea)$;

 /* break the fragment in num new fragments

*/

$\{frag_1 \dots frag_{num}\} = break(frag, num)$;

$i = 1$;

forall $a \in EAGs(ea)$ **do**

 AddKey($a, (frag_i, chain/a)$);

$i++$;

end

end

Authority Evaluation and Key Distribution

When a crisis situation occurs, the agencies involved must first of all ensure that responders (including those from other agencies) are aware of their data protection policies, especially if a common DSA is not available. We must consider this case when dealing with crisis management scenarios where different agencies intervene on the scene and must cooperate even with nor previous agreement. As we will describe later, this implies that rescuers must exchange all information on the DSA they operate under, so that the corresponding policies can be enforced also by users of other organisations that did not agree on it. When operative in the area, roots of authority (in our example BCCs) first try to locate nearby directly trusted authorities (in our example BCCs from different agencies) and contact them. If communication is possible (as we generally assume for BCCs), they exchange the respective authority tables and policy lattices. Even if direct communication is not possible, officials of the different agencies would typically have a face to face meeting (e.g. a briefing) to organise the operations. Data could then be exchanged. Thus, initially all BCCs are considered to know the policy graphs and policy lattices of all the initially involved agencies. If organisations share the same DSA, the DSA specifies

who must generate the authority table and key sets to be distributed. Otherwise, different policy lattices from different DSAs can be easily merged into one lattice including all tags, CVPs and transformations. However, policy graphs cannot be merged so easily. Consider two policy lattices lat_a and lat_b whose evaluation authorities are specified by two different policy graphs pg_a and pg_b and let them be merged into a policy lattice lat_c . If a new data object satisfies CVPs from both lat_a and lat_b it will be labelled with non-default sensitivity levels for both lattices. Access to the data will thus be subject to the evaluation of several policies coming from different lattices, where each policy requires a different evaluation authority. In other words, users would be forced to require access from an authority defined by pg_a and an authority defined by pg_b at the same time, which would slow down the data access process. This is why having a common DSA for all participating agencies, or different DSAs whose policy lattices do not overlap (i.e. the set of data objects they protect do not overlap), is the best situation possible. Once all BCCs are aware of all available policies, the corresponding authority tables (with no private keys) can be disseminated to all responders.

At this point the roots of authority begin sending to each nearby directly trusted authority the key sets for the groups they are authorised to evaluate. In our case, police, ambulance services and fire brigade share a DSA and are thus aware of the common policy graph and policy lattice. Let us assume that they agree on the bronze police command centre (BPC) generating the corresponding key sets. The BPC inserts $keyset_{pol_srg}$ in the *keydata* column of its authority table at row pol_srg . Since the BAC is a directly trusted authority for the group amb_ld , the BPC sends to it $keyset_{amb_ld}$ (via the same channel used to exchange the table).

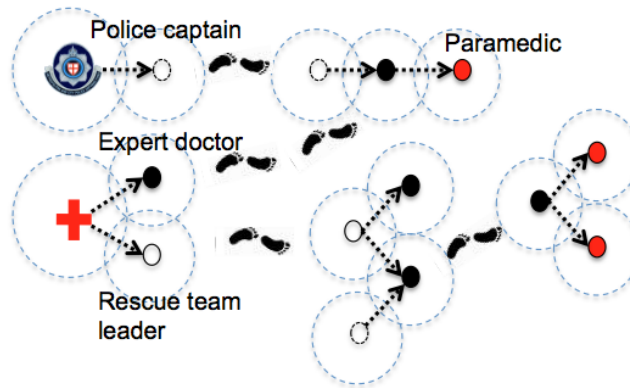


Figure 5.10: Opportunistic dissemination of authority keys

Starting from directly trusted authorities, group membership is opportunistically evaluated as follows (figure 5.10). Whenever two peers e_i and e_j come into communication range, unless they have previously met during the current crisis and have not received new keys in the meantime, they exchange the list of authorities (and of (tag, level) pairs) they are authorised to evaluate (according to the authority groups they are currently members

of). Amongst these, each peer chooses the authority groups of which they want to become members. The choice can be made either by the human user or automatically, according to a pre-defined strategy. The two peers exchange the list of chosen authorities along with the certificates needed for the evaluation of the corresponding policies. Each peer evaluates the policies with the received certificates and available context parameters. For each satisfied policy, i.e. for each new group the evaluated peer must become member of, a new key set is created containing all the key data structures available whose evaluation chain contains that group. The key sets are then exchanged and used to update the authority table. Key fragments are also merged whenever possible to obtain a new fragment or a whole key. Consider a police sergeant member of group *pol_srg* meeting a fire brigade team leader. After evaluating the policy for group *team_ld*, the police sergeant's device will generate $keyset_{team_ld} = \{(KP_{fire_fig}^{-1}, fire_fig / team_ld / pol_srg), (KP_{team_ld}^{-1}, team_ld / pol_srg)\}$ collecting the key data structures containing *team_ld* in their chains. The fire brigade team leader will then receive the key set and use it to fill the row *team_ld* in its authority table. With this mechanism, authority keys are distributed in the crisis area only according to the requirements specified in the policy graph.

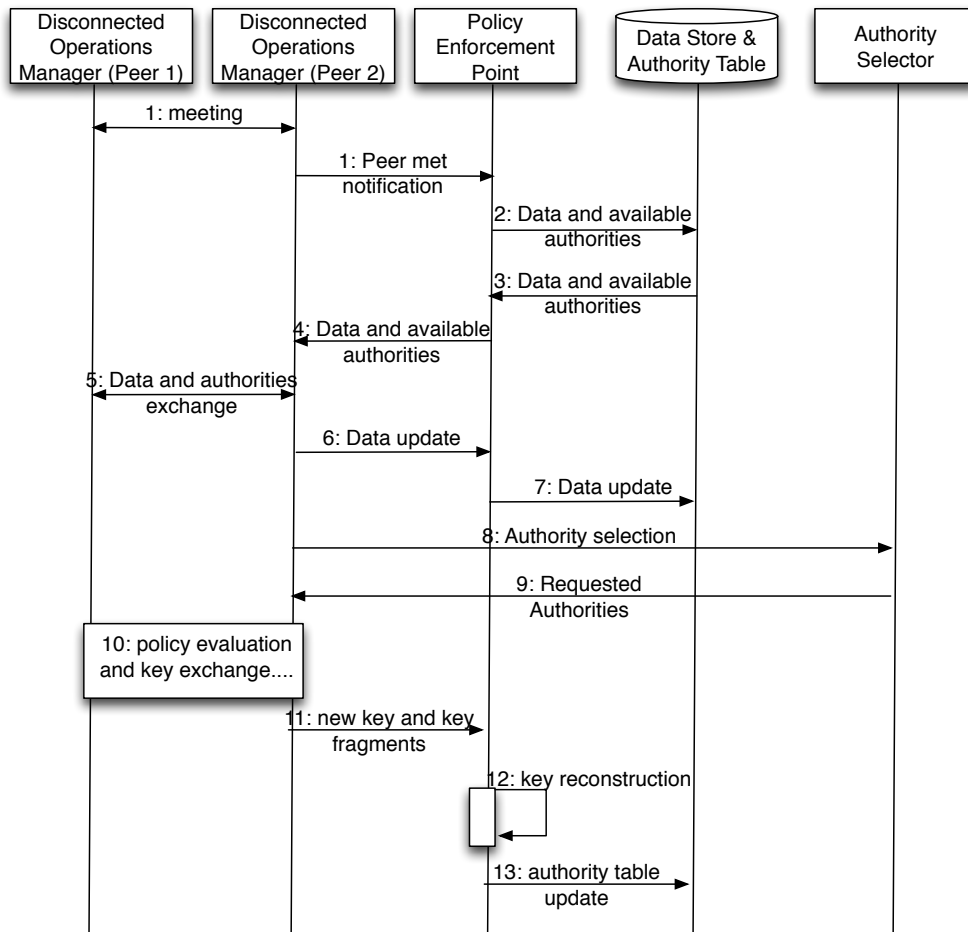


Figure 5.11: Interaction of components for policy evaluation in disconnected environments

Figure 5.11 shows the interaction of components for policy evaluation. The figure does not show the details of the policy evaluation that is performed as already described in Sections 3.4.3 and 4.7.3. The main difference with the protocols described in previous chapters is that peers do not contact neither the DSA Architecture for policy updates nor the secure token and certification architecture to get credentials. Such services are in fact unreachable when connectivity is not available. Also, before evaluating each other, peers exchange all available data (including all known public keys and policies), according to the opportunistic dissemination paradigm. This is all the more important since disseminated data can contain context tokens and revocation lists to be used in policy evaluation such as the emergency code token generated by a silver command centre. Organisations may even assign to well-known rescuers deployed in the crisis area the right/duty to make decisions upon specific context values (e.g. again, the emergency level). Other rescuers could then update their local knowledge with the most recent received information. Also, revocation can be partially realised by BCCs performing re-keying operations at fixed time intervals or whenever important context information changes. This results in the dissemination of a new authority table and of new key sets together with the updated context information. Note that re-keying does not correspond to a revocation of the already obtained access rights, but would only prevent recipients from obtaining new data without a new evaluation. However, re-keying is the first recovery procedure to enact when something goes wrong, i.e. when unauthorised users get access to confidential data. The fact that peers act as evaluation authorities that cannot contact external services explains why we did not use specialised attestation authorities when evaluating trusted platform modules as described in Chapter 3. Clients must store locally the needed integrity measurements to use when evaluating peers. Similarly, peers are expected to store public keys of certification authorities to verify the validity of credentials. The mutual evaluation between peers is possible because the client architecture is already provided with all components (PAP, PDP, PIP) it needs to evaluate policies, exactly as traditional remote evaluation authorities. Figure 5.12 depicts the updated and final version of the proposed architecture, where remote evaluation authorities are substituted by clients. Note that this is true also for connected environments, where evaluation authorities can have the exact same component structure as clients.

Data Publication in Partially Disconnected Environments

Data publication follows the same process described in Section 4.7.2. However, since connectivity is not available, peers do not contact the DSA architecture for policy updates, and the publishing licence is built differently. Once keys for each (tag,level) pair have been generated as described by Algorithm 3 and the data has been labelled and encrypted as described in Section 5.3.2, the data package is built appending the encrypted data to the information generated by algorithm 6. The metadata is encrypted with a symmetric key k_m . Then,

the encryption keys to send to authorised rescuers. Also, any alteration of the package would result in the impossibility for any rescuer to decrypt the metadata, the labels and decryption keys or the whole document.

Algorithm 6: Publishing licence generation for disconnected publication

Input: A policy graph, the list of (tag,level) pairs and corresponding keys $(t,l) : k_{t,l}$ used for the data and a key k_m

```

package =  $\{metadata\}_{k_m}$ ;
forall nodes  $n_i = (P_i, DEA_i)$  do
  if  $out(n_i) = \emptyset$  then
    generate symmetric key  $k_{P_i}$ ;
    package = package |  $\{list\ of\ all\ (t,l) : k_{t,l} \in P_i\}_{k_{P_i}}$ ;
    forall  $ea \in DEA_i$  do
      | package = package |  $\{k_{P_i}\}_{PK_{ea}} | ID_{ea}$ ;
    end
    if Node Type( $n_i$ )  $\neq$  "strict" then
      forall node  $n_j \in in(n_i)$  do
        | package = package |  $\{k_{P_i}\}_{APK_{P_j}}$ ;
      end
    end
    else if Node Type( $n_i$ ) = "strict" then
      num =  $\#in(n_i)$ ;
      /* break the key in num fragments */
       $\{frag_{i,1} \dots frag_{i,num}\} = break(k_{P_i}, num)$ ;
      l=1;
      forall node  $n_j \in in(n_i)$  do
        | package = package |  $\{frag_{i,l}\}_{APK_{P_j}}$ ;
        | l++;
      end
    end
  end
end

```

Data Access in Partially Disconnected Environments

Once evaluation authorities have been distributed in the disconnected area, evaluation of usage control policies can be performed as for connected environments, provided that the recipient meets an evaluation authority after she receives the data. Evaluation of usage control policies (i.e. policies associated with (tag, level) pairs) can be performed only with respect to a specific data object, since the corresponding encryption keys $k_{t,l}$ are only stored in the data object's licence and are unique for every published data object. When meeting (i.e. when getting into communication range with each other), if an authority claims it can evaluate the sets of policies corresponding to a certain set of (tag, level) pairs contained in a license, then the user can ask it to decrypt the corresponding keys. The authority can in fact use the private key of its authority group to access the key encrypting the keys $k_{t,l}$ associated with the (tag,level) pairs. Obtained keys are then stored so that they can

be combined with keys $k_{t,l}$ from other authorities, in case they were not sufficient to build the whole key corresponding to a whole classification label. Indeed, if wireless remote connectivity becomes available, the recipient can also contact a remote authority. Note that the absence of a nearby authority to contact may still prevent authorised rescuers from accessing the data. However, in our example rescuers such as paramedics and fire fighters will always be able to access it (provided the context conditions are verified) since they themselves are considered trusted authorities for the evaluation of the usage control policies. As such, it is preferable to specify policy graphs such that all the possible intended recipients are included in at least an authority group. In this way recipients/authorities can directly access the (tag,level) pairs and $k_{t,l}$ keys to decrypt (possibly part of) the data and to evaluate locally the corresponding usage control policies. Also, if the (tag,level) pairs contained in a leaf node were not sufficient to generate any label, data recipients may need to acquire more than one authority, i.e. to acquire access to the encrypted information of more than one leaf node. For example, consider a policy graph specifying that only firefighters were authorised to evaluate policies corresponding to any level of the confidentiality tag and that only paramedics were authorised to evaluate policies corresponding to any level of the privacy tag. This policy graph would have two leaf nodes, one containing all the (c,i) pairs and one containing all (p,j) pairs, whose parent nodes would be the firefighter and paramedic authority groups, respectively. A recipient should be (or contact an authority that is) both a firefighter and a paramedic at the same time to access information labelled as $\{(c, 1) (p, 1)\}$. This situation is clearly unlikely. This is why it is important to balance the tradeoff between granularity of control and performance and why in our example all (tag,level) pairs are contained in the same leaf node in the policy graph.

An important aspect to consider is what happens when users regain connectivity. The *disconnected operations manager* would detect the newly available network and data would no longer be published according to the opportunistic PAES protocol. However, the system must guarantee that already disseminated data is still accessible and at the same time leverage the new dissemination context. This is why it is advisable that in the PAES policy graph each node (or at least the leaf nodes containing the usage control policies) is associated with at least one directly trusted evaluation authority that is also a traditional, remote evaluation authority that users can contact when connectivity is available. In this way the passage from disconnected to connected networks can occur seamlessly. In the opposite case (i.e. the passage from connected to disconnected networks) data would instead need to be republished under the new conditions.

5.4 Evaluation

When used in connected environments to determine trusted evaluation authorities, the main limitation of PAES is scalability. When one of the policies is evaluated negatively, the entity under evaluation must choose a different evaluator or return back to the previous entity a negative response. This also happens if no trusted evaluators can be found. Therefore, a policy chain evaluation may return a negative result even if the entity under evaluation actually satisfies the specified requirements. The flexibility that PAES offers to receive a negative response and try a different evaluation authority raises a scalability issue. If n entities are required to evaluate a policy at each level of a chain spanning l levels, in the worst case the number of messages exchanged would be $O(n^l)$. However, we must consider that: i) an evaluation chain can be shorter than the corresponding policy chain (e.g. if the chosen evaluation authority is directly trusted by the policy writer), ii) policy recipients can filter out the entities they do not trust or that do not probably satisfy the chain requirements, iii) policy chains for most application scenarios are very short.

In any case, the major application PAES was intended for is its use in disconnected environments. In this context, the PAES-driven key distribution creates an overhead on data distribution and data access. Responders acting in the crisis area may receive useful data that they cannot access because no valid authority has evaluated them yet. It is therefore important to verify that such an overhead does not create undue burden and does not significantly hinder rescue operations. We have simulated the movements and communications of rescuers in the crisis scenario described in section 1.3 using the GUS (Geographic Urban Simulator) [BD09]. We performed tests that considered the dissemination of data originally sent from the BPC, using the example policy in Figure 5.8 and considering paramedics and firefighters as the intended recipients. The aim of the simulation was to evaluate the rates at which keys and data are distributed, i.e. to verify that delay between the delivery of the data and the delivery of the key is negligible. The simulation has been performed over only one (although randomised) mobility model, but a pertinent, realistic and pessimistic one, thus guaranteeing the results are general.

5.4.1 Mobility Model

The simulation area is divided into five different sub-areas:

- The accident area, i.e. the 100m radius perimeter centred on the car crash location;
- The evacuation area, i.e. the area external to the incident perimeter and limited by a 400m radius perimeter with the same centre. This is the area used by rescuers to evacuate victims and to move towards the respective BCCs;

- The bronze police command centre area, the bronze fire brigade command centre area and the bronze ambulance command centre area, placed respectively north-west, north-east and south on the evacuation area perimeter.

Again, we based our simulation and thus the simulated behaviour of rescuers on the study of the actual procedures followed by public safety organisations in the United Kingdom [GBLD]. Rescuers move according to the following patterns, partially depicted in Figure 5.13. Fire fighters move in teams of three, including a team leader. They first move from the BFC area towards the accident area. We simulate the search for victims, fires or collapsing structures inside the area by having groups of rescuers move to several destinations in the centre of the emergency area. For each destination reached they wait 120 seconds and then keep moving to a new one. Constables follow the same pattern as fire fighters. Police sergeants move back and forth from the BPC area to a random point in the accident area, to simulate a control-and-report activity. Paramedics and doctors follow the same pattern to simulate rescue and evacuation of victims. All the rescuers move at 1.1 m/s, which corresponds to the speed of a fast walking person, and broadcast messages every second. This mobility model is actually the most conservative choice for peer mobility as it represents a worst-case scenario for the performance of our protocol. Obstacles, forced routes (e.g. roads) and attraction points in the area constrain in fact the rescuers to move on the same paths, increasing the number of times they meet and thus the performance of the protocol. Our simulation is thus based on a situation that is more demanding with respect to the protocol than a real world setting.

The simulation was executed multiple times, each time varying parameters, including the number of rescuers involved and the communication range of their devices. In particular, we considered 30, 75 and 150 rescuers (evenly divided between the three participating organisations) with 10m, 20m and 30m communication ranges. Rescuers were divided as follows in their respective organisations: police sergeants represent 20% of the police force; doctors and rescue team leaders represent 9% and 20% of the ambulance services force respectively; fire fighters team leaders represent 33% of the fire brigade force.

5.4.2 Results and Evaluation

To measure the effectiveness of the PAES-driven key distribution, we compare data delivery with the key delivery time for all rescuers in the system. In the opportunistic networking context, message delivery time refers to the elapsed time taken for a message to be received by all intended peers within the system. The delivery ratio represents the total fraction of rescuers who have successfully received a message at a specific instance in time. In contrast, key delivery time measures how quickly peers receive authority group keys they are entitled for based on the policy graph available.

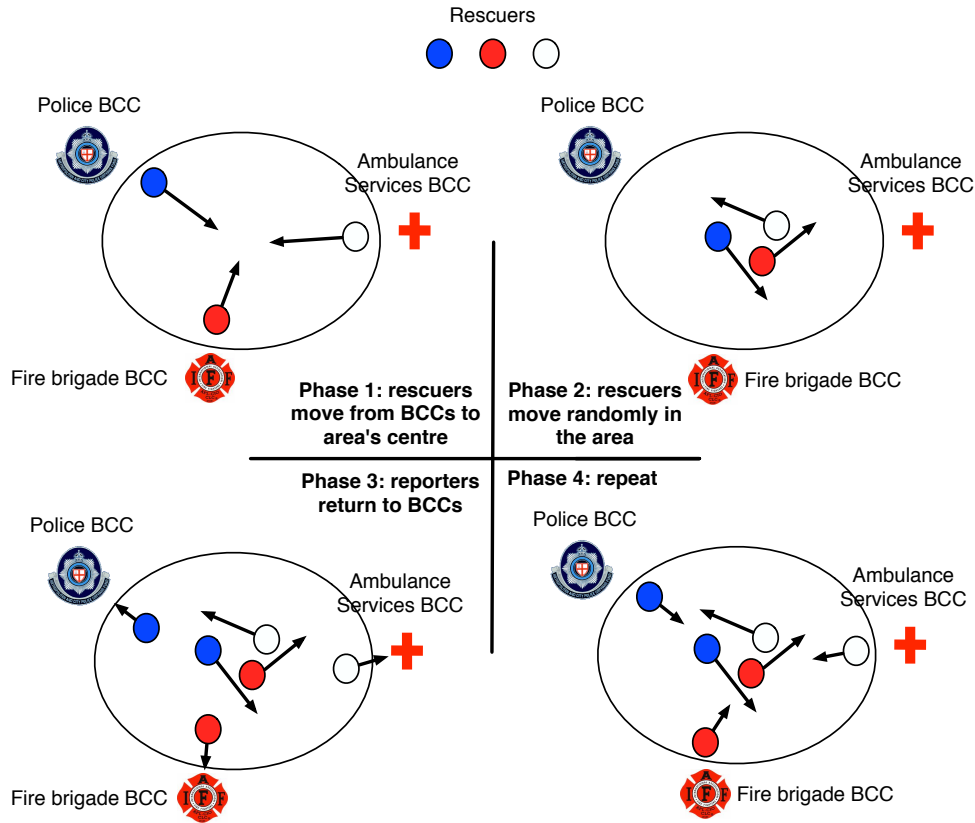


Figure 5.13: Mobility model of rescuers in the emergency area

Rescuers	10m	20m	30m
30	1386	979	887
75	1040	943	883
150	869	804	756

Table 5.2: Average key delivery times (seconds)

Rescuers	10m	20m	30m
30	1333	891	759
75	1197	1117	900
150	1031	864	760

Table 5.3: Average data delivery times (seconds)

Tables 5.2 and 5.3 show the average key delivery times and average data delivery times (in seconds) obtained varying the devices' communication range and number of rescuers. Figures 5.14, 5.15 and 5.16 depict the results when varying the communication range of 30 rescuers. Figures 5.17, 5.18 and 5.19 depict the results when varying the communication range of 75 rescuers. Figures 5.20, 5.21 and 5.22 depict the results when varying the communication range of 150 rescuers. All results show a sigmoid function representing key and data

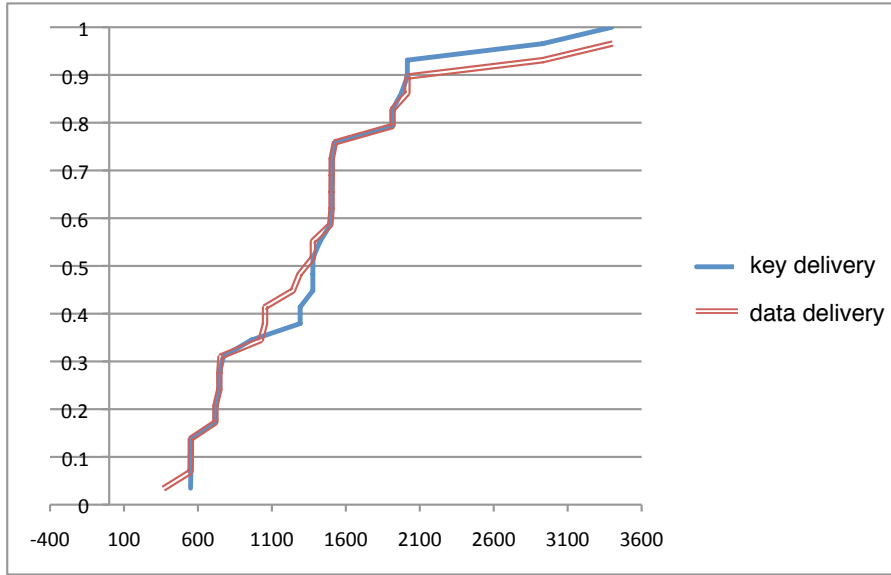


Figure 5.14: Key delivery and data delivery ratio (30 rescuers, 10m communication range)

delivery times similar to that found in closed systems. If we refer to the elapsed experiment time as t , then we see a sigmoid shaped delivery function where the function tends towards 1.0 for increasing t ($\lim_{t \rightarrow \infty} f(t) = 1.0$). Increasing the communication range was seen to improve the derivatives of both data delivery and key delivery, as it does increasing the peer population, as expected. We emphasise that both sigmoid functions plotting data delivery times and key delivery times are close to each other in all experiments. As expected, data delivery starts prior to key delivery (that is subject to positive policy evaluation). However, while for simulations involving 30 rescuers the data delivery function overlaps or overcomes the key delivery function for most of the simulated time, for simulations involving a larger amount of rescuers key delivery quickly overcomes data delivery.

Only in scenarios with 30 rescuers is data delivery faster than key delivery for most of the simulations, as confirmed by the average results plotted in Figure 5.23. The results indicate that keys can be received before the data, so the overhead posed by the protocol is negligible. We can explain such results as follows. As the rate of increase in performance fundamentally depends on the multiplicity of broadcast messages to neighbouring rescuers (i.e. the number of peers and the communication range), the rate of data delivery and the rate of key delivery are affected by mobility and by the complexity of the policy graph (see later). Policies are evaluated starting from all BCC areas (i.e. wherever directly trusted authorities are) and thus keys are distributed from three different locations (and by three different organisations) towards the centre of the crisis area. Also, having rescuers from the same organisation moving in teams eases the evaluation process, as keys are directly passed from team leaders to their subordinates. Therefore, performance only depends on a single step in the dissemination process, namely the passage of keys from a rescuer belonging to an organisation to a

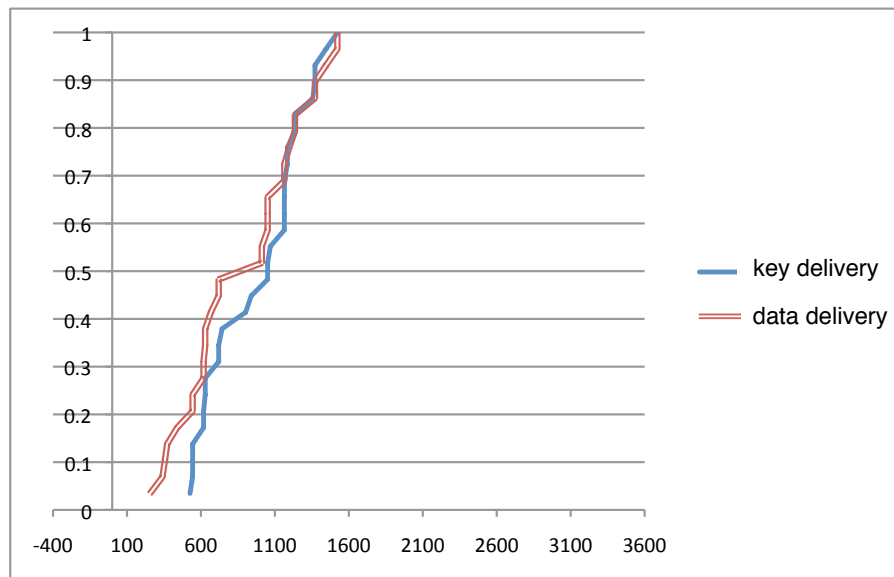


Figure 5.15: Key delivery and data delivery ratio (30 rescuers, 20m communication range)

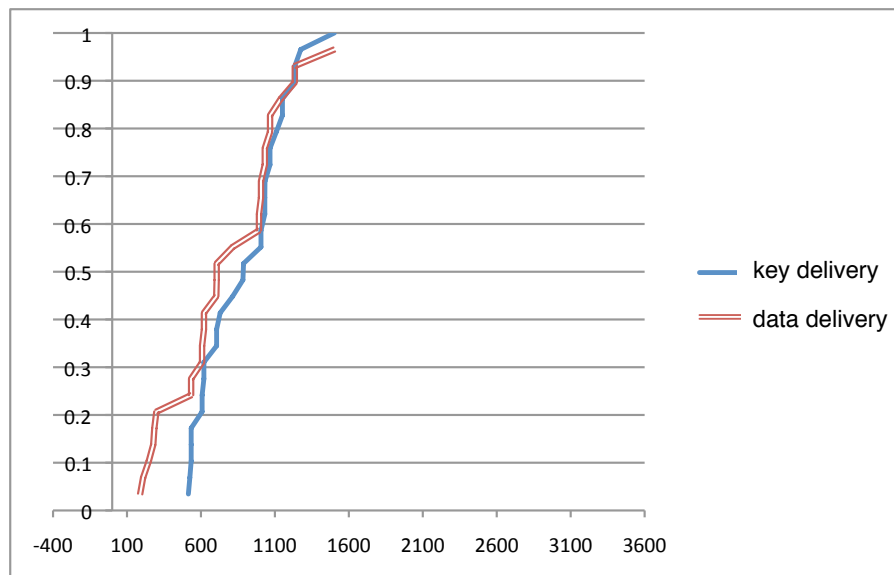


Figure 5.16: Key delivery and data delivery ratio (30 rescuers, 30m communication range)

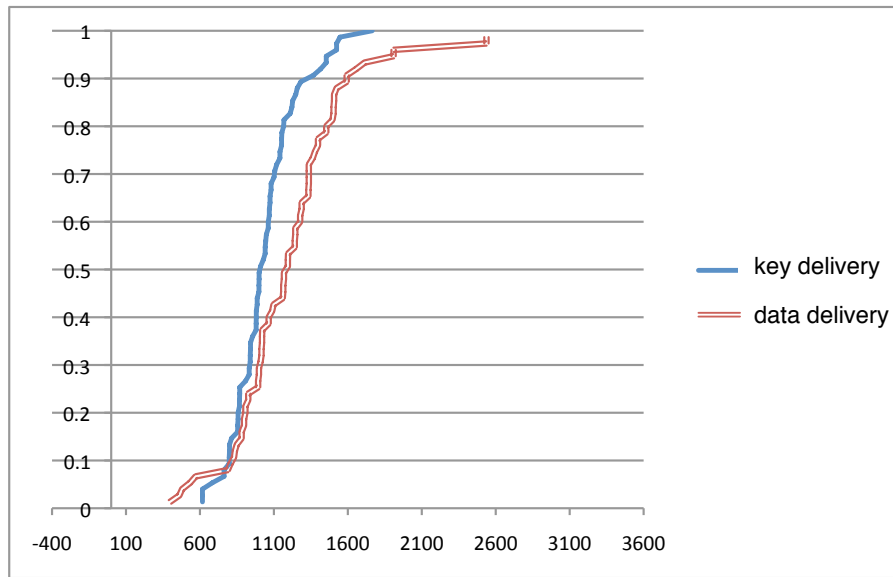


Figure 5.17: Key delivery and data delivery ratio (75 rescuers, 10m communication range)

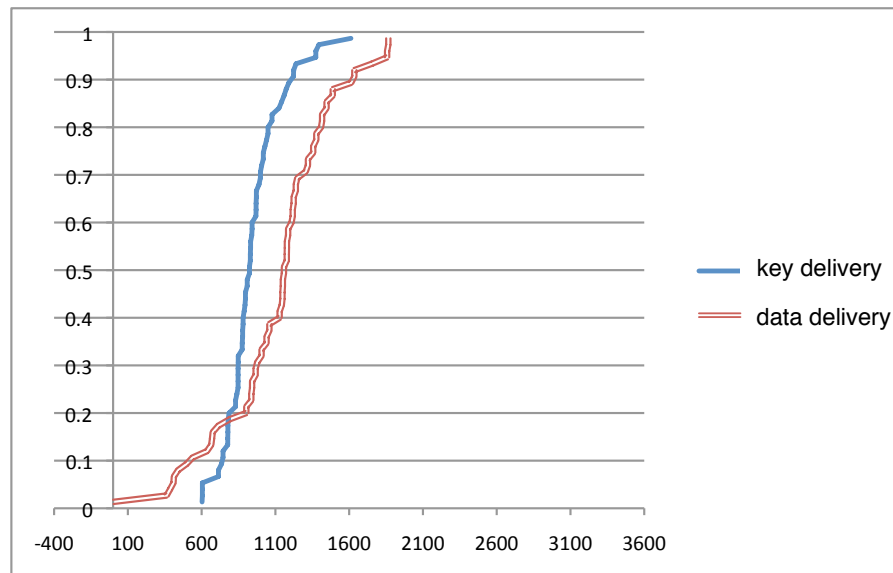


Figure 5.18: Key delivery and data delivery ratio (75 rescuers, 20m communication range)

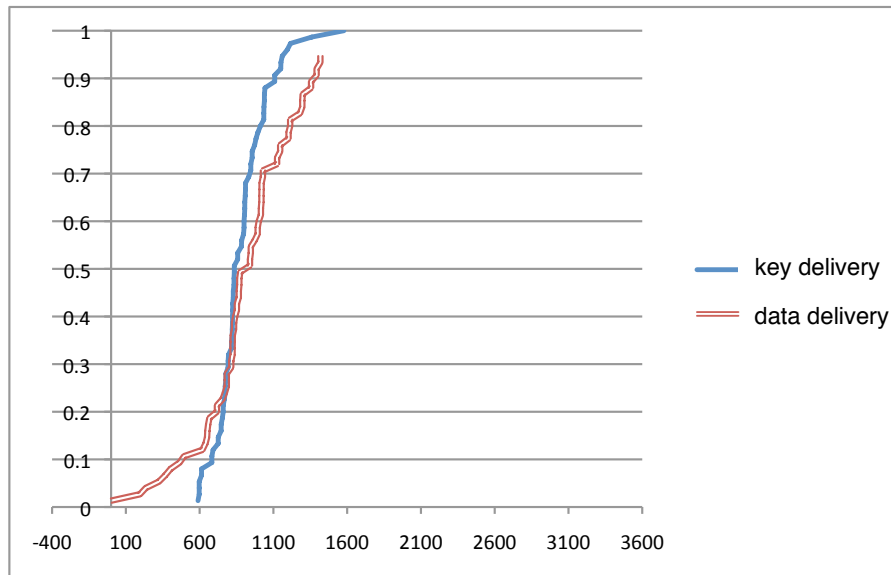


Figure 5.19: Key delivery and data delivery ratio (75 rescuers, 30m communication range)

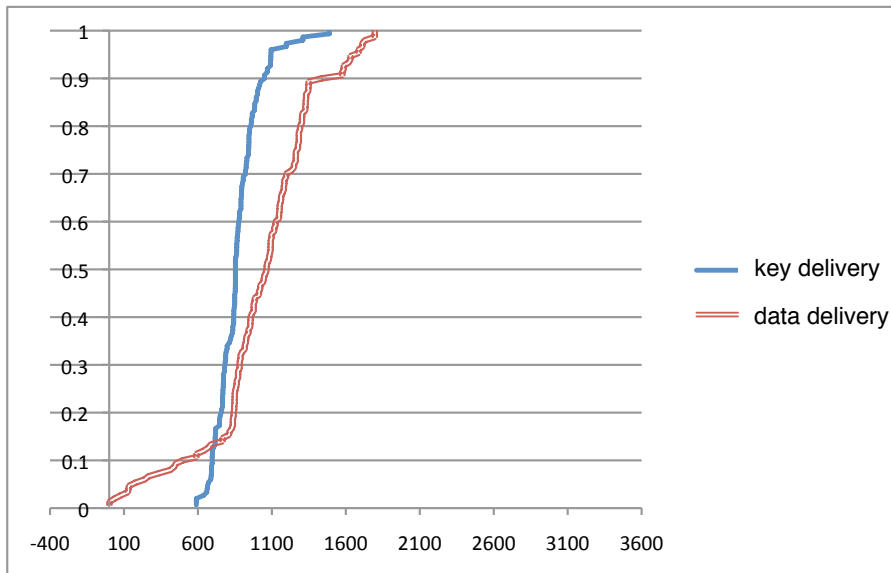


Figure 5.20: Key delivery and data delivery ratio (150 rescuers, 10m communication range)

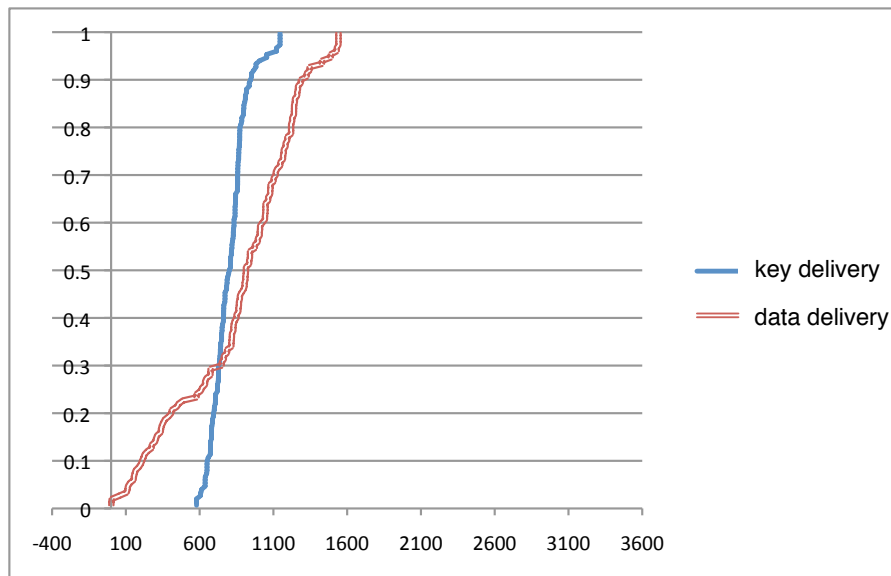


Figure 5.21: Key delivery and data delivery ratio (150 rescuers, 20m communication range)

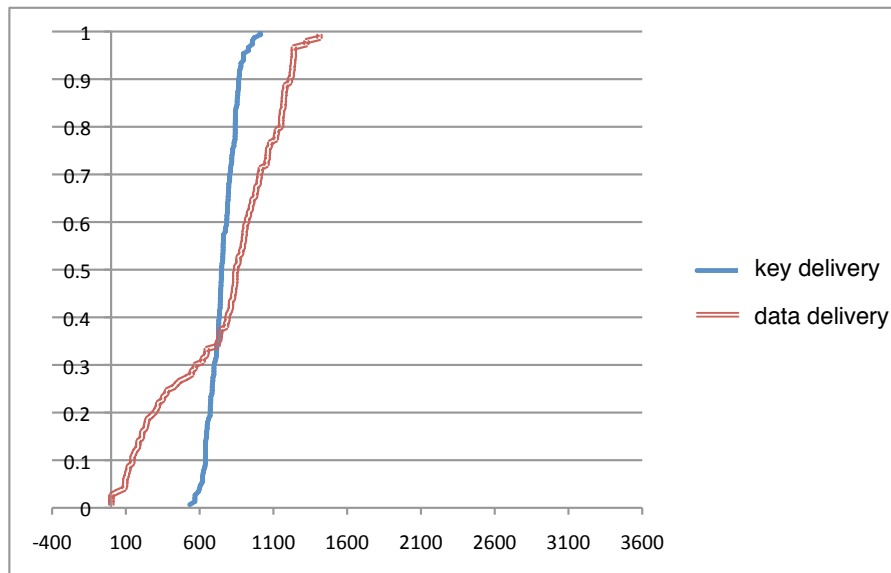


Figure 5.22: Key delivery and data delivery ratio (150 rescuers, 30m communication range)

user belonging to another one. After a key has crossed organisational boundaries, it is quickly disseminated in the other organisation leveraging the close movements of its rescuers. We emphasise that these consideration are valid in the general case and not just for our simulation model. First, as we described in Section 5.3.3, policies are always distributed starting from more than one point. Second, our model represents the actual mobility pattern rescuers follow during an emergency. Rescuers do not move alone following random routes, but move in groups in a single direction for a given length that depends on the movement area considered. Contrary to keys, the data considered for distribution is initially disseminated only from the police BCC area

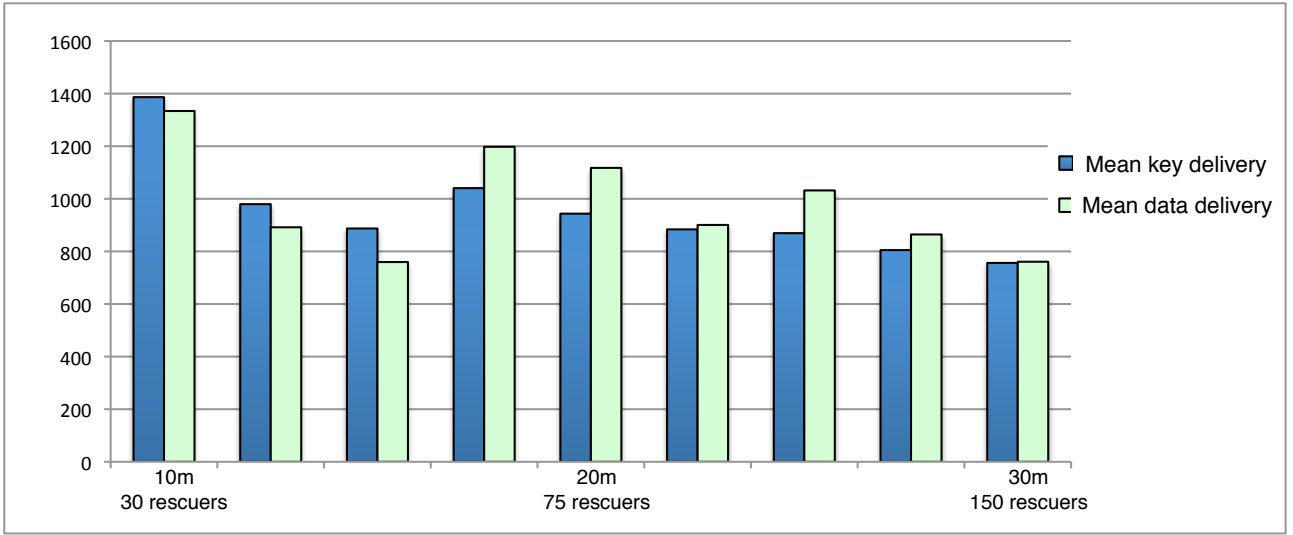


Figure 5.23: Average key delivery times and data delivery times with different parameters

and only by police units. To be received by all peers, the data must not only reach the centre of the area, but also the more distant cooperating BCCs.

We define the complexity of a policy graph as the inverse of the average probability for peers satisfying certain requirements to meet authorities that can evaluate them, i.e. the probability of obtaining membership of the corresponding authority groups. Several factors contribute to the complexity of a policy graph:

- the depth of the policy graph, i.e. the minimum number of evaluations that must be performed before a peer can become an authority;
- the width of the policy graph, i.e. the number of different disjoint groups peers can be members of;
- the number of strict-evaluation nodes, i.e. the number of peers that must meet more than one authority to acquire whole authority private keys.

The complexity of the policy graph also depends on the deployment scenario. A high probability that context-dependent conditions are verified during the crisis and a high number of peers satisfying policies at high level-nodes speeds up the evaluation process. This is why an important aspect of the mobility model is the percentage of rescuers divided amongst the several groups. Intuitively, weakening the requirements to be an evaluation authority increases the number of policy evaluations and thus speeds up the key distribution process. Similarly, stricter requirements mean fewer authorities are present in the area, thus resulting in a higher key delivery delay.

Building a more complex policy graphs for a specific scenario impacts negatively on the performance of the system (shifting the resulting sigmoid functions) and thereby influences the capacity of the system to deliver keys. Simulations run with the simpler policy graph depicted in Figure 5.24 confirm this hypothesis. The results

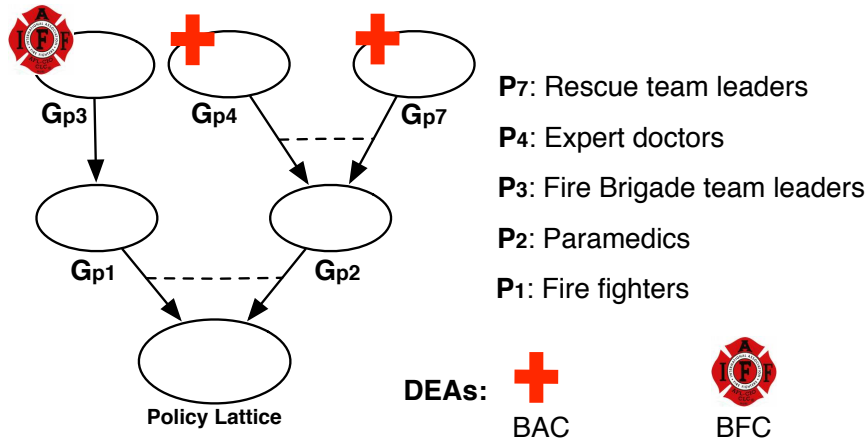


Figure 5.24: A simpler policy graph

in Figure 5.25 clearly show a shift to the left of the sigmoid function for the key delivery rate. The policy graph we used for our experiments and depicted in Figure 5.8 is more complex than required in most emergency scenarios. Moreover, we emphasise that the simulation we performed is based on the dissemination of a data package starting immediately as the rescue operations begin. This is not a realistic scenario, as rescuers would arrive at the scene with a set of pre-loaded information and only afterwards, while the operations unravel, new data would be created and disseminated. In other words, the sigmoid function for data delivery should be shifted right in all plotted graphs by an amount of time corresponding to the delay between the operations starting point and the creation and dissemination of the first data package. Moreover, since crisis management is a delicate activity that must often rely on human judgement rather than automated procedures, users entrusted with specific authorities may also be allowed to give to users memberships of certain authority groups manually, effectively providing a *break-the-glass* override. Consider for example the case where a seriously wounded victim is being assisted by a fire fighter, but no paramedic is available nearby. Assume that the fire fighter has access to the victim's information and that an uninjured victim of the accident is willing to lend assistance, attesting she is a medic. In this situation, despite the fact that the medic does not satisfy any authority policy, the fire fighter might decide to give him access to the victim's data. Break-the-glass policies would also increase the performance of the key dissemination process, as meetings between peers may result in a key exchange by means of a manual overriding of the normal authority policies that would not permit it. Finally, break-the-glass policies represent the main recovery mechanism when data dissemination goes wrong and data becomes inaccessible.

Nevertheless, the results of the simulations showed that keys for data access are mostly received before the data itself, thus granting data availability and timely accesses. This means that while responding to a crisis,

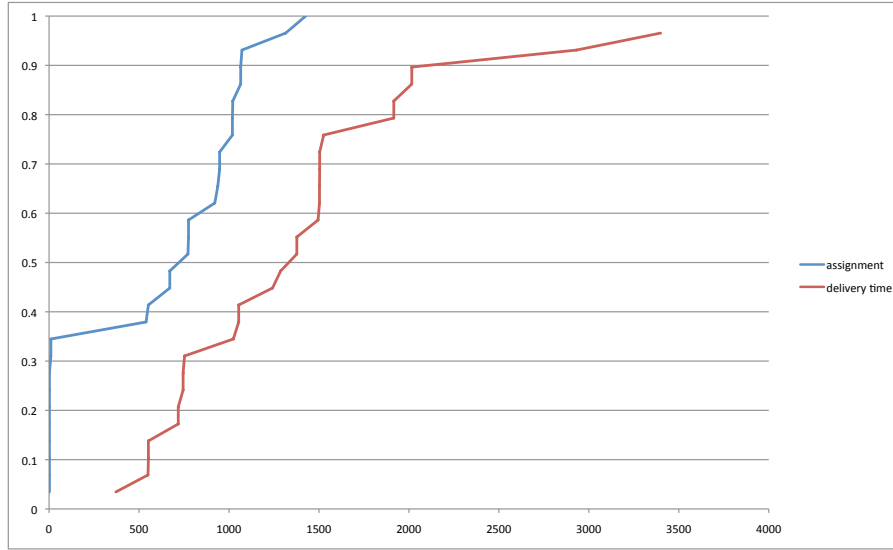


Figure 5.25: Key delivery and data delivery ratio (150 rescuers, 10m communication range) with a simpler policy graph

rescuers may use the data received immediately (or with negligible delays)as they already possess the necessary decryption keys.

5.5 Implementation Aspects

Simulations have been performed using the GUS (Geographic Urban Simulator) [BD09], which is capable of simulating large systems of interacting peers within urban settings. The GUS is built on top of the Java in Simulation Time (JiST) framework [BHvR03], which provides a discrete event simulation base. Applications can be written in Java and subsequently simulated in the environment, merging mobility traces with application logic.

The implementation of the architecture is a simple extension of the one described in Chapter 4 and most of the components have been reused. Given the simulated environment where our implementation works, communications between clients have been realised with standard TCP/IP connections. The new components performing the PAES key generation and distribution have also been implemented in Java and integrated with the existing architecture. The standard Java crypto and security libraries have been used for encryption, key generation and fragmentation.

Once the main components had already been implemented for our derived data protection architecture, implementation and integration of the PAES protocol was straightforward. Most of the components were in fact left unchanged (PAP, PIP, PDP) while others only required minor changes (PEP). The derived data controller and communication manager components were completely unrelated and separated by the PEP, which helped

in the integration. Also the basic cryptographic components were reused. In fact, while PAES and our derived data solution address completely different problems, they actually reuse the same primitives, combined according to different protocols.

5.6 Summary and Conclusions

In this chapter we presented the second and third major contributions of this thesis, i.e. the Policy-based Authority Evaluation Scheme (PAES) and its application to connected and partially disconnected environments. We first introduced policy graphs as the means to specify PAES constraints on policy evaluation, i.e. to specify what entities are trusted as evaluation authorities for specific policies. We then showed how PAES can be integrated with the general architecture described in Chapter 3 and the derived data control system described in Chapter 4. In particular, we distinguished between two different cases: data protection in fully connected environments and data protection in partially disconnected networks. In the first case, PAES allows data recipients to choose their own trusted policy evaluation authority among those satisfying the criteria defined by the data originator. Also, it enhances existing solutions by distributing the policy evaluation process, so that the set of evaluation authorities can dynamically change. This is useful whenever some authorities are unknown or unreachable. In the second case, PAES allows policies to be evaluated when only *ad hoc* connections can be formed amongst moving peers. To prove the solution does not hinder data availability, we performed simulations showing that the overhead imposed by PAES varies with the defined policy graph but is acceptable for most policies. Stricter policies come at the cost of a performance decrease. However, we consider most policies to be much simpler than the example policy we used throughout our simulation and with which the scheme's overhead was already negligible.

6 Conclusions

6.1 Summary of Thesis

Data sharing agreements between cooperating companies require that access to and usage of exchanged data are controlled whatever the environment in which data recipients are located and whatever the modifications to which data has been subjected. Enterprise rights management systems and several data encryption systems have been proposed to protect shared data. The former solutions allow data originators to publish their data on remote publishing servers that can be contacted by data recipients who want to access the data. The latter solutions allow instead data originators to directly encrypt their data for a subset of data recipients possessing specific asymmetric keys.

In this thesis we investigated two of the main problems that affect current data protection solutions and proposed solutions that enable future progress to be made. First, policies governing data access and usage are associated with the data at publication time and do not change for the whole duration of the data's lifespan. In the real world, data is not just accessed and read by recipients, but used to derive new information possibly by aggregating or merging it with data published by different originators. Not only modified data may require stricter or looser protection policies, but policies associated with aggregated data must consider the protection requirements of all information that contributed to its creation. Second, existing protection systems suffer from major limitations when data recipients or originators operate in disconnected environments. On one hand, ERM systems require that data recipients connect to remote publishing and evaluation authorities, which is impossible when connectivity is not available. On the other hand, encryption schemes such as attribute-based encryption cannot take into consideration the context into which access is requested, while keys must be distributed amongst all potential recipients *a priori*, thus assuming a perfect universal agreement between all organisations whose employees might receive the data.

We started from reviewing existing solutions, distinguishing between solutions for policy specification and solutions for policy enforcement. In particular, information flow systems and trust management systems were at the basis of the solution we propose. We described several trust management languages underlying their

problems related with credential finding and privacy issues. Finally, we introduced trusted computing technologies as the hardware/software basic layer assuring enforcement mechanisms correctly enforce the specified policies.

Given the two different problems we tried to address, we introduced a general data protection architecture that we integrated with the solutions we devised. In particular, the architecture makes use of trusted computing technologies to allow data originators to specify policy conditions on the hardware/software configuration of recipients' machines who try to access the data. The use we propose for TC technologies differs from other existing ones as we separate TC from the enforcement architecture, so that adoption of TC verification can be dictated by high-level policies.

The first major contribution of this thesis is a data labelling and information flow model for derived data control. The model is designed to allow the partial ordering of sets of policies according to the protection they provide and the specification of transformation labels that specify how pre-defined transformation functions modify the security requirements of the input/output data. The idea at the basis of the solution is to adapt discretionary policies so that they behave as floating labels in a label lattice. Policies are associated with labels that are in turn attached to data, so that the flow of labels from data to its derivation corresponds to a transition in the set of applicable policies. The proposed solution allows the control of the sensitivity levels and thus the policies associated with derived data, at the cost of restricting the usages that data can be subject to.

The second major contribution of this thesis is the policy-based authority evaluation scheme (PAES). Inspired by trust management systems, PAES tries to apply their concept to ERM systems by allowing data originators to specify a strictly controlled graph of delegated evaluation authorities and by implementing a policy evaluation protocol that does not require the collection of all the needed credentials into a central point.

The third major contribution of this thesis is the application of PAES in partially-disconnected environments. We showed that a policy evaluation across *ad hoc* links allows policy evaluation and data access and usage rights to be devolved in disconnected environments, guaranteeing timely access to data with negligible delays.

6.2 Contributions

Crisis management is one of the most demanding application scenarios for applying usage control to disseminated data, because not only is connectivity not available, but timely access to data is a fundamental requirement when human lives are at stake. The overall solution this thesis puts forward addresses all requirements by enabling data dissemination, usage control and derived data control as dictated by a data sharing agreement in disconnected environments. Moreover, it provides protection against the attacks we presented as our target in

Section 3.2.3 and depicted in Figure 3.1. For the sake of completeness, we repeat them here:

- **Curious originators trying to learn recipients' private credentials.** The PAES implementation we described in Section 5.3.2 addresses this problem. Data recipients are in fact allowed to choose their most trusted policy evaluation authority amongst a policy-defined set of acceptable ones. If a curious originator published data for an evaluation authority that recipients do not trust or know and she did not specify an additional authority policy, recipients would be aware of the risk they incur when sending their credentials for policy evaluation.
- **Attacks on the integrity of policies and metadata.** All cryptographic schemes we presented for data publication also protect the integrity of policies and metadata. When data is published using PAES in both connected and disconnected environments, the disseminated protected package (Sections 4.7.2, 5.3.2 and 5.3.3) contains an encrypted version of the data labels (from which policies can be reconstructed) and metadata. Alteration of this encrypted information would result in the impossibility of accessing the data, but not in an incorrect policy evaluation.
- **Malicious software reading unprotected data and keys.** We addressed this threat by integrating our architecture with trusted platform technologies capable of verifying the presence of untrusted software in remote machines (Section 3.5).
- **No connectivity.** The PAES implementation we described in Section 5.3.3 addresses this problem. Usage control policies can be evaluated even in disconnected environments, provided that users' devices are capable of creating *ad hoc* connections. None of the existing ERM systems would be able to provide access to data gathered or created during the rescue operations described in the application scenario of Section 1.3. On the contrary, our solution allows rescuers to protect data, exchange it and access it in a timely manner, with only negligible delays. The simulations we used to prove our approach considered a realistic mobility model with worst-case characteristics (such as the absence of pre-defined paths and the dissemination of data starting at the beginning of the rescue operations) in which we did not include *break-the-glass* policy overrides that would further improve performance.
- **Unprotected derived data.** The derived data solution we designed and presented in Chapter 4 allows newly created and derived data to be automatically protected with no intervention needed by human users and no delays. Solutions that cater for changing data [Ado04] are limited as they expect data authors to modify the protection policies manually and push them through a centralised publishing server. These solutions are rather superficial as not only may users specify incorrect policies, but they could neglect

the issue altogether. Once a policy lattice is specified, it can be used to control data automatically for its entire lifecycle, from creation through modifications to destruction.

The framework presented here does not aim to address more common attacks and threats such as denial-of-service attack, man-in-the-middle attack, substitution attack, credential forging, IP spoofing etc. for which traditional solutions can be deployed. Similarly, we have not addressed threats to the physical and human environment where data is shared, such as bribery, industrial espionage etc.

Our solution can also express traditional usage and access control policies. PAES can use directly trusted authorities to replicate the evaluation scheme of traditional ERM systems, while policy lattices can be specified with only one classification label (i.e. with one level only for all tags). The use of more complex policies depends on the specific scenario and on the requirements of the organisations sharing the data. In other words, we designed a system whose different components can be used independently and that, in its simpler form stripped of all its innovative solutions, expresses the behaviour of existing systems.

While industry and academic research focuses on developing new policy languages and models to express ever more complex data protection requirements, this thesis puts forward a solution that is orthogonal to existing solutions. Policies specified in different languages can in fact be organised in lattices and PAES graphs with minimum effort and enforcement architectures can be updated to enforce the corresponding additional controls. This is why we based our design on the architecture presented in Section 3.4 that, in its high-level aspects, is similar to traditional enforcement architectures. The significance of the ideas we put forward and their contribution to the broader picture of data protection systems lies not only in the requirements they satisfy, but also in that they do not require existing products and solutions to be fully re-designed. The novel components we integrated with the architecture presented in Section 3.4 could in fact be similarly and seamlessly integrated with other architectures.

More in general, what can be learnt from this thesis is that while current solutions can be improved and made more flexible, this always comes at the cost of either making policy specification more complex or of further constraining users in their usage of data. Balancing the tradeoff between protection and usability is no easy feat and greatly depends on the application at hand.

6.3 Operational Assumptions

The correct and efficient functioning of the solutions we proposed is based on a set of assumptions that we summarise again. First, we assumed a “homogeneous environment” where every user is equipped with a mobile device capable of wireless communication and of running the data protection architecture described

in the previous chapters. In particular, the policy decision point of such architecture makes use of a policy language and set of credentials that are assumed to be used by all the participating organisations. As such, role hierarchies and the semantics of user attributes, context attributes and obligations are considered to be well-known by all organisations. Similarly, all users are assumed to store in their device a copy of the public key of all certification authorities and root of authorities specified in a DSA. Moreover, the mobile devices are assumed to be equipped with a trusted platform module storing an endorsement key.

When DSAs are in place, we assumed that many domains are present to deal with different types of data, so that the amount of CVPs to be performed for every new data object is not overwhelming. For example, in a corporate environment new data created by researchers or product engineers in a development department may be protected by different CVPs with respect to data created by managers. Moreover, we have assumed that the set of transformations through which users can modify data is limited to those listed in a DSA. Similarly, the CVPs we proposed as example assume that users only edit documents that can be easily analysed by XPath expressions. This is however a restriction of the proposed data model and is not meant to be a general limitation of the approach.

When a *Req* CVP is specified to require users to decide the sensitivity level for a given data item, we assumed that users authorised to access and transform the data are also automatically trusted to take decisions over its protection.

Given that data is filtered before users can access it, we have assumed XML schemas and DTDs can be ignored. In other words, the parts of a filtered document that can be accessed by users may not constitute a valid document on their own.

In scenarios where connectivity is not available and credentials cannot be verified, we have considered a validity time frame during which revocation lists are assumed not to change. We have also assumed that BCCs can interact (not necessarily via wireless communication) to exchange policy graphs and keys before the start of the rescue operations.

For the sake of performance and efficiency, we have also assumed all (tag,level) pairs to be contained in the same node of a policy graph and that the lowest level authorities in the graph are the intended recipients of the data. We emphasise that this assumption is actually a best-practice that aims to increase data availability, but that the functioning of the protection system does not depend on it.

Finally, we have assumed that when users receive private keys, they can be trusted not to collude to exchange key fragments and do not evaluate policies unless all keys related to a strict-evaluation node have been already received.

Note that that any system that should integrate our solutions cannot still be considered acceptably safe without

providing an adequately safe operational envelope, based on the analysis of the hazards that threaten not only the correct usage and dissemination of data but also the data originators, recipients and users and their environment in general. Therefore, data usage must be monitored so as to discourage potentially malicious recipients and to intervene in case of policy breaches. Also, user training and physical security and handling practices for devices must be provided so that users can make the most of the available tools and so that the risk of human errors is minimised.

6.4 Caveat

One may argue that writing complex policy lattices and graphs is an additional burden for organisations at policy specification time, or at the moment when DSAs are being agreed upon. However, we think this is inevitable if organisations want to control derived data and data usage in disconnected environments with greater flexibility. Also, a greater control at corporate level implies more restrictions on final users who are forced to use data only through a pre-defined set of transformations and who cannot specify their own policies unless their organisation explicitly allows them.

A possible issue of our automatic data labelling mechanism via CVPs is that it may be a very resource-intensive operation. In fact, the time it can take depends not only on the number of CVPs available, but also on their semantics and on the size of the data to be labelled. This is why the scope of the available DSAs must be narrow enough so that only a very relevant set of CVPs is performed for every data object.

When dealing with disconnected environments, the existence of several unrelated DSAs that specify different policies for the same data objects may deteriorate the performance of our protection scheme. Users would be forced to require access to data to several different authorities (possibly coming from different organisations) which may not be present at the same time, or which may not agree on the access decision. In this case access could be delayed or denied altogether.

Finally, evaluation of policies in disconnected environments relies on the assumption that the validity of certificates can be verified with revocation lists that may have been received via opportunistic forwarding before the actual verification takes place. Similarly, evaluation relies on disseminated context tokens that may not represent the current situation. When a new version of a context token is created and disseminated, it is received by evaluation authorities with a delay during which incorrect policy evaluations may have been performed. This is inevitable as no remote services can be contacted to obtain up-to-date revocation lists and context tokens. In the same way, if policies are updated during an emergency, they must be pushed to all the rescuers in the emergency area. Dissemination of the new policies suffer from the inevitable delay caused by the opportunistic

distribution and thus users may be evaluated under different policies, depending on their location.

6.5 Future Work

The ERM and DRM industry has been frequently challenged in recent years by software products that could circumvent its data protection solutions. This situation led companies active in the DRM area to adopt a new strategy where software updates are continually pushed on clients to counteract the latest anti-DRM software, which in turn is usually developed by reverse-engineering DRM products. This caused the DRM market to reach a stalemate where few real innovations are proposed. One of these, trusted computing, has received several criticisms [Sch03, Fel03] that, together with the lack of a common standard implementation amongst software and hardware producers, has led to a slow adoption. In fact, while nowadays several hardware products carry built-in trusted computing chips, few software products are really integrated with them. On the other hand, ERM solutions proposed by both industry and academia have kept evolving by introducing new type of controls such as those defined in the $UCON_{ABC}$ model [PS04]. In the future, ERM solutions will be even more integrated within corporate IT infrastructures, controlling every data exchange not only between employees of the same organisation but also of partner organisations. However, because of the similarities between the architectures of all ERM solutions, it seems not enough effort is being put to study new enforcement mechanisms that could surpass traditional centralised approaches.

The work described in this thesis is a first step to address the problems we considered in existing data protection systems. However, it is by no means a complete work, as further studies can be conducted. One of the interesting directions for future work on our derived data solution is the study of possible solutions to track provenance, i.e. to keep track of the original data and sequence of transformations that ended up creating a specific data object. A provenance record for data could be very useful when deciding what policies should be applicable for a given access request, since all policies applicable to all data objects in the *lineage* of the requested data could be taken into consideration and merged. This could be performed by taking into consideration the length of the data lineage (e.g. the number of transformation steps it took to create the current data) and the depth at which each data object appears. Versioning systems are an example application where data modifications and transformations are tracked and can be rolled back if necessary, actually building a *provenance* record of data. Consider a document being modified so that its classification increases. Some recipients who could access the document prior to its transformation may now be denied access. However, they may be allowed to access the older versions of the modified elements in the XML tree, provided the document contains them too. In other words, recipients may have different views of the same document, being

able to access different versions of its elements. This possibility also raises other questions. What happens when a user transforms a document for which she can access only older versions? Which version does the transformation consider? Would the output document contain also the unmodified versions? Would users be able to see completely deleted elements of the document?

The possibility to hide parts of a document selectively also raises another problem. Consider the case where a paramedic is reading the medical details of a patient, but cannot access the section on allergies. Completely removing the section is extremely dangerous, since the paramedic may give the patient drugs she is not supposed to take. In other words, besides *read* and *transformation* rights it should be possible to specify a special *visibility* or *existence* right allowing recipients to see the existence of a document's element, but not its content. This possibility should then be integrated with version control. For example, let us consider the case of a recipient being granted *read* rights on an older version of an element and *visibility* rights on a more recent one. What would the recipient be allowed to see? Also, it is important to consider that documents may have rules guiding their creation and validation and filtering them may cause them to become invalid or to be shown in an invalid form. Future work should thus investigate the possibility of having filtered documents that work with such schemata.

Many directions for future work are also possible for PAES. Even if the approach offers great flexibility, data recipients are still limited in their choice of evaluation authorities to those acceptable to the originator as defined in the policies. A negotiation process between data recipients and originators could offer more flexibility. However, this possibility raises several problems. First, it would not be feasible in a data dissemination environment where users may not know each other and may be online at different times. Moreover, this may require the data to be repackaged for each negotiating recipient and would result in a set of disseminated packages containing the same information but being controlled by different rules.

Future work may try to integrate PAES with frameworks for hybrid networks such as Haggie [SHCD06, SSH⁺07]. In fact, although we implemented the PAES components to be integrated into our proposed architecture, its testing has been limited to simulated environments and communications via traditional TCP/IP on ethernet and wifi channels. Future work should thus investigate the use of different wireless *ad hoc* networks such as Bluetooth. It would also be interesting to study the applicability of PAES to new communication protocols such as near-field communication (NFC) and RFID. The processing power of active NFC and RFID tags could in fact be used to determine who should be entitled to access the tag's information.

Given the large availability of policy languages and models, the focus of research is now shifting towards finding new and more effective techniques for policy analysis and validation [CLM⁺09]. Writing long and complex policies can in fact become a waste of time if a minimum error in the specification not spotted by

the policy writer can compromise data protection. In this sense, automatic analysis tools for usage control policies would greatly help in the specification of both policy lattices and PAES policy graphs. For example, it could be possible to determine whether data can be transformed so that no one can access it or so that conflicting obligations must be performed to use it. Or it could be possible to determine whether a PAES policy graph contains hidden cycles where the constraints of policies at higher levels are a subset of the constraints of policies at lower levels. In other words, to determine whether entities could be evaluation authorities for themselves (which is not always an undesirable property, e.g. when policies must be evaluated locally).

Also, while in the solution we proposed policies can be directly written for specific data or for large sets of objects via CVP, resulting in a possibly complex specification, it would be possible to use machine learning techniques [CRL10, CRR⁺09] to learn policies from human behaviour and decisions.

Finally, while we have described how PAES can be integrated with data protection architectures, it may be possible to use it for different applications such as certification and trust management systems. It could for example be integrated with X509 certificates so that chains of trust can be specified that depends on PAES graphs.

A last problem that can be investigated and that regards both connected and disconnected scenarios, is the problem of recovering from errors in the data protection and thus from data leaks. This includes the problem of detecting the leaks and identifying who first caused them, and then to take measures to prevent future ones. To do so it may be possible to rely on quantitative trust management techniques so that the trustworthiness of users is decreased as soon as they misbehave. Data distribution can then be made conditional on the level of trust that can be put in the data recipient.

Bibliography

- [Abr93] Marshall Abrams. Renewed understanding of access control policies. In 16th National Computing Security Conference, pages 87–96, 1993.
- [act98] Data protection act, 1998. <http://www.legislation.gov.uk/>.
- [Ado04] Adobe. A primer on electronic document security, 2004.
- [AFB05] Mikhail J. Atallah, Keith B. Frikken, and Marina Blanton. Dynamic and efficient key management for access hierarchies. In Proceedings of the 12th ACM conference on Computer and communications security, CCS '05, pages 190–202, New York, NY, USA, 2005. ACM.
- [AFMT04] N. Aschenbruck, M. Frank, P. Martini, and J. Tölle. Human mobility in manet disaster area simulation - a realistic approach. In 29th Annual IEEE Conf. on Local Computer Networks (LCN), pages 668–675, Tampa, Florida, U.S.A., November 2004.
- [AG02] V. Atluri and A. Gal. An authorization model for temporal and derived data: securing information portals. ACM Trans. Inf. Syst. Secur., 5(1):62–94, 2002.
- [AGLL05] D. Agrawal, J. Giles, K. Lee, and J. Lobo. Policy ratification. In POLICY '05: Proceedings of the Sixth IEEE International Workshop on Policies for Distributed Systems and Networks, pages 223–232, Washington, DC, USA, 2005. IEEE Computer Society.
- [AGPG⁺07] N. Aschenbruck, E. Gerhards-Padilla, M. Gerharz, M. Frank, and P. Martini. Modelling mobility in disaster area scenarios. In Proc. of the 10th Int. Symp. on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM), pages 4–12, Chania, Crete Island, Greece, October 2007.
- [AHK⁺91] M. Abrams, J. Heaney, O. King, L. LaPadula, M. Lazear, and I. Olson. A generalized framework for access control: Toward prototyping the ORGCON policy. In 14th NIST-NCSC National Computing Security Conference, pages 257–266, 1991.

- [AT83] Selim G. Akl and Peter D. Taylor. Cryptographic solution to a problem of access control in a hierarchy. ACM Trans. Comput. Syst., 1:239–248, August 1983.
- [Aut05] Authentica. Enterprise rights management for document protection, 2005. White Paper.
- [BBC⁺00] Elisa Bertino, M. Braun, Silvana Castano, Elena Ferrari, and Marco Mesiti. Author-X: A java-based system for xml data protection. In IFIP TC11/ WG11.3 14th An. Conf. on atabase Security (DBSec), pages 15–26, Schoorl, The Netherlands, 2000. Kluwer.
- [BCF01] Elisa Bertino, Silvana Castano, and Elena Ferrari. Securing XML documents with Author-X. IEEE Internet Computing, 5(3):21–, 2001.
- [BD09] R. Ball and N. Dulay. Approximating travel times using opportunistic networking. In 2nd IEEE Int. Workshop on Opportunistic Networking, pages 844–849, May 2009.
- [Bel73] D. E. Bell. Secure computer systems: A refinement of the mathematical model. Technical Report ESD-TR-278, The Mitre Corp., 1973.
- [BF01] Dan Boneh and Matt Franklin. Identity-based encryption from the weil pairing, 2001.
- [BFG10] Moritz Y. Becker, Cédric Fournet, and Andrew D. Gordon. Secpal: Design and semantics of a decentralized authorization language. J. Comput. Secur., 18:619–665, December 2010.
- [BFIK99] Matt Blaze, Joan Feigenbaum, John Ioannidis, and Angelos D. Keromytis. Secure internet programming. chapter The role of trust management in distributed systems security, pages 185–210. Springer-Verlag, London, UK, 1999.
- [BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In Proceedings of the 1996 IEEE Symposium on Security and Privacy, SP '96, pages 164–, Washington, DC, USA, 1996. IEEE Computer Society.
- [BHvR03] R. Barr, Z. J. Haas, and R. van Renesse. Jist: Embedding simulation time into a virtual machine. In 8th Int. Symp. on Modeling Analysis and Simulation of Wireless and Mobile Systems (MSWiM), page 16, Montreal, Quebec, Canada, 2003.
- [Bib77] K. J. Biba. Integrity considerations for secure computer systems. Technical report, MITRE Corp., 04 1977.
- [BJS99] Elisa Bertino, Sushil Jajodia, and Pierangela Samarati. A flexible authorization mechanism for relational data management systems. ACM Trans. Inf. Syst., 17:101–140, April 1999.

- [BL73] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations. Technical Report M74-244, The Mitre Corp., May 1973.
- [BN89] Dr. David F.C. Brewer and Dr. Micheal J. Nash. The chinese wall security policy. Security and Privacy, IEEE Symposium on, 0:206, 1989.
- [BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy, SP '07, pages 321–334, Washington, DC, USA, 2007. IEEE Computer Society.
- [BW96] B. Bollig and I. Wegener. Improving the variable ordering of obdds is np-complete. Computers, IEEE Transactions on, 45(9):993 –1002, sep 1996.
- [CEE⁺02] Dwaine Clarke, Jean-Emile Elie, Carl Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in spki/sdsi. J. Comput. Secur., 9:285–322, February 2002.
- [CFL⁺97] Yang-Hua Chu, Joan Feigenbaum, Brian LaMacchia, Paul Resnick, and Martin Strauss. Referee: trust management for web applications. In Selected papers from the sixth international conference on World Wide Web, pages 953–964, Essex, UK, 1997. Elsevier Science Publishers Ltd.
- [CG07] M. Conti and S. Giordano. Multihop ad hoc networking: The reality. IEEE Communications Magazine, 45(4):88–95, April 2007.
- [Cha07] Melissa Chase. Multi-authority attribute based encryption. In TCC, pages 515–534, 2007.
- [CLM⁺] Lorrie Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. The platform for privacy preferences 1.0 (p3p1.0) specification.
<http://www.w3.org/P3P/>.
- [CLM⁺09] Robert Craven, Jorge Lobo, Jiefei Ma, Alessandra Russo, Emil Lupu, and Arosha Bandara. Expressive policy analysis with enhanced system dynamicity. In Proceedings of the 4th International Symposium on Information, Computer, and Communications Security, ASIACCS '09, pages 239–250, New York, NY, USA, 2009. ACM.
- [Con] The Consequence Consortium. Context-aware data centric information sharing.
<http://www.consequence-project.eu/>.

- [Cra04] Jason Crampton. Applying hierarchical and role-based access control to xml documents. In Proceedings of the 2004 workshop on Secure web service, SWS '04, pages 37–46, New York, NY, USA, 2004. ACM.
- [CRL10] Domenico Corapi, Alessandra Russo, and Emil Lupu. Inductive logic programming as abductive search. In ICLP (Technical Communications), pages 54–63, 2010.
- [CRR⁺09] Domenico Corapi, Oliver Ray, Alessandra Russo, Arosha K. Bandara, and Emil C. Lupu. Learning rules from user behaviour. In AIAI, pages 459–468, 2009.
- [cry97] IBM Cryptolope, 1997.
http://news.cnet.com/IBM-closes-Cryptolopes-unit/2100-1001_3-206465.html.
- [CW87] D. D. Clark and D. R. Wilson. A comparison of commercial and military computer security policies. In IEEE Symposium on Security and Privacy, pages 184–195, 1987.
- [DD77] D. E. Denning and P. J. Denning. Certification of programs for secure information flow. Commun. ACM, 20:504–513, July 1977.
- [DDLS01] Nicodemous Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The Ponder Policy Specification Language. In Int. Workshop on Policies for Distributed Systems and Networks (POLICY), volume 1995, pages 18–39, Bristol, England, 2001.
- [DdVPS00] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Sarnarati. Securing xml documents. In INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, pages 121–135. Springer, 2000.
- [DdVPS02] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, and P. Samarati. A fine-grained access control system for xml documents. ACM Trans. Inf. Syst. Secur., 5(2):169–202, 2002.
- [Den76] D. E. Denning. A lattice model of secure information flow. Commun. ACM, 19(5):236–243, May 1976.
- [Don08] Changyu Dong. Non-monotonic Trust Management for Distributed Systems. PhD thesis, Imperial College London, 2008.
- [DR] Tien Tuan Anh Dinh and Mark Dermot Ryan. Trusted computing: Tcg proposals.
<http://www.cs.bham.ac.uk/~mdr/teaching/modules/security/lectures/TrustedComputingTCG.html>.

- [Dra06] Boris Dragovic. Casper: Containment-Aware Security for Pervasive Computing Environments. PhD thesis, University of Cambridge, 2006.
- [DRD08] Changyu Dong, Giovanni Russello, and Naranker Dulay. Flexible resolution of authorisation conflicts in distributed systems. In DSOM, pages 95–108, 2008.
- [EKV⁺05] P. Efstathopoulos, M. N. Krohn, S. Vandebugart, C. Frey, D. Ziegler, E. Kohler, D. Mazières, M. F. Kaashoek, and R. Morris. Labels and event processes in the Asbestos operating system. In SOSP, pages 17–30, 2005.
- [Ell99] C. Ellison. Spki requirements, 1999. RFC 2692.
- [ELM⁺03] Paul England, Butler Lampson, John Manferdelli, Marcus Peinado, and Bryan Willman. A trusted open platform. Computer, 36:55–62, July 2003.
- [EMC] EMC. Emc documentum information rights services datasheet. White Paper.
- [eud12] European data protection directive, 2012.
<http://ec.europa.eu/justice/data-protection/>.
- [fai07] How fairplay works: Apple’s itunes drm dilemma, 2007. <http://www.roughlydrafted.com/RD/RDM.Tech.Q1.07/2A351C60-A4E5-4764-A083-FF8610E66A46.html>.
- [Fel03] Edward W. Felten. Understanding trusted computing: Will its benefits outweigh its drawbacks? IEEE Security and Privacy, 1(3):60–62, 2003.
- [foi66] Freedom of information act (usa), 1966. <http://www.foia.gov/>.
- [foi00] Freedom of information act (united kingdom), 2000. <http://www.legislation.gov.uk/>.
- [GBLD] D. Golby, S. Barker, F. Lockwood, and A.G. Davies. Consequence requirements specification for the policy- based security for the crisis management test bed.
<http://www.consequence-project.eu/>.
- [GGSL11] Anandha Gopalan, Vaibhav Gowadia, Enrico Scalavino, and Emil Lupu. Policy driven remote attestation. In Proceedings of the third Conference on Security and Privacy in Mobile Information and Communication Systems, Aalborg, Denmark, 2011. ICST.
- [GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on

Computer and communications security, CCS '06, pages 89–98, New York, NY, USA, 2006. ACM.

- [Gra89] Richard Graubart. On the need for a third form of access control. In 12th National Computing Security Conference, pages 296–304, Baltimore, MD, USA, 1989.
- [Gro] Trusted Computing Group. <http://www.trustedcomputinggroup.org/>.
- [Gua01] Content Guard. eXtensible rights Markup Language (XrML) 2.0, 2001. Specification.
- [GZS09] M. Graf, N. Zunic, and M. G. Swimmer. Process and method for classifying structured data, 05 2009.
- [HMM⁺00] Amir Herzberg, Yosi Mass, Joris Michaeli, Yiftach Ravid, and Dalit Naor. Access control meets public key infrastructure, or: Assigning roles to strangers. In Proceedings of the 2000 IEEE Symposium on Security and Privacy, pages 2–, Washington, DC, USA, 2000. IEEE Computer Society.
- [Ian02] R. Iannella. Open Digital Rights Language (ODRL), Version 1.1. W3c note, World Wide Web Consortium, 2002.
- [IDS02] Takeshi Imamura, Blair Dillaway, and Ed Simon. Xml encryption syntax and processing, 2002. <http://www.w3.org/TR/xmlenc-core/>.
- [IOfS] Switzerland International Organization for Standardization, Geneva. ISO/IEC 27001:2005 - information technology – security techniques – information security management systems – requirements.
- [JD96] Hastert J. Dennis. Health insurance portability and accountability act, July 1996. <http://www.gpo.gov/>.
- [JS91] Sushil Jajodia and Ravi Sandhu. Toward a multilevel secure relational data model. SIGMOD Rec., 20(2):50–59, 1991.
- [JSSS01] Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. ACM Trans. Database Syst., 26:214–260, June 2001.
- [key03] Cryptographic world: The cryptography guide, 2003. <http://www.cryptographyworld.com/>.

- [Lam74] Butler W. Lampson. Protection. SIGOPS Oper. Syst. Rev., 8:18–24, January 1974.
- [LDS⁺90] Teresa F. Lunt, Dorothy E. Denning, Roger R. Schell, Mark Heckman, and William R. Shockley. The seaview security model. IEEE Trans. Software Eng., 16(6):593–607, 1990.
- [LGY07] L. Lilien, A. K. Gupta, and Z. Yang. Opportunistic networks for emergency applications and their standard implementation framework. In 26th IEEE Int. Performance Computing and Communications Conf. (IPCCC), pages 588–593, New Orleans, Louisiana, USA, April 2007.
- [Lip82] Steven B. Lipner. Non-discretionary controls for commercial applications. Security and Privacy, IEEE Symposium on, 0:2, 1982.
- [liq06] Liquid Machines and Microsoft Windows Rights Management Services (RMS): End-to-end Rights Management for the Enterprise, 2006. Accessed September 2009. URL: [http://www.cmdsolutions.com/pdfs/LiquidMachines% 20Windows%20RMS%20Business %20White%20Paper%20FINAL%20060213.pdf](http://www.cmdsolutions.com/pdfs/LiquidMachines%20Windows%20RMS%20Business%20White%20Paper%20FINAL%20060213.pdf).
- [LKB⁺06] L. Lilien, Z. Huma Kamal, V. Bhuse, A. Gupta, and Wise (wireless Sensornet Lab. Opportunistic networks: The concept and research challenges in privacy and security. In Proc. NSF Int. Workshop on Research Challenges in Security and Privacy for Mobile and Wireless Networks (WSPWN), Miami, Florida, U.S.A., March 2006.
- [LM09] Alexander Lux and Heiko Mantel. Declassification with explicit reference points. In ESORICS, pages 69–85, 2009.
- [LMW02] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In Proceedings of the 2002 IEEE Symposium on Security and Privacy, pages 114–, Washington, DC, USA, 2002. IEEE Computer Society.
- [LR00] Q. Li and D. Rus. Sending messages to mobile users in disconnected ad-hoc wireless networks. In 6th ACM/IEEE Int. Conf. on Mobile Computing and Networking (MOBICOM), pages 44–55, Boston, Massachusetts, USA, August 2000.
- [LY06] Zude Li and Xiaojun Ye. Towards a dynamic multi-policy dissemination control model: (dmdcon). SIGMOD Rec., 35:33–38, March 2006.
- [mar09] The Marlin Open Digital Content Sharing Platform, 2009. Accessed February 2010. URL: <http://www.marlin-community.com/>.

- [McL90] John McLean. The specification and modeling of computer security. IEEE COMPUTER, 23:9–16, 1990.
- [Mic05] Microsoft. Technical overview of windows rights management services for windows server 2003. Accessed July 2011, 2005. <http://download.microsoft.com/download/8/d/9/8d9dbf4a-3b0d-4ea1-905b-92c57086910b/rmstechoverview.doc>.
- [ML00] Andrew C. Myers and Barbara Liskov. Protecting privacy using the decentralized label model. ACM Trans. Softw. Eng. Methodol., 9(4):410–442, 2000.
- [MPB03] M. Casassa Mont, S Pearson, and P. Bramhall. Towards accountable management of identity and privacy: Sticky policies and enforceable tracing services. In 14th Int. Workshop on Database and Expert Systems Applications (DEXA), pages 377–382, Prague, Czech Republic, September 2003.
- [MS03] Gerome Miklau and Dan Suciu. Controlling access to published data using cryptography. In Proceedings of the 29th international conference on Very large data bases - Volume 29, VLDB '2003, pages 898–909. VLDB Endowment, 2003.
- [MS04] Gerome Miklau and Dan Suciu. Containment and equivalence for a fragment of xpath. J. ACM, 51:2–45, January 2004.
- [MSN⁺04] Hiroshi Maruyama, Frank Seliger, Nataraj Nagaratnam, Tim Ebringer, Seiji Munetoh, Sachiko Yoshihama, and Taiga Nakamura. Trusted platform on demand (tpod). Technical report, 2004. <http://www.tr1.ibm.com/people/munetoh/RT0564.pdf>.
- [MTMA85] Stephen J. MacKinnon, Peter D. Taylor, Henk Meijer, and Selim G. Akl. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. IEEE Trans. Comput., 34:797–802, September 1985.
- [Mye99] Andrew C. Myers. Jflow: Practical mostly-static information flow control. In In Proc. 26th ACM Symp. on Principles of Programming Languages (POPL), pages 228–241, 1999.
- [NS02] Frank Neven and Thomas Schwentick. Xpath containment in the presence of disjunction, dtds, and variables. In Proceedings of the 9th International Conference on Database Theory, ICDT '03, pages 315–329, London, UK, 2002. Springer-Verlag.
- [NV10] Aarthi Nagarajan and Vijay Varadharajan. Modelling dynamic trust with property based attestation in trusted platforms. In Sara Foresti and Sushil Jajodia, editors, Data and

Applications Security and Privacy XXIV, volume 6166 of Lecture Notes in Computer Science, pages 257–272. Springer Berlin / Heidelberg, 2010.

- [OSW07] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In Proceedings of the 14th ACM conference on Computer and communications security, CCS '07, pages 195–203, New York, NY, USA, 2007. ACM.
- [pat88] Copyright, designs and patents act, 1988. <http://www.legislation.gov.uk/>.
- [pat10] Copyright law of the united states, 2010. <http://www.copyright.gov/>.
- [pc] Oasis project consortium. Open advanced system for disaster & emergency management. <http://www.oasis-fp6.org/>.
- [PHB06] Alexander Pretschner, Manuel Hilty, and David A. Basin. Distributed usage control. Commun. ACM, 49(9):39–44, 2006.
- [PMK99] Vassilis Prevelakis, Jean-Henry Morin, and Dimitri Konstantas. Controlling the dissemination of electronic documents. In 10th Int. Workshop on Database and Expert Systems Applications (DEXA), pages 869–873, Florence, Italy, 1999.
- [PS02] J. Park and R. Sandhu. Originator control in usage control. Policies for Distributed Systems and Networks, IEEE International Workshop on, 0:0060, 2002.
- [PS04] J. Park and R. S. Sandhu. The $UCON_{ABC}$ usage control model. ACM Trans. Inf. Syst. Secur., 7(1):128–174, 2004.
- [PSK⁺03] P.Ashley, S.Hada, G. Karjoth, C. Powers, and M. Schunter. The enterprise privacy authorization language (epal 1.1), 2003. Reader’s Guide to the Documentation.
- [PSS00] J. Park, R. S. Sandhu, and J. Schifalacqua. Security architectures for controlled digital information dissemination. In 16th An. Computer Security Applications Conf. (ACSAC), pages 224–, New Orleans, USA, 2000. IEEE Computer Society.
- [RDD07] Giovanni Russello, Changyu Dong, and Naranker Dulay. Authorisation and conflict resolution for hierarchical domains. In 8th IEEE Int. Workshop on Policies for Distributed Systems and Networks(POLICY), pages 201–210, Bologna, Italy, 2007.

- [RHB⁺07] R. Ramanathan, R. Hansen, P. Basu, R. Rosales-Hain, and R. Krishnan. Prioritized epidemic routing for opportunistic networks. In Proc. of the 1st int. MobiSys Workshop on Mobile Opportunistic Networking (MobiOpp), pages 62–66, New York, USA, March 2007. ACM.
- [RL96] Ronald L. Rivest and Butler Lampson. Sdsi - a simple distributed security infrastructure. Technical report, 1996.
- [RLB⁺] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo. An algebra for fine-grained integration of xacml policies. CERIAS tech. report 2008-21.
- [San93] Ravi S. Sandhu. Lattice-based access control models. IEEE Computer, 26(11):9–19, 1993.
- [SCFY96] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. Computer, 29:38–47, February 1996.
- [Sch03] Seth Schoen. Trusted computing: Promise and risk. Technical report, Electronic Frontier Foundation, 2003.
- [SdV00] Pierangela Samarati and Sabrina De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In FOSAD, pages 137–196, 2000.
- [Sec07] Avoco Secure. Choosing an enterprise rights management system: Architectural approach, 2007. URL: www.windowsecurity.com/uplarticle/Authentication_and_Access_Control/ERM-architectural-approaches.pdf.
- [SHCD06] J. Scott, P. Hui, J. Crowcroft, and C. Diot. Huggle: A Networking Architecture Designed Around Mobile Users. In IFIP Conference on Wireless On demand Network Systems (WONS), pages 78–86, Les Menuires, France, January 2006.
- [SK05] A. Seth and S. Keshav. Practical security for disconnected nodes, 2005. Centre for Applied Cryptographic Research (CACR): Technical Report.
- [Slo94] Morris Sloman. Policy driven management for distributed systems. Journal of Network and Systems Management, 2:333–360, 1994. 10.1007/BF02283186.
- [sox02] Sarbanes-oxley act, 2002. <http://www.soxlaw.com/>.
- [SP03] Ravi S. Sandhu and Jaehong Park. Usage Control: A Vision for Next Generation Access Control. In Proc. 2nd Int’l Workshop on Mathematical Methods, Models, and Architectures for Computer Network Security, volume 2776 of LNCS, pages 17–31. Springer, 2003.

- [SRZ06] Ravi Sandhu, Kumar Ranganathan, and Xinwen Zhang. Secure information sharing enabled by trusted computing and pei models. In Proceedings of the 2006 ACM Symposium on Information, computer and communications security, ASIACCS '06, pages 2–12, New York, NY, USA, 2006. ACM.
- [SS04] Ahmad-Reza Sadeghi and Christian Stübke. Property-based attestation for computing platforms: caring about properties, not mechanisms. In Proceedings of the 2004 workshop on New security paradigms, NSPW '04, pages 67–77, New York, NY, USA, 2004. ACM.
- [SS05] Andrei Sabelfeld and David Sands. Declassification: Dimensions and principles. In Proc. of the 18th IEEE Comp. Sec. Foundations Workshop (CSFW05), pages 255–269, 2005.
- [SSH⁺07] J. Su, J. Scott, P. Hui, J. Crowcroft, E. de Lara, C. Diot, A. Goel, M. Lim, and E. Upton. Hagggle: Seamless networking for mobile applications. In Ubiquitous Computing, 9th Int. Conf. (Ubicomp), pages 391–408, Innsbruck, Austria, September 2007.
- [SSR06] Vipin Swarup, Len Seligman, and Arnon Rosenthal. A data sharing agreement framework. 4332:22–36, 2006.
- [SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In EUROCRYPT, pages 457–473, 2005.
- [SZJvD04] Reiner Sailer, Xiaolan Zhang, Trent Jaeger, and Leendert van Doorn. Design and implementation of a tcb-based integrity measurement architecture. In Proceedings of the 13th conference on USENIX Security Symposium - Volume 13, SSYM'04, pages 16–16, Berkeley, CA, USA, 2004. USENIX Association.
- [SZRC06] Ravi S. Sandhu, Xinwen Zhang, Kumar Ranganathan, and Michael J. Covington. Client-side access control enforcement using trusted computing and pei models. J. High Speed Networks, 15(3):229–245, 2006.
- [TS04] Roshan Thomas and Ravi Sandhu. Towards a multi-dimensional characterization of dissemination control. In 5th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY), page 197, 2004.
- [VB00] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. 2000. Duke Technical Report CS-2000-06.

- [VEK⁺07] S. Vandebugart, P. Efstathopoulos, E. Kohler, M. N. Krohn, C. Frey, D. Ziegler, M. F. Kaashoek, R. Morris, and D. Mazières. Labels and event processes in the Asbestos operating system. ACM Trans. Comput. Syst., 25(4), 2007.
- [win11] Windows media licensing program, 2011.
<http://www.microsoft.com/windows/windowsmedia/licensing/>.
- [Woo02] Peter T. Wood. Containment for xpath fragments under dtd constraints. In Proceedings of the 9th International Conference on Database Theory, ICDT '03, pages 300–314, London, UK, 2002. Springer-Verlag.
- [WOR⁺74] K.G. Walter, W.F. Ogden, W.C. Rounds, F.T. Bradshaw, S.R. Ames, and D.G. Sumaway. Primitive models for computer security. Technical Report TR ESD-TR-4-117, Case Western Reserve University, 1974.
- [xac05] eXtensible Access Control markup language (xacml) version 2.0, 2005.
- [xpa99] Xml path language (xpath), 1999. <http://www.w3.org/TR/xpath/>.
- [xpa02] Xml-signature xpath filter 2.0, 2002. <http://www.w3.org/TR/xmlsig-filter2/>.
- [xqu10] Xquery 1.0: An xml query language (second edition), 2010.
<http://www.w3.org/TR/xquery/>.
- [ZJMW05] Nicola Zannone, Sushil Jajodia, Fabio Massacci, and Duminda Wijesekera. Maintaining privacy on derived objects. In WPES '05: ACM workshop on Privacy in the electronic society, pages 10–19, New York, NY, USA, 2005. ACM.