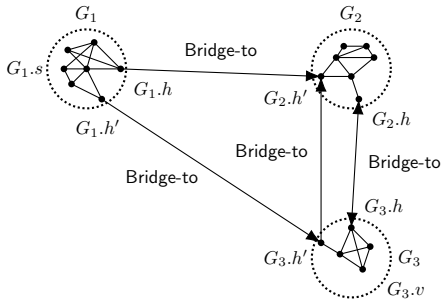
RPPM: A Relationship-Based Access Control Model Utilising Relationships, Paths and Principal Matching.

by

James Sellwood

(for the degree of PhD in Information Security)



Date: May 1, 2017
Supervisor: Professor Jason Crampton
Information Security Group, Royal Holloway, University of London

Submitted as part of the requirements for the award of the degree of:
PhD in Information Security at Royal Holloway, University of London.

I, James Sellwood, declare that this thesis is all my own work and that I have acknowledged all quotations from the published or unpublished works of other people. I declare that I have also read the statements on plagiarism in Section 1 of the Regulations on Assessment Offences and in accordance with it I submit this thesis as my own work.

Signature:

Date: May 1, 2017

Abstract

Since their introduction, the use (and abuse) of computer systems has grown astronomically, and consequently so has the need to manage the sharing of data between users, processes and systems. Within a computer it is the access control system, implementing a formally defined access control model, which is responsible for enacting the security policies to prevent unauthorized disclosure, manipulation, and deletion of system and user data. I begin this thesis by discussing the background and development of key historical access control models, and by highlighting their features and limitations. In the remainder of this thesis I then present the design of a relationship-based access control model, called RPPM, which I introduce with the intention of addressing the limitations of existing models, and to accommodate richer types of access control policy.

My contribution to the body of knowledge is, therefore, the design of the RPPM access control model. RPPM is the first relationship-based access control model formally, and fully designed for general computing applications, whether they comprise one or more isolated, networked or distributed systems. I first introduce a base functional RPPM model and subsequently introduce three sets of enhancements which provide incremental developments to the fundamental workings of RPPM; these enhancements increase the expressiveness of the base model's policy language, as well as introducing optimisations, such as caching, and support for history-based policies. I then introduce several enhancements focused on applying RPPM to general computing scenarios: administration; and inter-operation. I demonstrate how all of these features may be consolidated into a single model which may then be applied to publish/subscribe architectures. Finally, I tailor this relationship-based publish/subscribe access control system to Internet of Things as this is a particularly topical and important application domain in need of security controls.

"[E]ducation is the silver bullet. Education is everything. We don't need little changes, we need gigantic, monumental changes. Schools should be palaces. The competition for the best teachers should be fierce. They should be making six-figure salaries. Schools should be incredibly expensive for government and absolutely free of charge to its citizens."

Sam Seaborn in The West Wing (#1.18) - Six Meetings Before Lunch [172]

"It is a capital mistake to theorise before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts."

Sherlock Holmes in The Adventures of Sherlock Holmes [70]

Dedicated to my loving and patient family;
my sincerest thanks.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

1.1 Overview

In modern computing there are very few computer systems where every user of that system is required to be able to perform all possible actions on all possible resources. More usually there is a need to selectively limit the actions which can be performed on resources; access control is the security service which provides this capability. The reasons why such limitations are required vary depending on the computer system. There may simply be a distinction between configuration (administrative) functions versus operation (user) functions, or between a write mode and a read mode. There may be a need to isolate the data belonging to each individual user from every other user, or there may be more complex requirements based on concepts such as security clearance level, job role or previous activity. Whatever the reason, access control's purpose is to allow authorized access requests whilst denying unauthorized requests, whether from an external actor (or non-system device) or from a legitimate system user (or system device).

In many situations, access control is policy-based: interactions between users and resources are modelled as "requests" and the policy specifies (either implicitly or explicitly) which requests are to be granted and which denied. An access control system is based on an access control model, which will define the data structures used to specify an access control policy, and an algorithm, to determine whether a request is authorized by a given policy. Traditional authorization policies are user-centric, in the sense that authorization is defined, ultimately, in terms of user identities. However, I believe that this user-centric approach is inappropriate for many applications, and that what should determine authorization are the relationships that exist between entities in the system. As is customary in the literature I will use the terms *subjects* and *objects* when referring to the parties who are to,

respectively, perform and be the target of authorization (inter)actions.

In this thesis I present my contribution to the body of knowledge – a relationship-based access control model designed for general computing applications. To a crude approximation my model takes inspiration from three sources: the overall design of the decision algorithm is similar to UNIX; the path conditions (sequences of relationships which are matched during policy evaluation) are similar in spirit to some of the existing proposals for relationship-based access control; and the use of implementation-specific authorization principals bears some resemblance to RBAC. However, I believe my path conditions provide a more rigorous foundation for access control mechanisms than existing proposals for relationship-based access control. I also believe my use of authorization principals provides highly desirable abstraction and scalability properties. Additionally, my model is generic, thus able to describe systems of various forms, be they social networks, IT systems (singularly or as networks) or entire businesses.

1.2 Brief Motivation

There have been numerous access control models defined since the topic first attracted interest in the 1960s. I provide a detailed review of many of these models and their underlying principles in Chapter 2, along with indication of existing issues which motivate my work in Chapter 3. For now I provide a brief motivating introduction as part of orientation to this thesis.

One of the earliest access control models was the protection matrix [123]. The protection matrix model simply enumerated all authorized actions. This is a conceptually simple approach; however it is inefficient when dealing with more than a few subjects and objects. New models have since been introduced with the intention of addressing limitations in existing models, or to accommodate richer types of access control policy.

A prime example of this is the role-based access control (RBAC) model which allows permissions to perform actions to be granted to job roles [79]. Subjects are assigned to their applicable roles and thus gain the permissions to perform the necessary actions for that role. The RBAC model offers several significant benefits over previous models. In particular, it reduces the administrative burden of managing the access control system by abstracting policy assignment away from subjects to roles; additionally, it is conceptually simple, thereby being easily understood and implemented. It is principally for these reasons that it (or some close variant) has become so widely utilised in modern computing systems. Since RBAC's inception

there have been numerous variations and extensions suggested to adapt it for specific applications. These extensions have included, for example, support for role hierarchies, as well as geographical and temporal constraints [30, 67].

More recently, alternative models have been growing in popularity, with attribute-based access control (ABAC) [109] and usage control (UCON) [143] receiving particular attention. All of these models assume that authorization should, essentially, be based on user attributes (particularly user identities). However, in many computing systems it is not the individual that is relevant to the access control decision, but the relationship that exists between the individual requesting access and the resource to which access is requested. Consider, for example, a request by a user u to read the records of a patient p. The fact that u is a doctor is a necessary, but not sufficient, condition for access to be granted. Specifically, u should be one of p's doctors. A second example arises when the same user may occupy different roles in different contexts. A PhD student, for example, may be an enrolled student on course c_1 and a teaching assistant for course c_2. Clearly, a request to read the coursework of another student should be disallowed if the coursework is for course c_1 and allowed if for course c_2.

Whilst parameterized variants of RBAC are able to bundle the context into the role [88], this often leads to a proliferation of roles as each specific context must be "identified". As the number of roles tends towards the number of users this undermines RBAC's reduced administrative burden. Access control languages based on first order logic or logic programming can express complex access control policies that can deal with such situations [22, 94]. However, this comes at the cost of complexity, both for end users that have to specify policies and in terms of policy evaluation.

A new paradigm, known as relationship-based access control, has emerged, particularly to address access control in online social networks [44, 82]. In this thesis, I extend relationship-based access control to arbitrary computing systems. I provide a richer policy framework than RBAC, taking relationships into account, while retaining conceptual simplicity. However, I also exploit features of RBAC and UNIX to provide a scalable and intuitive policy language and evaluation strategy. I introduce the concept of a path condition, which is used to associate a request with a set of security principals at request time. The security principals are authorized to perform particular actions. Thus, at a high-level a security principal is analogous to a role.

As indicated, my model takes its inspiration from the UNIX access control model, RBAC and existing work on relationship-based access control. However, it provides

a much richer and more flexible basis for specifying access control policies than any of these models. In particular, it provides arbitrary flexibility in the definition of principals, unlike UNIX; it supports policy specification based on relationships, unlike RBAC; and it provides policy abstraction (based on principals) and support for general-purpose computing systems, unlike existing work on relationship-based access control (which has focused on social networks).

1.3 Contributions

In this thesis I introduce RPPM, a novel relationship-based access control model designed for general computing applications, which:

1. Is able to model a wide range of systems, including online social networks which originally motivated relationship-based access control, due to its support for any entity and relationship types required.

2. Can be configured to guarantee that a conclusive authorization decision (approve or deny) can be computed for any request.

3. Can control actions requested by any type of entity on any other type, such that subjects may be users, autonomous entities, automated agents or even inanimate objects if so desired.

4. Abstracts permission assignment away from subjects to principals, reducing the administrative burden and enabling the system graph to be managed in isolation to the policies.

5. Employs flexible policy rules which support the definition of policies covering individual entities, multiple specific entities, all entities of one (or more) types, or every entity in the system.

6. Can define set-based or graph-based policies comprising multiple rules, where each can employ a required and forbidden path of relationships built with regular expression-like operators to support concatenation, disjunction, conjunction, Kleene plus, optional and Kleene star.

7. Is able to cache the result of principal-matching directly within the system graph such that future requests between that subject and object (no matter what action is requested) may be processed far faster.

8. Is able to support useful policy configurations such as history-based access control, separation of duty, binding of duty and Chinese Wall.

9. Can match principals based on paths of relationships between arbitrary entities, thus enabling subgraph patterns to be used to evaluate requests.

10. Can be used to completely administer instances of itself without extra-model authorization.

11. Can evaluate both local and remote requests, such that subjects may interact with objects in the local security domain or a remote security domain to which their system is connected.

12. Can implement a range of access control models not based on relationships, specifically: multi-level security; RBAC; UNIX; and multi-level security's *-property.

13. Can consolidate all of these features to produce a robust model for access control which can be further tailored to popular communication architectures.

1.4 Thesis Structure

The body of this thesis describes the features and workings of RPPM, a relationship-based access control model built upon the concepts of relationships, paths, and principal matching. Following a systematic approach commonly used in the field of access control [3, 49, 58, 144, 164, 165], I present a basic functional RPPM model (which I will refer to as RPPM_0) and subsequently introduce groups of enhancements to define more robust and capable RPPM models; the inter-dependencies of these models are shown in Figure 1.1, where directed arrows indicate the flow of functionality from the base model, through various models which depend on it, to more functionally-rich models (i.e. the head of each arrow points to the dependent model).

This systematic approach allows the model enhancements to form a natural progression through the course of the thesis. However, the enhancements I will present take two forms which I choose to highlight. The first three enhancements provide incremental developments to the fundamental workings of RPPM, whilst the latter enhancements are focused on supporting RPPM's ongoing use in general application domains. In order to provide clear discussions for these two distinct aspects I have structured the body of this thesis in two, corresponding, parts. However, prior to that I introduce the topics relevant to this thesis and my motivation for developing RPPM. The complete structure of this thesis is, therefore, as follows.

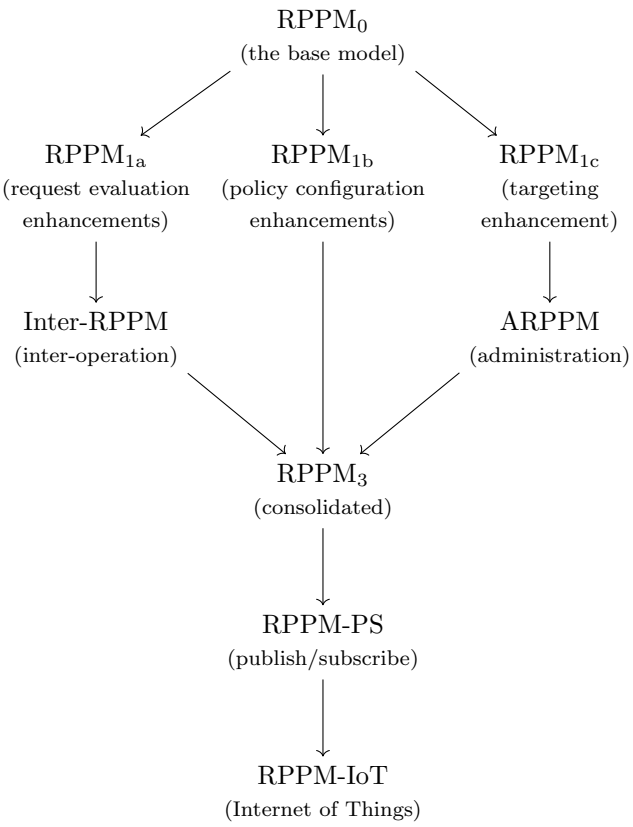


Figure 1.1: RPPM model dependencies

Chapter 2 – Background I provide a history of the significant developments in the field of logical access control, from its inception in the 1960s through to present day. Specifically, I provide an introduction to core security concepts relevant to understanding the design and implementation of computer protection systems, and discussions of the evolution of the most significant access control models: the access matrix; multi-level security (MLS); role-based access control (RBAC); attribute-based access control (ABAC); and relationship-based access control (ReBAC), on which this thesis is based.

Chapter 3 – Motivation I provide a snapshot of the current norm with regard to access control within modern operating systems and web applications. I then discuss a range of existing issues with the models commonly employed, highlighting the models' features and limitations whilst providing insight into the way in which models were intended to resolve the issues of their predecessors. From these issues I motivate the development of RPPM, a relationship-based access control model suitable for general computing applications.

Part I – RPPM's Fundamentals Part I discusses the basic functional model RPPM_0, and three enhancements, RPPM_{1a}, RPPM_{1b} and RPPM_{1c}, which independently develop the model's fundamental workings. Whilst these enhancements are discussed independently, they may equally be employed together in some combination. Due to the distinct nature of these enhancements their combining leads to no additional interaction and so requires no specific handling.

Chapter 4 – RPPM_0 I introduce a base RPPM model (RPPM_0) which is able to evaluate authorization requests for a modelled system; RPPM_0 uses a two step evaluation process which first matches security principals to the request, and second determines whether those principals are authorized to perform the requested action. I compare RPPM_0 to other existing relationship-based access control models to illustrate how it is more broadly applicable and less constrained. I also demonstrate how RPPM_0 can be used to implement several popular access control models which are not based on relationships: multi-level security and RBAC.

Chapter 5 – RPPM_{1a} I introduce three *request evaluation enhancements* to the base RPPM model to produce the RPPM_{1a} model. The first of these enhancements, policy graph evaluation, increases the policy language's expressiveness by allowing RPPM_{1a} to support list-oriented policies, conjunction and principal activation. The other two enhancements, target-based request evaluation and caching edges, provide optimisations of the request evaluation process (specifically the most complex, compute principals step). I compare these enhancements to other access control models and demonstrate how RPPM_{1a} can be used to implement the UNIX access control model.

Chapter 6 – RPPM_{1b} I introduce audit edges to the base RPPM model to produce the RPPM_{1b} model. With these edges RPPM is no longer "memory-less" with regards to request evaluation. Specifically, using audit edges RPPM_{1b} can support four common *policy configuration enhancements*: history-based policies; separation of duty; binding of duty; and Chinese Wall. I compare these enhancements to other access control models and demonstrate how RPPM_{1b} can be used to implement multi-level security's *-property.

Chapter 7 – RPPM_{1c} I introduce path expressions as a *targeting enhancement* to the base RPPM model. Path expressions enable RPPM_{1c} to match principals based on paths of relationships between arbitrary entities within the modelled system,

rather than just between the subject and object of the request. Through careful definition of path expressions RPPM_{1c}'s policies can, further, match principals based on (limited) subgraph patterns. I compare this enhancement to other relationship-based access control models.

Part II – Applying RPPM Part II discusses several enhancements focused on applying RPPM to general computing scenarios. ARPPM is built incrementally on RPPM_{1c} to provide the administration support which is important for any ongoing use. In turn, Inter-RPPM is built incrementally on RPPM_{1a} to enable authorization decisions to be made across multiple distinct security domains. I combine all of the RPPM functionality described to this point to produce RPPM_3, and then make use of this model as part of a relationship-based publish/subscribe access control system called RPPM-PS. Finally, I consider how RPPM-PS may be tailored to an Internet of Things.

Chapter 8 – ARPPM I define a set of administration requirements which I require any administrative RPPM model to satisfy. I then introduce *administrative enhancements* to the RPPM_{1c} model to produce the ARPPM model. These enhancements include four minor request and policy changes as well as an initial system graph state. Together these changes enable ARPPM to meet the administration requirements defined previously. I compare these enhancements to other access control models.

Chapter 9 – Inter-RPPM I introduce two *inter-operation enhancements* to the RPPM_{1a} model to produce Inter-RPPM. The first provides the means by which distinct system graphs may be connected together, whilst the second introduces the request evaluation processing necessary to allow system graphs in Inter-RPPM to evaluate remote requests in addition to the existing local ones. I briefly compare these enhancements to other access control models; inter-operation has not previously been considered in the case of relationship-based access control.

Chapter 10 – RPPM-PS I provide an introduction to publish/subscribe systems as a specific approach to communication in large, distributed information dissemination environments. I combine the RPPM functionality so far described in this thesis into a single RPPM_3 model and then demonstrate how this can be used as the basis for a relationship-based access control model (RPPM-PS) tailored to publish/subscribe systems.

Chapter 11 – RPPM-IoT I introduce Internet of Things as a topical and important application domain within which the publish/subscribe approach is being applied. I then discuss how RPPM-PS may be tailored to provide access control for publish/subscribe systems in an Internet of Things to produce RPPM-IoT.

Chapter 12 – Conclusions I revisit my motivations as part of reviewing my contribution to the body of knowledge. I also introduce a range of topics for future work with the hope of stimulating further discussion in the research community.

1.5 Contributing Published Works

This thesis is underpinned by research conducted between May 2013 and April 2017, much of which has been published during that period in the following works:

[62] Jason Crampton and James Sellwood. Path conditions and principal matching: A new approach to access control. In Sylvia L. Osborn, Mahesh V. Tripunitara, and Ian Molloy, editors, *19th ACM Symposium on Access Control Models and Technologies, SACMAT 14, London, ON, Canada - June 25 - 27, 2014*, pages 187-198. ACM, 2014.

[60] Jason Crampton and James Sellwood. Caching and auditing in the RPPM model. In Sjouke Mauw and Christian Damsgaard Jensen, editors, *Security and Trust Management - 10th International Workshop, STM 2014, Wroclaw, Poland, September 10-11, 2014. Proceedings*, volume 8743 of *Lecture Notes in Computer Science*, pages 49-64. Springer, 2014.

[61] Jason Crampton and James Sellwood. Caching and auditing in the RPPM model. *CoRR*, abs/1407.7841, 2014. Longer version including proofs.

[63] Jason Crampton and James Sellwood. Relationships, paths and principal matching: A new approach to access control. *CoRR*, abs/1505.07945, 2015.

[64] Jason Crampton and James Sellwood. ARPPM: Administration in the RPPM model. In Elisa Bertino, Ravi Sandhu, and Alexander Pretschner, editors, *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, CODASPY 2016, New Orleans, LA, USA, March 9-11, 2016*, pages 219-230. ACM, 2016.

[65] Jason Crampton and James Sellwood. Inter-ReBAC: Inter-operation of relationship-based access control model instances. In Silvio Ranise and Vipin

Swarup, editors, *Data and Applications Security and Privacy XXX - 30th Annual IFIP WG 11.3 Conference, DBSec 2016, Trento, Italy, July 18-20, 2016. Proceedings*, volume 9766 of *Lecture Notes in Computer Science*, pages 96-105. Springer, 2016.

1.6 Acknowledgements

There are two individuals to whom I am particularly grateful and without whom I would have not enjoyed this research so greatly, learnt so much, and produced this thesis so soon. Professor Jason Crampton has been a fantastic guide to my research escapades and has been kind enough to offer his time and significant expertise to enlighten me in the techniques of academic research and scientific writing. (I look forward to further co-authoring opportunities.) However, without the support and patience of my wife I would have been unable to commit much of the time that this endeavour has required. The pair of you have helped me in so many ways that this thesis is as much a testament to your involvement as it is to mine.

During my studies I have been fortunate enough to attend a number of interesting conferences and I thank the Information Security Group at Royal Holloway, University of London for supporting my attendance to these. I have found the hosts and attendees of these conferences very welcoming and I am particularly grateful to Philip Fong and Silvio Ranise for their friendship and kind words of encouragement. I'm also grateful to Jaehong Park for a very enjoyable discussion about access control and academia in general.

Lastly, I would like to thank the ever present support and encouragement of my family. In particular I cannot thank my mother enough for all that she always does for me. I cannot count the number of times she has asked me how my studies are going during the last four years. It is her, and my father's, approaches of hard work and dedication which have always stood me in good stead. I relied upon those lessons throughout this journey, as I have in everything else I've ever done.

My thanks to you all.

I'd also like to thank my examiners, Michael Huth and Charles Morisset, who kindly took the time to review this thesis and to discuss it with me in my viva. Thank you both for your comments and suggestions.

Chapter 2

Background

This thesis will present the design of a relationship-based access control model tailored for general computing applications. An access control model provides a syntax for authorization policies and a specification of the algorithm used to evaluate requests. In this chapter I provide a history of the significant developments in the field of logical access control, from its inception in the 1960s through to present day. In so doing I highlight a range of concepts, models, and features which will be relied upon (and referenced) throughout the remainder of the thesis. Those readers with a broad knowledge of access control may choose to skip this chapter, referring back to specific sections only as they find necessary.

2.1 Early Multi-User Systems

2.1.1 The Reference Monitor

Since the advent of electronic data sharing, introduced by the multi-user computer systems of the 1960s and early 1970s, there has been a need to *protect*[1] stored information from *unauthorized* "use". A single-user computer system fully isolated from outside influences is both inherently more secure, and inherently less useful (but not useless) than one which enables data exchange and may be used by multiple individuals. However, given the social proclivity of humankind it is no surprise that the development of computer systems rapidly passed beyond isolation towards a more interactive and communicative paradigm. This trend continues to this day, with the increasing use of mobile devices and socially-interactive apps, even as flaws and security incidents receive growing attention from the mainstream media and the public alike.

[1]The term *protection* was, historically, used to refer to security techniques controlling the access of executing programs to stored information [162].

In the beginning, as multi-user computer systems began to develop, the three core concerns of information security (confidentiality, integrity, and availability) took shape, and a set of minimum requirements for secure multi-user systems was identified, I quote [7]:

a) A physically secure environment for the computer, and other physical elements of the system including terminals where present;

b) Control of access to the system;

c) An adequate method of internally isolating individual programs (users) simultaneously resident on the system; and

d) An adequate method of limiting access to programs and data files.

However, whilst physical access control was reasonably well understood at this time, the mechanisms for logical access control (necessary for the last two requirements) were receiving preliminary, but concerted, attention. From this early work came the concept of the *reference monitor*, a core component in a model for a secure computing environment designed to mitigate the threat of malicious users [6].

The reference monitor mediates all "references" (access requests) to programs or data which occur as part of a process running on the system. Each reference is validated according to the access granted the user executing the process. Thereby, the reference monitor enforces the wider protection system's defined protection policy, one request at a time. In order to ensure this critical role is performed securely, three operating principles were identified for the reference monitor, I quote [6]:[2]

a) The reference validation mechanism must be tamper proof;

b) The reference validation mechanism must always be involved; and

c) The reference validation mechanism must be small enough to be tested (exhaustively if necessary).

These principles help ensure that the reference monitor is able to consistently perform the role intended. The relevance of this design has led the reference monitor to be a key consideration of many new protection systems as they have been developed.

It is worth briefly noting that the operation of a reference monitor relies upon the successful authentication of the user and the integrity of various data, including the association of executing processes to the authenticated user identity. To this end, Creps identifies seven divisions of "*what* a secure system must protect", I quote [66]:

[2]The later two principles are clearly extended in Saltzer and Schroeder's design principles, discussed below.

- Security management – the management of security-relevant system attributes (e.g., subject/object classification changes, dynamic policy modifications).
- Secrecy – the prevention of information disclosure to unauthorized subjects (e.g., no read up, no write down, discretionary permissions, distribution caveats, aggregation issues, covert channels, emanations control).
- Integrity – the prevention of information modification by unauthorized subjects (e.g., no read down, no write up, domain separation and type enforcement, separation of duties, transaction-based process control).
- Availability – the assurance that system services and information will be available whenever needed (e.g., fault tolerance, reaction to attack, metrics defining acceptable levels of availability).
- Identification – the association of a system subject or object with its identification attributes (e.g., naming issues, uniqueness, mobility of identifiers, organizational identification schemes, organization head).
- Authentication – the assurance that a subject's or object's identification attributes are valid (e.g., mutual, multiple, staged, ongoing/continual).
- Audit – the logging of all security-relevant system events (e.g., events to be audited, information to be recorded and derived from audited events, real-time attack interdiction requirements).

Whilst I will not focus on the later five topics any further, they are relevant to protection systems of all kinds and are relied upon for the effective operation of access controls.

2.1.2 Secure By Design

By the early 1970s there were five main types of computing system protection available, with many systems in use still being *unprotected* [162]. Some (*all-or-nothing*) systems offered two ways of working, whereby information was either freely shared or was totally isolated. More significantly, the recent developments of that time had introduced *controlled sharing* systems, as well as those providing a protected subsystem within which *user-programmed sharing controls* could be employed. The first of these enabled the owner of a resource to control low-level access to it (such as read, write, and execute) in isolation to any other resource managed by the system. The second allows control over higher-level functions by abstracting the user from the resource, leaving them to call specific entry points which may enforce arbitrary

authorization decisions on the requested operation (much like a sandbox with predefined interfaces). Whilst these protection systems focused on restricting sharing itself, considerable interest in computer systems within the defence sector was focusing some attention on *putting strings on information* which would indicate how that information should be treated "after" its sharing (see Section 2.1.4).

Amongst the early work which produced these protection mechanisms, a broad range of factors impacting the effectiveness of such security controls was identified; disappointingly, many of these are still often overlooked, leading to today's commonplace security issues. As an example, Saltzer and Schroeder's seminal work from 1975 contains a multitude of insights [162], with the fundamentals of many modern protection schemes clearly visible in the topics discussed. Unfortunately, whilst these insights are regularly discussed and employed in theoretical or academic models, they are given less attention in commercial products. There are many potential reasons for this, but no doubt competing requirements, the complexity of certain implementations or environments, and a lack of awareness or understanding have contributed in many cases. Whatever the reason, numerous security violations do indeed occur, as Saltzer and Schroeder predicted, because of implementation flaws preventable through the appropriate use of their key design principles for protection systems:

- Economy of mechanism – which requires a simple design, thus reducing the likelihood of mistakes and increasing the chance of their discovery.
- Fail-safe defaults – which promotes a psychology of granting access where otherwise it is denied; an approach more noticeable than the inverse when incorrectly operating (whether through implementation or configuration).
- Complete mediation – which ensures that every access attempt is evaluated and caching of such evaluations' results is limited and discarded when invalid.
- Open design – which avoids reliance being placed on the secrecy of a mechanism whose distribution and use will, ultimately, likely result in its disclosure.
- Separation of privilege – which segregates cooperative keys to prevent total compromise of the mechanism from a single breach.
- Least privilege – which limits the repercussions from unintended operations.
- Least common mechanism – which reduces the routes for un-managed user interaction and minimises the number of parties for whom a single mechanism must be "acceptably" secure.

- Psychological acceptability – which enables regular, automatic use of a mechanism by users by aligning it with their frame of reference for such a control.

In particular the principles of "economy of mechanism", "open design", "least privilege", and "psychological acceptability" are commonly lacking in modern implementations (although not always together). As an example, Porter Felt *et al.* identified that developers producing apps for the Android mobile operating system commonly (35.8%) ignored the principle of "least privilege" by requesting unnecessary permissions [75]. In the defence of these developers, however, Porter Felt *et al.* identified that many of these extra permissions were likely requested as a result of documentation errors and a lack of understanding – suggesting an associated failing in the use of the "economy of mechanism" and "psychological acceptability" design principles by the creators of the Android platform and its documentation. Such failings are certainly the cause of some identified weaknesses in modern operating systems such as Android [168].

2.1.3 The Access Matrix

Whilst Saltzer and Schroeder's design principles resonate with stories of modern-day flaws, their discussion of list-based (e.g., *access control list* systems) and ticket-based (e.g., *capability* systems) protection mechanisms, used individually or in combination[3], also highlights concepts still relevant today. Both of these mechanisms developed from the concept of an *access matrix*, which was initially motivated by the need to provide a systematic way of identifying, and controlling access to, objects which are shared [123].

The rows in the access matrix (see Table 2.1, adapted from [123]) represent the protection domains (e.g., protection rings) within which a process may be operating, whilst the columns represent the objects which may be targeted by the process. The intersecting cells contain *access attributes*, which determine the privileges that a process from that domain (row) is granted on that object (column).[4]

A domain may grant (add) an attribute on any object to another domain if it has the "control" attribute for that domain's object (e.g., Domain 1 may add attributes for any object to Domain 2 in Table 2.1, because the access attributes for Domain 1 (row) on the Domain 2 object (column) include the "control" attribute). In addition, a domain may grant (copy) an attribute on an object to any other domain if its own access attribute for that object contains the attribute in question marked with the

[3] Saltzer and Schroeder note that "[m]ost real systems use a combination of these two forms, the capability system for speed and an access control list system for human interface" [162].

[4] The close association with the row and column header labels and the access attributes means that this type of approach has been referred to as *Identity-Based Access Control (IBAC)* [105].

copy flag, represented by * (e.g., Domain 3 may copy its "owner" attribute for object File 2 to either of the other domains). Finally, a domain may revoke (remove) an attribute on an object from another domain it is has the "owner" attribute for that domain's object.

	Domain 1	**Domain 2**	**Domain 3**	**File 1**	**File 2**	**Process 1**
Domain 1	*owner control	*owner control	*call	*owner *read *write		
Domain 2			call	*read	write	wakeup
Domain 3			owner control	read	*owner	

Table 2.1: Portion of an access matrix

List-based and ticket-based protection mechanisms rely on different "viewpoints" of the access matrix, as illustrated by Figure 2.1. An access control list (ACL) for an object is formed by taking the `(domain name, access attribute)` pairs for an object column and "attaching" them to the object in question. Equivalently, a capability for a domain is formed by taking the `(object name, access attribute)` pairs for a domain row and associating them with the relevant domain.

	File 1
Domain 1	*owner *read *write
Domain 2	*read
Domain 3	read

(a) Access control list for File 1

	Domain 1	**Domain 2**	**Domain 3**	**File 1**	**File 2**	**Process 1**
Domain 2			call	*read	write	wakeup

(b) Capability for Domain 2

Figure 2.1: Access matrix viewpoints

The key difference that results in these mechanisms relates to the manner in which an authorization request is evaluated by the reference monitor. With list-based systems, an authorizing check is performed by the reference monitor when a process requests to use a particular privilege on an object. The reference monitor searches the object's access list and determines whether the identified process may be granted the requested access. (This approach is akin to a doorman checking whether a visitor is on the guest-list for an event.) In contrast, in a ticket-based

system, the process selects and presents the relevant ticket from its capability list when requesting the access. In this instance the reference monitor's authorizing check simply validates the presented ticket. Operationally, evaluating a list-based system is principally a search problem (assuming the access list is trusted), whilst evaluating a ticket-based system is principally a validation problem (to ensure invalid capabilities are not accepted). In order to make the most of both approaches, hybrid mechanisms may be employed (as noted by Saltzer and Schroeder [162]).

Considering the differences between these mechanisms further, it can be desirable to answer two questions in respect of a protection system: "what can a specific subject do?" and "which subjects can do things to a specific object?" (where I use the term *subject* here, as is usual, to refer to the entity which makes an authorization request, be that a user or a process working on their behalf within a security domain). Whilst answers for both of these questions can be determined from the access matrix, access control lists and capabilities can each readily answer only one of these questions. To determine "what can a specific subject do?" requires information about a specific domain's access attributes, and so is easily determined by looking at that domain's capability. In contrast, to determine the answer when access lists are employed involves parsing every object's access list to identify access attributes associated with the particular domain. The inverse is true when determining "which subjects can do things to a specific object?" as this requires information about a specific object's access attributes. This is easier to determine when using a list-based system than when using a ticket-based system.

Whilst the ability to answer such questions is not usually the most significant factor in selecting a protection mechanism, it is important to understand that the ultimate selection impacts the ease of auditing, as well as the function, of the system protected. As I will discuss, there have been numerous access control models offered since the access matrix. However, its uncomplicated approach means that it continues to be heavily relied upon (often from the viewpoint of an access list) to manage fine-grained control of access to individual objects. Whilst this is often in conjunction with a more versatile access control model, it highlights the simplicity and longevity of the access matrix approach.

2.1.4 Mandatory Access Control

The fact that the access matrix model may be enforced from two different perspectives (list-based and ticket-based) is not unique; other aspects of access control can, equally, be considered from multiple viewpoints. Another such aspect, identified

during the early work on protection systems, is the distinction over whether protection policies are: unchangeable for an object and must be satisfied throughout its life (i.e., mandatory); or dynamic and modifiable by the object's owner or other authorized party (i.e., discretionary) [36]. Commonly, this distinction is extrapolated such that *Mandatory Access Control (MAC)* systems are assumed to involve a centrally defined policy (possibly managed outside of the access control model which enforces it) whilst *Discretionary Access Control (DAC)* systems are assumed to be decentralized. Whilst this extension isn't fundamental to either definition, it is a logical corollary.

It should be clear from the previous discussion that the access matrix is an example of a DAC system. It enables domains to make modifications to objects' access attributes based on the presence of the attributes "control" and "owner", or the presence of a copy flag. These modifications, where authorized, may be made at any time through the life of the target object.

In contrast, the defence sector provides the most commonly cited example, even considered synonymous by some [69], for MAC; security labels are employed to limit the disclosure of labelled (or classified) objects to subjects which have at least as high a (clearance) label [24]. Once created and labelled, the labels of subjects and objects are not changed and, thereby, the policy which applies to them doesn't either.

Early developments of secure computer systems in the defence sector by Bell and LaPadula had at their core this fixed, mandatory protection policy. They focused on preventing unauthorized disclosure by ensuring that the initial, and all future, states of the protection system were secure and devoid of compromise [24]. In these (multi-level) secure systems of the 1970s, a (direct) compromise was identified by the allocation of an object to a subject which had a lower classification (a breach of what was later known as the *simple security property* [25]), or which did not have the necessary need-to-know categories to match those of the object (enforced through the inclusion of a discretionary security mechanism). In order to protect against potential future compromise (rather than direct compromise) the **-property* was also included, ensuring that a subject could not (intentionally, or otherwise) write to an object of one classification, the data it was reading from a higher classification object [23].

Bell and LaPadula's approach later became formalised within Division B and Division A of the US Department of Defense's Trusted Computer System Evaluation Criteria (TCSEC) [69], frequently referred to as the Orange Book. These higher security divisions required evaluated systems to enforce mandatory access control

policies, and supporting policy elements, in conjunction with the fine-grained discretionary access control policies required of the (lower) Class C2 systems.[5] This arrangement should not be viewed as indicating that mandatory access control is more secure than discretionary access control. A comparison of security cannot easily be made at such an abstract level. Instead, it is indicative of the view within the defence sector at the time that mandatory protection, as used to implement security classification levels, was an additional, centrally-defined protection required on top of discretionary protection, which limited users' access but which users could "pass on" to each other [69].

Whilst the work of Bell and LaPadula focused on preventing unauthorized disclosure, Biba placed greater focus on ensuring the integrity of information using mandatory (and discretionary) protection mechanisms. Specifically, he introduced integrity levels, alongside the existing security levels, such that modifications could only be made to objects with equal or lower integrity, thus preventing malicious sabotage [36]. This approach prevents writing up to a higher integrity level in a similar way that Bell and LaPadula prevent writing down to a lower security classification level.

Clark and Wilson would later also focus on integrity rather than disclosure, although they moved away from military systems and instead highlighted the significant role that integrity requirements have within commercial systems [54]. Rather than implementing a policy associated with levels of security, their mandatory policy targeted two mechanisms "at the heart of fraud and error control: the well-formed transaction, and separation of duty". Well-formed transactions ensure internal consistency within the system and are enforced by limiting data manipulation to a small number of inspected programs, and by logging (for auditing purposes) all modifications made. In contrast, separation of duty ensures external consistency with the real world by preventing a single party from maliciously manipulating internal checks and balances in order to reflect a false representation of reality. Clark and Wilson's consideration of commercial systems in 1987, and their introduction of separation of duty were key parts of a shifting of focus away from multi-level secure systems toward alternative models of access control.

[5] The distinction between Division B and Division A being that to meet Division A's requirements a system must be able to be formally verified.

2.2 The Introduction of Roles

2.2.1 Role-Based Access Control

As the adoption of commercial multi-user computer systems grew significantly through the late 1980s and early 1990s, several issues became clearer. The concept of grouping users or processes in some way had been introduced during earlier research [123, 162], and this idea had been further developed and implemented in versions of the Multics [161] and UNIX [155] operating systems. Subsequently, this approach had also been included in the discretionary access control requirements of TCSEC [69]. However, it was the increasing commercial adoption which cemented the reality: administration of access control policies would be a significant bottleneck if each user of the system had to be individually assigned permissions to access any of the system's objects.

Not long after, the idea that commercial systems would solely implement DAC was challenged. TCSEC's definition of MAC had not focused on the mandatory nature of the policy being enforced, but had instead focused on how it could be used to restrict access to objects based on their sensitivity. As such, mandatory controls were thought of more as being appropriate to the defence sector, rather than more broadly as being appropriate to any situation requiring consistent, system-level policies. However, in 1992 Ferraiolo and Kuhn identified that there were key commercial access control policies which were not discretionary because they were associated with some legal, ethical, regulatory, best practice, or business-owned data management policy. As such, the user should not be in a position to pass on their access to others at their own discretion. Whilst such policies are mandatory, in order to avoid confusion with the association of MAC and the military they described a need for a *non-discretionary* form of access control in the commercial sector [79].

The non-discretionary access control model that Ferraiolo *et al.* proposed for commercial systems was the *Role-Based Access Control (RBAC)* model [77, 79]. In the RBAC model, *roles* are granted *permissions* to perform *operations* on *objects*. Users are then assigned as members of roles, and by activating a subset of these roles within a *session* a user is able to utilise the activated roles' permissions in order to act on an object (as shown in Figure 2.2, adapted from [80]). This design enables a non-discretionary (i.e., mandatory) protection policy to be defined through the granting of permissions to roles and the assignment of users to roles. Assignment to a role indicates a user meets the minimum *competency* to perform tasks appropriate to that role. Users are unable to arbitrarily "pass on" their privileges to other users, as only membership of the relevant role will grant access. In addition, the

abstraction of users behind roles was intended to reduce the administrative overhead of managing the access control model. Rather than having to assign permissions for each new user to access the system's objects, instead in RBAC the new user is simply added to appropriate roles.

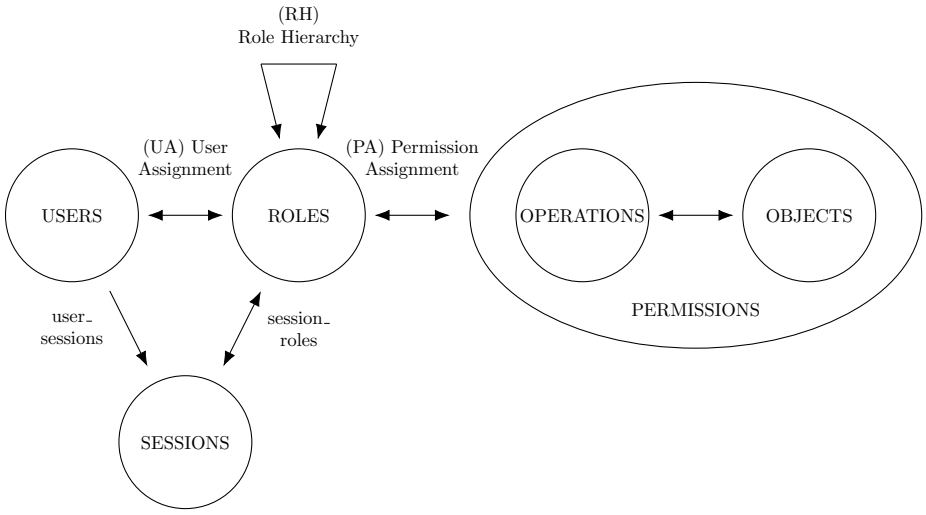


Figure 2.2: Hierarchical RBAC

Whilst these are the most distinctive, RBAC's features extend beyond its ability to enforce mandatory policies, and the abstraction of permission assignment away from users using roles. Access control models had previously solely focused on defining policies for low-level (file-system or memory) functions such as read, write and execute. By enabling the definition of access control policies for system-specific operations, RBAC is able to control more complex application interactions relevant to the roles and objects to which operations are granted. (These are referred to as *abstract permissions* in [163].)

Additionally, RBAC's support of role hierarchies offers a convenient way to further ease the administration burden using a partial ordering of roles.[6] Through the resulting hierarchy, permissions assigned to (junior, or less specialised) child roles are inherited by (senior, or more specialised) parent roles, and, thereby, may be utilised by members of those parent roles. This inheritance reduces the number of roles to which a user must be assigned; when made a member of a senior role, users gain the benefits of the more junior roles in the hierarchy implicitly.

[6]Role hierarchies were included as a core component of the original definitions of RBAC [77, 79] but, likely inspired by the partitioning of Sandhu *et al.* [164], later became an optional component, as did separation of duty, of the INCITS ANSI standard version of RBAC [108].

2.2.2 RBAC Administration

As shown in Figure 2.3, the features and components of the RBAC model (as previously discussed and illustrated in Figure 2.2) naturally define a sequence of management steps for the administration and use of an instance of RBAC.

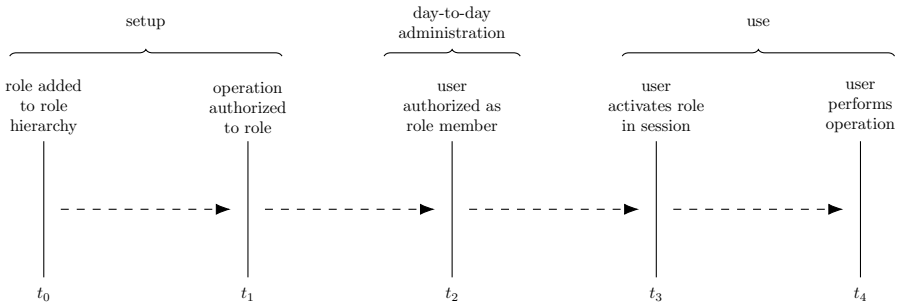


Figure 2.3: RBAC management and use time-line

The "earlier" steps (t_0 and t_1) enable the necessary setup of (part of) an RBAC model, with new roles added into the hierarchy and selected operations explicitly authorized to them. (As such, these steps are not performed frequently.) In contrast, day-to-day administration is focused around authorizing users to be members of particular roles (t_2). As users join or leave an organisation, their membership of appropriate roles must be granted or revoked. Further, as users change job function there is likely a need to change their associated roles to reflect their new duties. Hence this management step is more commonly performed than the initial setup of these roles within the model. Once the model is appropriately configured, in many environments authorized users interact with the system frequently, activating roles and performing operations on objects (t_3 and t_4) in order to complete their necessary business tasks.

In order to support all of these management steps, the RBAC model previously defined was supplemented, producing the administrative RBAC '97 (ARBAC97) model introduced by Sandhu *et al.* [163]. Given the fact that RBAC's own features help ease administration of access control, Sandhu *et al.* saw it as only "natural to ask how RBAC itself can be used to manage RBAC". The ARBAC97 model, therefore, contains administrative equivalents of several (regular) RBAC components from Figure 2.2. An administrative role hierarchy comprising administrative roles compliments the regular role hierarchy; in addition, specific administrative permissions compliment the regular permissions. Using these administrative components a user can activate administrative roles in a session, and, thereby, perform administrative operations on objects within the system.

In reality, the ARBAC97 model is actually a collection of three models [163]. RRA97 (role-role assignment '97) enacts the t_0 management step from Figure 2.3, whereby broad policy is set through the organisation of the regular role hierarchy. PRA97 (permission-role assignment '97) enacts the application administration relevant to the t_1 management step by controlling the assignment of permissions to roles. Finally, URA97 (user-role assignment '97) enacts the (more regular) personnel management of step t_2 by authorizing users' memberships of roles.

It is important to note that each of these models only controls the administration of regular RBAC model components. Administrative authorization requests are evaluated and if successful result in a change in the assignment of a regular role, permission, or user to a regular role. In contrast, ARBAC97 assumes that an extra-model chief security officer is available to manage the administrative components using some undefined external process.

The assignments of the ARBAC97 sub-models can be viewed as relations between the constituent model elements. These relations (whether associated with assignment or revocation operations) each identify an authorized administrative role within which the request's subject must be a member. They additionally identify the (regular) role(s) to which the request's object, be it a role, permission or user, may be assigned. In order to provide fine grained control over the use of administrative permissions, ARBAC97 also enforces *prerequisite conditions* on assignment relations within each of the three sub-models.[7] In its simplest, non-trivial form a prerequisite condition is a *prerequisite role* within which the request's object must already be a member. Less trivially, a prerequisite condition is a Boolean expression employing combinations of such (regular) roles and utilising negation, logical AND, and logical OR operators.

2.2.3 RBAC Constraints

Whilst prerequisite constraints enable fine grained control over the targeting of administrative operations, more generally RBAC supports the enforcement of a wide variety of policy configurations at different management steps, as shown in Figure 2.4 [77].

The principle of *least privilege* (as identified by Saltzer and Schroeder [162]) limits the repercussions from unintended operations by preventing users having access to privileges that they do not require to perform their job function. An administrator

[7] Prerequisite conditions are enforced for `can_assigna` and `can_assignga` within RRA97, for `can_assignp` within PRA97, and for `can_assign` within URA97. The equivalent `can_revokea`, `can_revokeg`, `can_revokep`, and `can_revoke` relations do not enforce prerequisite conditions.

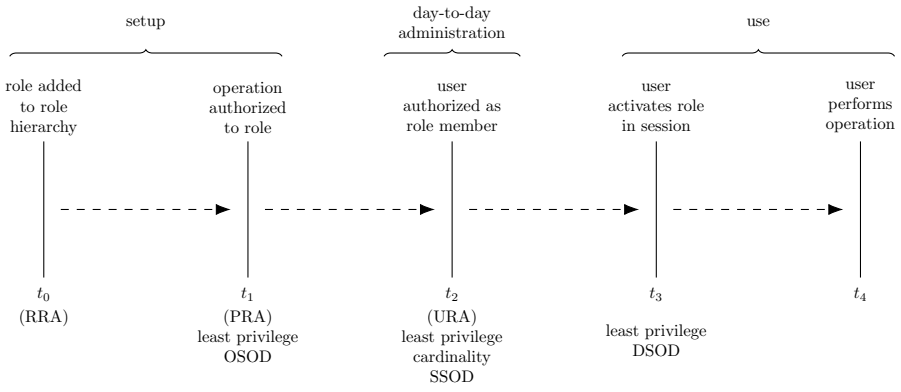


Figure 2.4: RBAC management and use time-line with constraints

can enforce least privilege through two mechanisms in ARBAC97 [163]. Firstly, the administrator limits the operations granted to roles (t_1) so that only the minimum necessary to perform the role are authorized to it. Secondly, the administrator limits the roles to which a user is authorized (t_2) so that they may not activate roles unnecessary for their job function.

A user may, further, support the principle of least privilege within each session (t_3), by only activating the roles that they require for the tasks they are about to perform. Whilst this last point will not protect the system from a malicious authorized user, it can protect the system from operations triggered by an authorized user unknowingly. Malware, such as a Trojan horse (which masquerades as a legitimate program), executed unwittingly by a user will frequently attempt to undertake more sensitive operations than those the user would commonly perform. By only activating the roles necessary for a session, the user reduces the likelihood of the malware's requested operations being authorized.

In order to prevent another form of over-privileged user, in some situations it is desirable to constrain certain roles using *cardinality* limits, such that the "capacity" of the role is not exceeded. This policy configuration is enacted when the administrator authorizes users to roles (t_2), ensuring that excess authorized users are removed before new ones are added. For instance, this may be used to limit the membership of particular management roles to a single user, or the membership of a process approver role to an easily audited number.

In contrast, there are several policy configurations which are concerned with preventing individual users employing multiple, conflicting roles as part of a business process. Such *separation of duty* constraints (as identified by Clark and Wilson [54]) are designed to limit authorized users from abusing multiple roles to their own gain. The common example given is one in which a business wishes to prevent individual

workers in a finance department from adding a new supplier, posting an invoice for that supplier, and authorising payment of that invoice. It should be clear that if a single individual could perform all these operations, they would be able to defraud the business to their own benefit.[8]

The most restrictive separation of duty constraint is *Static Separation of Duty (SSOD)*, which is enforced in ARBAC97 when users are assigned to roles (t_2) [163]. In SSOD an administrator is prevented from authorizing a user's membership of multiple conflicting roles. In this arrangement, the user is only ever able to fulfil a single (consistent) role from a set of conflicting roles (for example, they are only ever able to post invoices for existing suppliers). As an alternative, *Dynamic Separation of Duty (DSOD)* is enforced when users activate roles for a session (t_3). In this arrangement, the user is a member of multiple conflicting roles and can take on different roles at different times, but is unable to take on more than one during a single session. With an appropriate definition of a session, this enables a user to perform each of the operations during separate instances of the business process, but not in an arrangement which would enable fraud to occur. Finally, *Operational Separation of Duty (OSOD)* is a, less commonly identified, variant of the policy which relies on support for high-level, system-specific operations. OSOD is enforced when abstract permissions are granted to roles (t_1), with the constraint preventing all of an end-to-end process' system-specific operations from being assigned to a single role. OSOD may be less commonly used as this approach only works when SSOD is also employed. This is because the OSOD constraint is negated if a user may be assigned to multiple roles which together are authorized the full set of operations.

2.3 Generalised Attributes

2.3.1 Attribute-Based Access Control

Whilst RBAC was principally developed between 1992 and 2004,[9] another, now prominent, form of access control has seen a far longer period of refinement. *Attribute-Based Access Control (ABAC)* was first referenced towards the end of 1989, when it was proposed as a means of preventing (direct) unauthorized disclosure of information in military command, control and communications (C3) systems [66]. Since then research and standardisation activities have been varied (and

[8] Separation of duty prevents a single individual abusing such a process, it does not, however, prevent such abuse in the case of a conspiracy. Alternative n of m policies can require a minimum cardinality subset of the authorized users to undertake tasks within a business process. However, such policies are not readily supported by RBAC.

[9] Ferraiolo and Kuhn first proposed their non-discretionary approach in 1992 [79] and the INCITS ANSI standard for RBAC was published in 2004 [108].

intermittent), although a significant drive is well under way at present.

Where RBAC employs role membership as the specific basis for its access control decisions, ABAC instead supports the evaluation of whatever system-specific attributes are relevant. In this way, properties of the subject and object (and sometimes the environment) may be considered during a request's evaluation, with a subject's role membership just one viable attribute for consideration.

From a standardisation point of view there have been several distinct contributions, all made within the last 15 years. *eXtensible Access Control Markup Language (XACML)* was first introduced in 2003, undergoing several revisions before version 3.0 was released in January 2013 [141]. Whilst it doesn't use the term attribute-based access control specifically, the XACML standard provides a policy language for expressing access control policies, requests, and responses where authorization decisions are based on attributes of the subject and the resource (i.e., the object).

In 2013, the functional definition of another, less language-focused, ABAC model, called *Next Generation Access Control (NGAC)* was standardised [109]. In contrast to XACML, NGAC is a more architectural standard, concerned with the arrangement and interaction of components rather than the specific language employed during those interactions. Most recently, in 2014, a guide to the definition of ABAC and considerations for its deployment and use in enterprise environments was published [105].

From whichever viewpoint ABAC is considered, it is the attributes of entities within the system that are used to evaluate policy. When ABAC was first proposed, in 1989 and 1990, a number of attributes[10] were highlighted, including: unique identifier; security level; integrity properties; roles; caveats; nationality; and organisation [66, 126]. This collection of attributes was inspired principally by ABAC's consideration as an alternative to specific MAC or DAC approaches in use within the defence sector. Hence these attributes were selected based on considerations associated with implementing multi-level security and need-to-know constraints. Whilst only a small number of attributes was initially highlighted, in contrast to the existing models of the time ABAC was positioned as a general framework through which various policy approaches could be enforced. A significant benefit of such a general approach was believed to be that policy constructs (such as a lattice of security levels or a role hierarchy) did not need to be hard coded into the access control model. Even so, public research into ABAC initially faltered and, as discussed, RBAC instead was the focus of 1990s access control research.

[10] Recently Ahmed *et al.* presented a comprehensive classification of attribute types through which these and other attributes may be viewed and compared [3].

After the initial military focus, the approach of checking attribute values as part of the evaluation of authorization policies resurfaced a decade later, with applications motivated by the growth of the World Wide Web [112, 145]. Digitally signed attributes, however stored and communicated, were suggested as a trustworthy basis for distributed access control decisions across the Internet. However, it was not until 2002 that ABAC's use in decentralized access control was more formally and completely presented.

2.3.2 RT

Whilst referred to as a role-based trust-management framework, the RT model introduced in 2002 was identified as being "especially suitable for attribute-based access control" [125]. In defining RT, Li *et al.* proposed that any expressive ABAC system support:

a) Decentralized attributes – where entities assert attributes for other entities;

b) Delegation of attribute authority – where entities trust other entities' judgements about attributes;

c) Inference of attributes – where entities infer attributes from other attributes;

d) Attribute fields – where attributes may include values which constrain the attribute's use; and

e) Attribute-based delegation of attribute authority – where entities delegate to other entities, trusting them because of their attributes.

Whilst RT supports these features and embraces ABAC, in other ways its presentation is less forward-thinking. Somewhat confusingly, given that ABAC is more general than RBAC, Li *et al.* present the outcome of predicates on attributes through role membership (or lack thereof). This presentation of ABAC in the terms of RBAC, rather than the other way around, may well have been motivated by the dominating position that the RBAC model held within access control research at the time. Whatever the cause, there is no doubt that this resulted in RT being given the "role-based" part of its name.

However presented, RT enables credentials to be determined for a subject using local, and distributed, attributes evaluated using rules defined as logical expressions. In order to ensure that distributed attributes can be effectively used to infer local attributes, and to determine credentials, RT includes requirements for clear notation and naming across authority domains. Without such requirements even simple policies could not be defined.

As well as simple policies enabling the identification of credentials from individual attribute values, RT supports more complex arrangements, such as static separation of duty and *threshold* policy configurations. These are both achieved through the use of role-product operators applied to pairs of roles (whether local or distributed). (Recall that membership of such roles may be determined through the evaluation of attributes.) Whilst separation of duty requires agreement between two distinct entities, threshold structures, more generally, require k entities agree so that an inference can be made. In this way, threshold structures are a dynamic constraint, as they do not *a priori* fix how entities may participate in satisfying the constraint. This is reinforced by RT's use of two different operators, allowing threshold structures to distinguish between requiring multiple entities and multiple different entities – where the second more closely aligns to the concept of separation of duty.

2.3.3 XACML

Like RT, eXtensible Access Control Markup Language (XACML) provides a way for distributed, and locally stored, attributes to be evaluated when enforcing an authorization policy defined using logical expressions. In addition, the XACML specification focuses on defining: an XML-based policy language through which authorization policies, requests and responses may be communicated; the mechanisms through which such language elements are processed; and the architectural elements which participate [141]. The combination of these specification elements has led to the XACML standard being readily implemented by practitioners into products and systems.

To more fully support a distributed authorization architecture, XACML partitions functionality amongst architectural elements which may be distributed across distinct system components in a particular implementation (as illustrated in Figure 2.5, adapted from [141]).[11] Key amongst these architectural components are the:

- *Policy Administration Point (PAP)* – where the access control policy is entered into the system.
- *Policy Enforcement Point (PEP)* – where the authorization request is received and the authorization decision is enforced.
- *Policy Decision Point (PDP)* – where the access control policy is evaluated and an authorization decision determined.

[11] The XACML specification refers to these architectural elements as "actors".

- *Policy Information Point (PIP)* – where information relevant to policy evaluation is stored or collated prior to being used by the PDP.
- *Context Handler* – where authorization requests and decisions are converted between system-specific formats and XACML, and where information is routed from PIP to PDP.

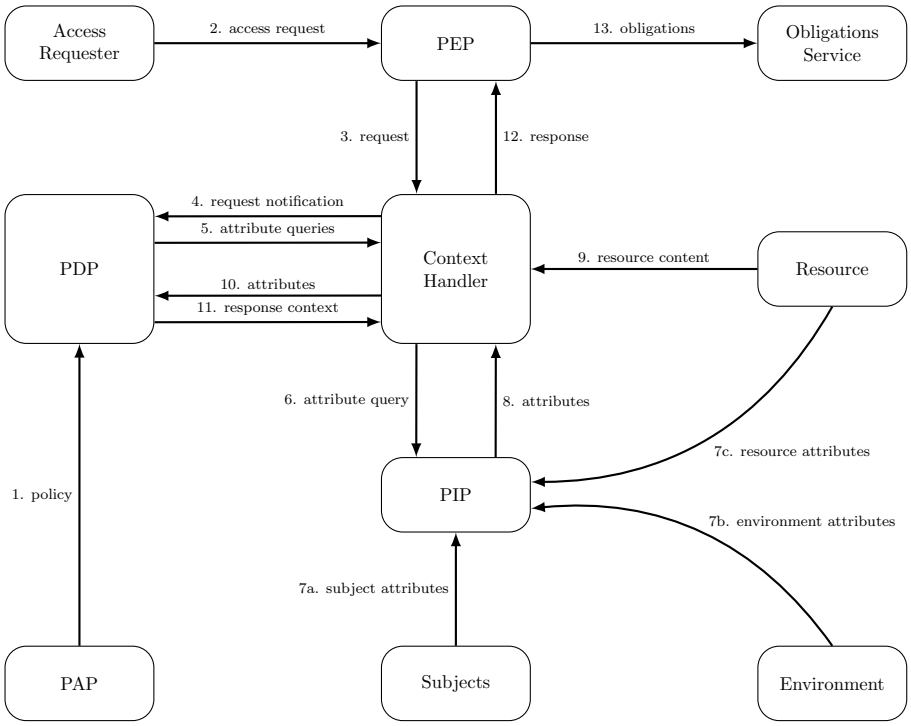


Figure 2.5: XACML data flow

The PxP architectural components have since been commonly referred to when discussing the compartmentalisation and distribution of the functions of authorization systems of all forms. However, these concepts were not original to XACML. The concepts of a policy decision point and policy enforcement point were originally defined in an RFC draft from 1997 [191], and the policy information point was defined in another such draft from 2000 [183]. Whilst not formally defined until XACML, the policy administration point was first referenced in 2000 [8].

For the various, potentially distributed, XACML components to work together effectively, clearly defined language elements and processing mechanisms are required. XACML's policy language is constructed, naturally, around *rules*, *policies*, and *policy sets*. Rules are evaluated, where they apply, by the PDP and their defined *effect* results if their *condition* evaluates to True. However, as individual rules cannot be communicated between components, sets of rules are brought together within a

policy. Grouping rules within policies also provides for greater management and evaluation options. To extend these options further, XACML allows policies to be grouped within a policy set (and policy sets to be further collected together in parent policy sets).

Each of the three language elements supports a *target* (a logical expression over attributes which determines the applicability of the element), *obligations* (which must be performed by the PEP in conjunction with the evaluation), and *advice* (which provides the PEP supplemental information about the decision, but which may be ignored) [141, §3.3]. The data flow of Figure 2.5 shows how the components work together to transport policy elements, the request, and relevant entity attributes to the PDP so that request evaluation can be performed and the policy elements evaluated. There are four decisions available at each level of evaluation [141, §5.53]:

- Permit – returned when the requested access is granted at that level.
- Deny – returned when the requested access is denied at that level.
- NotApplicable – returned when no policy applies to the request at that level.
- Indeterminate – returned when an error occurs whilst evaluating the policy at that level.

A rule will result in "Permit" or "Deny", as indicated by its effect, as long as its condition evaluates to True when evaluated. If the condition evaluates to False the rule's result is "NotApplicable"; this is also the case if the rule's target doesn't match and the condition is never evaluated. Any error when evaluating the target or the condition results in "Indeterminate".

In addition to a target, obligations and advice, policies also support *rule-combining algorithms* which identify how decisions and obligations from the contained rules should be brought together to produce single outcomes. The decisions which result from evaluating individual rules are combined within the containing policy to produce a decision there. Policy sets have equivalent algorithms, called *policy-combining algorithms*, which are used to combine results from evaluating multiple contained policies or policy sets. If the policy is itself contained in a policy set then this decision, along with those of the other contained policies, are combined (and so on).

XACML supports a range of combining algorithms for evaluating lists of policy elements [141, §Appendix C]:[12]

[12]In reality the "Indeterminate" decision is frequently extended to indicate whether the evaluation where the error

- Deny-overrides (ordered and unordered) – prioritises "Deny", if found in any result, over "Permit", "Indeterminate" and "NotApplicable".
- Permit-overrides (ordered and unordered) – prioritises "Permit", if found in any result, over "Deny", "Indeterminate" and "NotApplicable".
- Deny-unless-permit (unordered) – defaults to "Deny", but prioritises "Permit" if found in any result; it never leads to decisions of "Indeterminate" or "NotApplicable".
- Permit-unless-deny (unordered) – defaults to "Permit", but prioritises "Deny" if found in any result; it never leads to decisions of "Indeterminate" or "NotApplicable".
- First-applicable (ordered) – defaults to "NotApplicable", but prioritises whichever other result occurs first, if any do.
- Only-one-applicable (policy-combining only) – results in "NotApplicable" if no policy or policy set applies, results in "Indeterminate" if more than one policy or policy set applies, and where a single policy or policy set is applicable it reflects the result of evaluating that one.

Once an ultimate decision is determined by the PDP this is enforced by the PEP (as illustrated in Figure 2.5) along with any associated obligations.

2.3.4 NGAC

Whilst XACML has been well received, more recent ABAC research has led to the functional definition of Next Generation Access Control (NGAC). The first version of the INCITS ANSI standard for the functional architecture of NGAC was published in 2013, more recently a public review of a draft revision was held from 30th December 2016 to 28th February 2017 [110]. Even though both XACML and NGAC are standards defining authorization using attributes, there are various differences [78]. Not all of these differences favour one model over the other; rather, the two specifications focus on different approaches to defining an access control model as well as employing different design decisions.

From an architectural perspective, NGAC employs the PAP, PEP, PDP, and PIP elements utilised by XACML.[13] However, the PDP interacts with the PIP via the PAP, rather than through an additional component (such as XACML's Context

occurred could have resulted in a "Permit" decision ("Indeterminate{P}"), a "Deny" decision ("Indeterminate{D}"), or either ("Indeterminate{DP}"). To simplify the discussion I have assumed a single "Indeterminate" decision.

[13] Although it refers to them as "entities".

Handler). To the base PxP components NGAC adds one or more *Resource Access Points (RAPs)*, which are used to interact with protected resources upon access being granted, and zero or more *Event Processing Points (EPPs)*, which identify and coordinate the processing of obligations related to the access request. NGAC's arrangement of these components is shown in Figure 2.6 (adapted from [110]), as are the interfaces between them which are used during the processing of resource requests (R), administrative requests (A), and obligations (O).

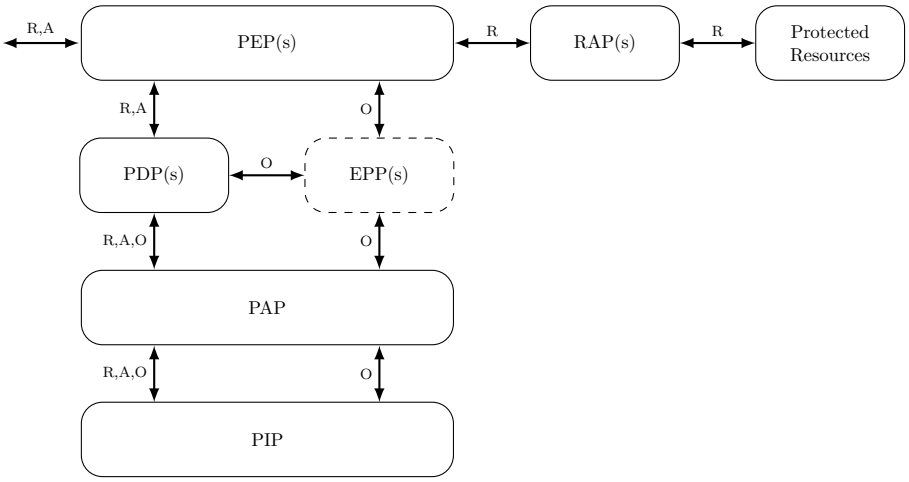


Figure 2.6: NGAC access information flows

Policies in NGAC are enumerations of relations rather than logic expressions. The policies can contain four types of *configured relations* [110, §6.2.4]:

- Assignment relations – which provide a hierarchy for policy elements (users, user attributes, objects, object attributes, and policy classes).
- Association relations – which identify access rights available to users in respect of a specific attribute.
- Prohibition relations – which identify access rights denied to users and processes in respect of specific attribute sets.
- Obligation relations – which modify policy information in association with events.

From these configured relations, four *derived relations* are calculated:

- Privilege relations (derived from association and assignment relations) – a permission given to users to perform specific access rights on policy elements.
- Capability relations (derived from privilege relations) – the capability associated with a privilege.

- Access control entry relations (derived from privilege relations) – the *Access Control Entry (ACE)* associated with a privilege, as would be present in an ACL.
- Restriction relations (derived from conjunction and disjunction over prohibition relations) – a permission denied to users and processes, preventing them performing specific access rights on policy elements.

Requests are evaluated in the PDP, with *privileges* in those policies positively supporting access. The actual granting of access, however, also requires there to be no relevant *restrictions* defined in the policy. Whilst user and object attributes are considered within policies, NGAC does not support evaluation informed by environmental attributes [78]. Once an authorization decision is made by the PDP, the PEP enforces this access decision.

If the request is granted and an EPP is present in the system then the authorization event will cause the EPP to coordinate the processing of any relevant obligations (using the interfaces indicated in Figure 2.6). The EPP triggers the PAP to identify relevant obligations within the PIP, which the PAP returns. Each relevant obligation is processed by the EPP, and the obligation's event response is passed to the PDP for execution. Whilst XACML does not prescribe what obligations may achieve, NGAC limits them to modifying NGAC policy information. Therefore, the PDP's execution of the obligation's event response ultimately directs the PAP to trigger a policy change within the PIP.[14]

Another significant difference between NGAC and XACML is the fact that NGAC supports specific administrative requests which are evaluated using the model itself (much like ARBAC97, described in Section 2.2.2).[15] Whilst resource requests are enforced by the PEP, in conjunction with a RAP, an administrative request in NGAC is enforced by the PDP passing the authorized instructions to the PAP, which in turn triggers policy changes within the PIP.

2.3.5 NIST's ABAC Guidance

Whilst the XACML and NGAC specifications define potential ABAC solutions, NIST's guide to ABAC (published in 2014) provides a general definition of the term attribute-based access control, and also documents considerations relevant to its use within large enterprises [105]. The definition provided in this guidance necessitates

[14]Interestingly, the processing of policy changes caused by obligations operates under the auspices of the user that created the obligation (not the one that triggered it). Therefore, the creating user must remain active in the policy configuration for the obligation to be operable.

[15]The topic of administration is not covered in the core XACML standard and is instead defined in a later, separate Administration and Delegation Profile specification [140].

the use of subject and object attributes, in conjunction with environmental conditions, during ABAC's request evaluation. (NGAC does not support environmental inputs, as I have mentioned, and so fails to meet NIST's definition.) However, like both XACML and NGAC the guidance assumes the same core architectural elements: the PAP; PEP; PDP; and PIP.[16] To these the NIST guide adds an optional Context Handler, similar to that of XACML, which executes workflow logic coordinating the evaluation and enforcement of policy.

The NIST guide identifies the human expression of a system's high-level authorization requirements as a *Natural Language Policy (NLP)* which is reflected within an access control model as a machine-enforceable *Digital Policy (DP)*. In turn, it identifies that the model's evaluation of DPs is managed using a *Metapolicy (MP)* which defines priorities and resolves conflicts. (XACML's combining algorithms provide just such a metapolicy element.) Within this conceptual arrangement, the NIST guide allows for rules to be defined through any appropriate computational language, such as a logical expression (as used in XACML) or an enumeration of relations (as used in NGAC).

Unlike the other ABAC standards, the NIST guide also considers the process through which an ABAC system may be deployed within an enterprise. To do so it focuses on all but the disposal phase of the NIST *System Development Life Cycle (SDLC)*, illustrated in Figure 2.7 (adapted from [105]).

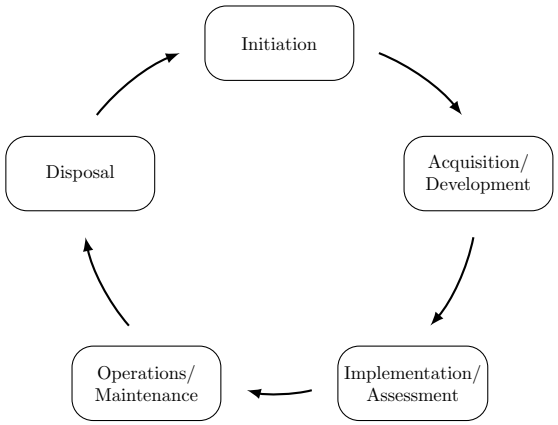


Figure 2.7: NIST system development life cycle

Within the *initiation phase* the guide highlights the need to: build a business case; identify scalability, feasibility, and performance requirements; and develop operational requirements. For the *acquisition/development phase* it identifies the need to: generate business processes; develop or acquire the system; and consider how it will be relied upon. In the *implementation/assessment phase* of the SDLC the

[16] Although this time they are referred to as "functional points".

guide indicates the need to consider: caching of attributes; minimizing the number of attribute sources; and clearly defining interface specifications. Finally, it identifies the availability of quality data as a consideration for the *operations/maintenance phase* of the SDLC.

2.3.6 Controlling Use After Access

Around the time that XACML was being standardised, an extension to access control was receiving attention and supposedly offering "a promising approach for the next generation of access control" [143]. (However, as seen in Section 2.3.4, the term next generation access control was ultimately co-opted for an attribute-based access control model.) *Usage Control (UCON)* combines the scopes of access control, trust management and digital rights management in order to, respectively, support the authorization of known individuals, the authorization of unknown individuals, and to perform client-side enforcement of usage. Whilst access control and trust management are focused on the control of access within the confines of the protection system, *Digital Rights Management (DRM)* is focused on maintaining control once the object in question has been disseminated to the subject. Given the existing demand for a reference monitor to mediate access requests, DRM introduces a specific requirement for a *Client-side Reference Monitor (CRM)*.

As it was developed when ABAC was first seeing application in distributed access control, UCON employs a simple form of ABAC for its access control component. However, it is the overall construction and function of UCON which are particularly noteworthy, rather than the fact that it makes use of an attribute-based access control model. The core components of UCON are highlighted within the ABC model, shown in Figure 2.8 (adapted from [165]), with *authorizations* (A), *obligations* (B), and *conditions* (C) determining the access and ongoing usage a subject may have over an object.

UCON's *rights* identify privileges a subject may action on an object; however, these rights are determined by a *usage decision function* when the access is attempted through the evaluation of attributes, authorizations, obligations, and conditions. Three different types of subject are identified: *consumer subjects*, who exercise rights to access objects; *provider subjects*, who provide objects and hold certain rights on them; and *identifiee subjects*, who are identified within objects which hold privacy-sensitive information. Whilst the research published focuses on consumer subjects, in reality the rights of related subjects (of all types) may be relevant to authorization requests on an object.

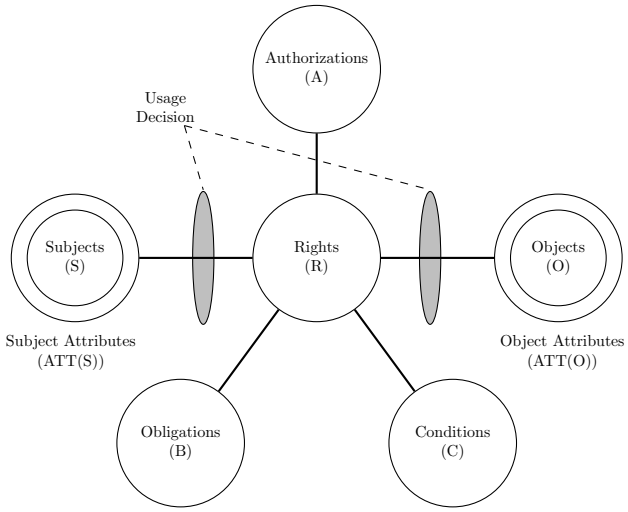


Figure 2.8: ABC model components

Firstly, authorizations determine whether the subject is allowed to perform the requested action on the object; these are determined by evaluating *authorization rules* using subject and object attributes. Authorizations may be evaluated before access is granted (as is normal for access control models), but may also be evaluated during usage (either continuously or periodically). An authorization may, additionally, require updates to some attributes of the subject or object, with these updates possible before, during, or after usage. UCON distinguishes *immutable attributes*, which cannot be changed by the user's actions on protected objects, and *mutable attributes* which can [144].

In contrast, obligations identify mandatory requirements which a subject must perform as part of an authorization. As with authorizations, these can be evaluated before and during usage. However, whilst obligations may require updates to the attributes of the subject, they cannot affect those of the object. Finally, conditions are environmental or system-oriented checks which must be satisfied. Whilst these may be evaluated before and during usage, they cannot affect attributes of either the subject or the object.

Whilst there are similarities between UCON and NGAC, it is important to note that common behaviour is achieved differently and common concepts behave distinctly. Specifically, whilst they both support updating attributes as part of the authorization process, in UCON this is simply a stated feature whilst in NGAC this requires the functionality provided by obligations. In contrast, obligations in UCON may determine initial (or ongoing) access, whilst in NGAC they are enacted subsequent to access being granted. More fundamentally, however, UCON is concerned with ensuring that a subject's ongoing access to an object continues to meet

the authorization policy, requirements placed on the usage, and the environmental conditions in which the use occurs. In so doing, UCON seeks to prevent authorized parties from abusing their access.

2.4 Relationships

2.4.1 Friend, of a Friend, of a Friend, of a Friend, ...

As the popularity of *Online Social Networks (OSNs)*[17] grew in the early 2000s, research into the sharing of resources between members of online communities developed. Such communities are less formally defined than commercial or military organisations, and are also far more dynamic in their make-up. Therefore, access control models grounded in relatively static and centralized concepts (such as security levels and roles) were not appropriate for these applications. Whilst initial work on ABAC was under way at the time, a more intuitive foundation was required and attributes were not believed to be appropriate for such situations [41]. This alternative grounding was identified, quite naturally, in *relationships* – a concept intrinsic to those applications.

The human race has developed and excelled as a social species, and this is reflected by the way that our brains develop and operate [55]. Whilst we rely on relationships with others (such as parents and teachers) for our own development, we also make significant use of the concept of relationships to comprehend and communicate abstract ideas to others. The use of relationships as the basis for describing (sharing) interactions is, therefore, intuitive and highly appropriate to access control.

Early work on access control in online communities focused on a single friend relation, and so introduced managed access to resources through the presence of friend and friend-of-a-friend relationships [120]. This concept was subsequently synthesized with trust relationships to create an access control model for social networks based on relationships [44, 45]. Carminati *et al.* represent a social network as a graph in which the edges are labelled by relationships (such as friendOf) and all nodes represent users. Each edge is also labelled with a trust value, indicating the "strength" of the relationship, as shown in Figure 2.9 (adapted from [44]).

Access Rules in their model determine the *access conditions* which must be satisfied for a user to be able to access the resources (not represented in the graph) of another user, the resource owner. Each access condition identifies at least one maximum length, minimum strength chain of relations of a specific type which must exist

[17] Within the access control literature social networks are also referred to as *Web-Based Social Networks (WBSNs)* and *Social Network Systems (SNSs)*. However, for consistency I shall limit myself to referring to OSN.

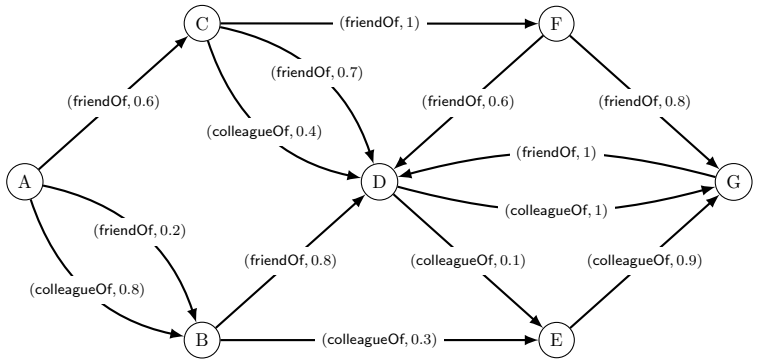


Figure 2.9: OSN subgraph

between the requestor and another user. This other user does not have to be the resource owner who defines the access condition, it can, instead, be any user within the graph, or a wildcard indicating no specific user is specified. Therefore, for example, a resource owner may require a chain of friendOf relationships with maximum length 3 and minimum strength 0.5 exist between the requestor and themselves, whilst also requiring a chain of colleagueOf relationships with maximum length 2 and minimum strength 0.8 exist between the requestor and a third party.

Around the same time, Carminati *et al.* produced an OSN access control model based on semantic web technologies [43]. Unlike the approach just described, their semantic web approach supports an OSN graph containing both users and resources. The OSN's users, resources, relationships and actions are modelled within an ontology to create a *Social Network Knowledge Base (SNKB)*. This enables automated processing and inference based on the hierarchy of semantics.

Security policies (for access and administration) in this semantic model are encoded using *Semantic Web Rule Language (SWRL)* and are stored in another ontology called the *Security Authorization Knowledge Base (SAKB)*. Thereby, authorizations and prohibitions may be defined for subjects attempting to act on objects. Specifically, the policy's *security rules* contain an antecedent and a consequent. Rule antecedents comprise a conjunction of atoms derived from the SNKB, and specified using *Web Ontology Language (OWL)*. Rule consequents contain either an authorization or prohibition for a particular action. Access requests are granted if the antecedents of applicable authorization security rules are satisfied and the antecedents of applicable prohibition rules are not satisfied.

The atoms supported within their security rule antecedents are:

- $\mathtt{C}(x)$ – which holds if x is an instance of the class or data range C.
- $\mathtt{P}(x, y)$ – which holds if x is related to y by property P.

- `sameAs`(x, y) – which holds if x and y are the same object.
- `differentFrom`(x, y) – which holds if x and y are different objects.
- `builtIn`$(r, x, \dots)$ – which holds if built-in relation r holds on the arguments x, ...

The first of these constructs enables rules to apply to specific resource types, e.g. `Photo(?targetObject)`, whilst the second can be used to require specific direct relationships exist, e.g. `Friend(?owner,?targetSubject)`.

Whilst these models enable relationships to be used to define access control policies, the policy languages are limited. Relationships are principally based on a single relation (whether a direct relation between two entities, or a chain). The semantic model does enable a rule to be constructed from multiple conjoined atoms referring to different direct relationships. However, specific identifiers or variables must be used for each object, and so this approach can only be employed for short paths which are pre-determined (i.e., passing through specific objects). More flexible relationship-based policies were needed to support the more diverse social network architectures envisaged.

2.4.2 Relationship-Based Access Control

In 2011, Fong introduced a *Relationship-Based Access Control (ReBAC)* model with a far richer policy language than those seen before [82].[18] Fong's work specifies a policy for each resource, where a policy is specified using multi-modal logic. The rationale for using a modal logic is that each relationship specifies an accessibility relation between users, which is used to provide semantics for the model's policies.

This richer language enables Fong to encode far more complex combinations of relationships into ReBAC's policies, thus allowing more diverse architectures and applications to be supported.[19] The policy syntax is specified by the grammar

$$\phi, \psi ::= \top \mid \mathsf{a} \mid \neg\phi \mid \phi \vee \psi \mid \langle i \rangle \phi$$

This grammar enables policies to be defined for direct relationships, where i is a relationship identifier, as well as chains of relationships.[20] The grammar also supports negation and disjunction, which in turn enable several other forms to be derived (including conjunction, derived from a combination of negation and disjunction).

[18] Fong *et al.* had previously generalised the Facebook access control model using a similar construction, but this was less well developed [83].

[19] Fong utilised an electronic health records case study in presenting this language.

[20] Within the grammar $\top$ is always true and a asserts the accessor is the owner.

Using this grammar Fong's policies can encode combinations of multi-relation paths between the accessor and either the owner or arbitrary third parties.

A social network in Fong's ReBAC model is a graph of binary relations where all of the vertices are users. Each resource (not in the graph) will have an owner, and will also have a *policy predicate*, formed from a modal formula, which returns a Boolean value if the resource's policy is satisfied between the accessor and the owner.[21] The *access context* of the social network is the configuration of the users and relationships. This context can be modified through a *state transition* which may add relationships (using the PUSH rule), remove relationships (using the POP rule), or replace relationships (using the EDGE rule).

Whilst the initial ReBAC work makes use of multi-modal logic, in 2012 Bruns *et al.* employ hybrid logic [41] to increase the language's richness further. The hybrid logic grammar, shown below, has some similarities to the multi-modal logic one. Direct and indirect relationships are supported in the same way, as is negation.

$$t ::= n \mid x$$
$$\phi ::= t \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle i \rangle \phi \mid \langle -i \rangle \phi \mid @_t \phi \mid \downarrow x\phi$$

There are several small changes introduced, conjunction is now a defined language component whilst disjunction is derived, and traversal against the direction of a relationship is also defined through $-i$.[22] More significantly, by using hybrid logic this revised version of ReBAC may bind variables within policies, $\downarrow x\phi$ binds the current node to variable x, and may "jump" to specific nodes, $@_t$ jumps to a node named by t. This allows policies to be defined requiring combinations of multi-relation paths between pairs of, arbitrary or specific, users.

Bruns *et al.* define an archetypal access control model using this hybrid logic policy language but continue to limit it in several ways. Firstly, the set of nodes they support only contains principals (users), and not their resources. Secondly, they limit the free variables to `own` and `req`, for owner and requester respectively, and determine satisfaction between them. Several more complex policies are illustrated, however, with the use of nominals, to identify specific users, and graded may modalities, such as $\langle i \rangle_n \phi$ to require the current node to have (at least) n distinct i-relations with other nodes from where ϕ holds.

[21] Whilst the grammar can support policies not terminated at the owner (e.g., $\langle$`spouse`$\rangle\top$, which simply requires the accessor to have a `spouse` relation with some user) Fong's satisfaction semantics are defined between the accessor and the owner.

[22] Within the grammar n is a nominal, x is a variable, and p is a proposition.

2.4.3 U2U and U2R

In 2012, Cheng *et al.* also focused on the use of relationship-based access control within OSN. Whilst their initial work only considered *User-to-User (U2U)* relationships [52], this was swiftly followed by work which also allows for the specification of *User-to-Resource (U2R)* relationships [51].

Cheng *et al.*'s U2R relationship-based access control model comprises six basic components (as illustrated in Figure 2.10,[23] adapted from [51]): users; sessions; resources; policies; a *Social Graph (SG)*; and a *Decision Module (DM)*. The social graph contains those users and resources which are part of the OSN, along with the relations between these entities. The resources concerned may be objects (O), policies (P) or user sessions, and the resource relations are not limited to user ownership as had been the convention up to this point.

When users log into the OSN they establish sessions and may, subsequently, request access to act on target entities (whether users or resources). As sessions are considered resources, and so may themselves be targeted by actions, Cheng *et al.* distinguish actions which may be made against offline target users, i.e. those without a session, and online target users with a session.

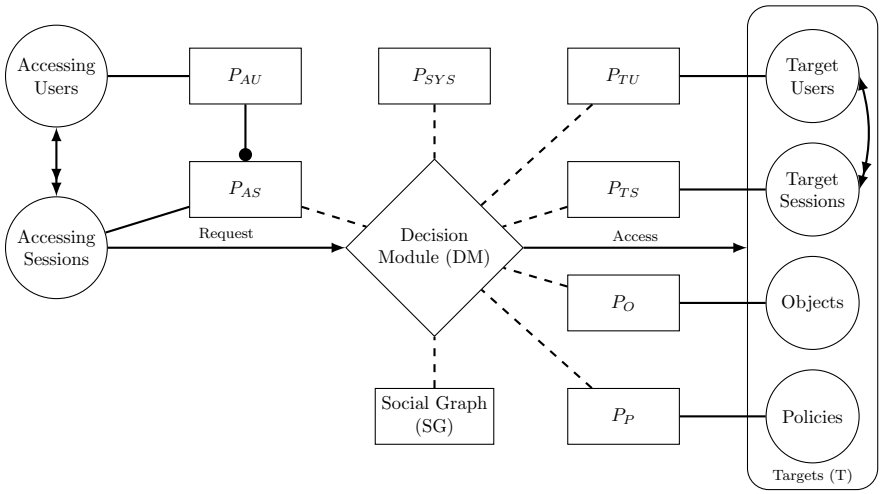


Figure 2.10: U2R model components

The U2R model supports numerous policy components, many of which are under individual control. The centralized *System-specified Policy (P_{sys})* includes an authorization policy which applies to all requests, and also includes conflict resolution policies for dealing with issues that occur when evaluating the many U2R

[23] Within Figure 2.10 dashed lines indicate inputs, solid lines without heads indicate attachment, solid lines with a single arrowhead indicate the request evaluation process, solid lines with multiple arrowheads indicate 1-to-n mappings (where the double arrowhead indicates the n end), and solid lines with a single circle head indicate constraints.

policies. Policy conflicts which arise may be resolved using disjunction, conjunction or prioritisation.

As well as these system-specified policies, the U2R model includes six other types of policy:

- Accessing User Policy (P_{AU}) – a policy defined by a user to constrain what they themselves can access.
- Target User Policy (P_{TU}) – a policy defined by a user to constrain what actions can be performed on them.
- Accessing Session Policy (P_{AS}) – a policy defined by a *Controlling User (CU)* to constrain what their sessions can access.
- Target Session Policy (P_{TS}) – a policy defined by a CU to constrain what actions can be performed on the controlled sessions.
- Object Policy (P_O) – a policy defined by a CU to constrain what actions can be performed on the controlled objects.
- Policy for Policy (P_P) – a policy defined by a CU to constrain what actions can be performed on the controlled policies.

Each of U2R's authorization policies specifies an action which is authorized if an associated *graph rule* is satisfied in the social graph. The graph rules, as illustrated in Figure 2.11, identify a starting node and a *path rule*, comprising one or more *path specs* which are combined using conjunction, disjunction and negation. Path specs, in turn, consist of a *path* of relations, based on regular expressions, along with a hopcount. Whilst Kleene operators ($\star$ and $+$) are allowed in these paths, the hopcount limits the distance (i.e., the number of relations) over which a path of relationships may be identified.

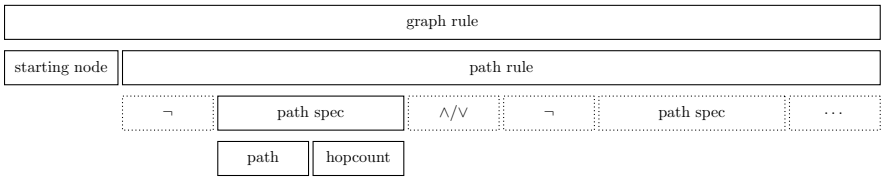


Figure 2.11: U2R graph rule structure

When evaluating requests, the decision module employs a path checking algorithm to evaluate each relevant path spec within the social graph, beginning from the appropriate starting node (which may be defined as the requester, target(s), or controlling user). The (destination) evaluating node depends upon the policy and

whether the requester is the starting node. If the policy is P_{sys} or P_{AS} the evaluating node is the requester; for other policies, if the requester is the starting node then the evaluating nodes are a set including the target(s) of the request and the controlling user of the policy, otherwise the evaluating nodes are just the target(s) of the request. The results of checking each path spec are combined within a path rule to determine the outcome for each of the relevant policies' graph rules. These outcomes are then composed across the relevant authorization policies using the conflict resolution policy to produce a final result.

2.5 Model Internals

So far in this chapter I have described significant developments in the field of logical access control, highlighting key concepts upon which access control may be grounded (such as the use of roles, attributes and relationships) and important model implementations (such as the access matrix, MLS, RBAC, XACML, and ReBAC). It should be clear from the discussion so far, that a protection system may employ various types of information to inform an authorization decision. Further, there are multiple ways in which this information may be structured and processed as part of the evaluation of the system's authorization policy. A comparison of the internals of some of the existing models is shown in Table 2.2, and discussed in the following paragraphs.

	Access Matrix	**MLS**	**UNIX**	**RBAC**	**XACML**	**ReBAC**
Grounding	Users	Levels	Security Principals	Roles	Attributes	Relationships
Data Structure	Matrix	Lattice	ACL	Sets	Tuples	Graph
Evaluation Algorithm	Matrix Lookup	Ordering Check	ACL Lookup	Set Membership	Policy Algebra	Policy Algebra
Algorithm Input	Subject and Object	Subject's Level and Object's Level	Security Principal and Object	Activated Roles and Permission Assignment	Tuple Collection and Policy Rules	Graph Relations and Policy Logic

Table 2.2: Comparison of existing models' internals

Access Matrix In the access matrix model, the privileges granted to subjects on objects are encoded within a matrix (as the name suggests). The matrix itself is a dynamic data structure; changes may be made to privileges as described in Section 2.1.3. However, the information required for the evaluation of an authorization request is encoded completely in the matrix at the time of request evaluation. The

evaluation process itself is a simple lookup of subject and object in the matrix, with no complex processing required.

MLS In contrast, in multi-level security subjects and objects are assigned levels which are themselves defined within a lattice. The lattice of levels is static, and whilst new subjects and objects may be introduced to the system, their security levels are managed centrally (as described in Section 2.1.4). The information required for the evaluation of an authorization request is encoded completely in the lattice of levels and the labels of subjects and objects. The request evaluation process involves an ordering check of the subject's level versus the object's level in the lattice.

UNIX UNIX employs access control lists, one per object, to store the permissions (read, write, and execute) granted to three security principals ("owner", "group" and "other"). The access control list data structure is static in size, as the number of principals is fixed, but permissions may be changed within the data structure using the write permission. The request evaluation process is performed in two steps. Firstly, a security principal is matched to the request by checking whether the subject is the object's "owner", or, if not, whether the subject is a member of the object's "group". If the subject has neither of these relationships with the object, then the "other" security principal applies. The second step involves a simple lookup of the matched (request-time) security principal in the access control list.

RBAC As described in Section 2.2.1, in RBAC roles are employed to abstract permission assignment away from subjects. The numerous assignments within RBAC (including users to roles and permissions to roles) can, fundamentally, be represented through sets. These sets are dynamic data structures, with the activation of roles within sessions being the most dynamic aspect. The request evaluation process involves a chain of set membership checks through which a subjects activated roles are identified and the permissions assigned to these roles are determined.

XACML Within the XACML model it is the evaluation of attributes against policy rules using a policy algebra which determines the authorization decision. This policy algebra not only determines how to process individual rules, it also identifies how decisions within policies and policy sets may be combined (as described in Section 2.3.3). Subjects and objects may have values for a wide range of attributes, where each attribute and value form a tuple (or key, value pair).

ReBAC In a similar way to XACML, ReBAC (as described in Section 2.4.2) employs a policy algebra to evaluate authorization policy. As I have described, the fundamental difference is that ReBAC employs relationships rather than attributes as the grounding for its protection system. The (social) graph of relationships between users in ReBAC is the dynamic data structure which the access control model relies upon. Each policy rule (predicate) defines a path of relationships which much be satisfied for the accessor to be granted.

RPPM In this thesis I shall introduce the RPPM model of access control. This relationship-based access control model employs a graph of relationships as its core data structure (similar to ReBAC), a two-step request evaluation process determining security principals at request-time (similar to UNIX), and implementation-specific security principals to abstract permission assignment away from subjects (similar to RBAC).

2.6 Summary

When compared to many scholarly endeavours, the development of logical access control is relatively young. That being said, interest in logical access control has existed throughout a considerable portion of the history of digital electronic computers, indicating its significance since multi-user sharing computer systems where developed. During that time there have been a variety of conceptual frameworks used to ground the designs of access control models. I have discussed privileges, capabilities, security levels, roles, attributes, and relations within this chapter. It is important to recognise that there are some natural constraints which flow from using each of these foundations, but that the design of the access control model determines the extent to which these constraints are limiting, and what additional constraints affect the model's use.

Chapter 3

Motivation

It should be clear from Chapter 2 that authorization is a key security service within modern computing environments and that there is an existing range of access control models available to implementers. Whilst a range of choices does exist, it should also be clear that it commonly takes time to develop and refine the concepts on which access control models are based. Throughout their development and use, even fundamental principles (such as what constitutes mandatory access control) have seen varied interpretation and debate, often driven by differences in perspective.

Given the increasing ubiquity of computing, and the increasing significance of information and cyber security, there is a general need for security services to support the growing use of technology in existing, and new, applications. Each of the access control models introduced in Chapter 2 was developed to achieve specific goals based on a particular body of knowledge and with particular use cases and constraints in mind. As each new model has been introduced this has, therefore, been with the intention of addressing limitations in existing models or to accommodate richer types of access control policy. I continue that trend in this thesis. Specifically, I believe that relationship-based access control may be applied far more broadly than prior work has considered, and that the use of relationships is more intuitive than other foundations for a range of topical environments.

3.1 The Current Norm

Whilst recently both attribute-based access control (ABAC) and relationship-based access control (ReBAC) have been undergoing considerable active research, many computer systems and organisations are still primarily using access control lists (ACLs) with privileges granted to groups and individual users. Whilst mandatory access control implementations are now commonly supported on UNIX and Linux

operating systems (as detailed below), these are primarily operating pre-configured, targeted profiles which protect specific system components and services rather than user data as a whole. Role-based access control (RBAC) is more prevalent in web applications and cloud services, but these equally place significant reliance on access control lists.

Microsoft Windows In Windows the underlying authorization and access control process employs *access control lists* to provide *discretionary access control* [130, 133, 135]. This is the primary access control model employed by users and administrators of Microsoft Windows. Two additional mechanisms are currently also supported, but I believe are less widely used and understood.[1] Authorization Manager, available since 2003, can enforce a *role-based access control* model [132]; however, this is deprecated in Windows Server 2012 and so will be removed in some future release of the operating system [131]. Dynamic access control (DAC), which implements an *attribute-based access control* model, was introduced in Windows 8 and Windows Server 2012 [134]. Whilst these features are installed by default, they cannot be implemented outside of an Active Directory domain and so can only be used in enterprise networks.

UNIX/Linux Both UNIX and Linux make use of UNIX file system *read, write and execute permissions* which are specified for: the object's "owner", identified by user id (uid); members of the object's assigned "group", identified by group id (gid); and everyone else (known as "other" or "world"). Individual subjects are mapped to one of these three principals at the time of request evaluation. In this way, whilst each object must still be enumerated, the enumeration of subjects is limited to just these three security principals. However, the root user (identified as uid 0) bypasses authorization checks and so is able to perform any action on any object, no matter the defined permissions. Since this initial, limited, security model was introduced, individual "flavours" of UNIX and "distributions" of Linux have added their own security features in order to support more robust models. UNIX extensions and Linux Security Modules (LSM) enable such additional security features to be "plugged in" through loadable modules:

- *UNIX extensions* – various modules support various extensions [86]:
  - BSD file flags – override specific UNIX permissions to prevent certain accesses by all users.

[1] Whilst articles identifying the existence and basic configuration of these mechanisms is available online, I have been unable to find any articles or case studies indicating their actual use outside of Microsoft IT.

  - POSIX access control lists – provide more fine-grained control over the permissions granted or denied to users or groups.
  - POSIX MAC modules – provide support for targeted or general policies using labels.

- *Linux Security modules (LSM)* – various such modules exist implementing mandatory access control models:
  - SELinux – is able to monitor or enforce configured policies using labels. SELinux is enabled by default with a targeted policy (i.e., covering only specific processes) in CentOS, Fedora and Red Hat [46, 74, 98].
  - AppArmor – is able to monitor or enforce configured policies using file paths and is enabled by default in Ubuntu [181].
  - Smack – is able to enforce configured policies using extended attributes and is enabled by default in Tizen [179].

Apple MacOS Apple's OS X contains several security subsystems due to the fact that it comprises several kernel components. The lower-level Mach kernel component provides a port-based interface to the CPU and memory, which it protects with port rights; these are *capabilities* required to grant permission to send or receive through the port [11, 12]. The upper-level BSD kernel component provides the file system, networking and UNIX security model. Through these components OS X gains [13]:

- Basic user credentials – if the caller is running as root, they bypass further authorization checks. Additionally, in OS X any member of the admin group can perform "almost all functions the root user can".
- Sandbox entitlements – OS X apps are limited to specific file system and networking resources no matter the permissions their processes may have. This sandboxing is provided by a *mandatory access control* framework, based on TrustedBSD [178].
- POSIX *access control lists* – OS X v10.4 and later enables users or groups to be granted or denied specific permissions on individual files and directories.
- UNIX permissions – *read, write and execute permissions* are specified for the owner, the group, and all others on each file system object.
- BSD file flags – when *file flags* are set they override specific UNIX permissions and so prevent certain file system operations such as backing up, moving, renaming, and over-writing objects.

Android Android is based on the Linux kernel and makes use of the file system *read, write and execute permissions* that this kernel provides [10]. To protect the system's core libraries, runtime and apps, Android *mounts the system partition as read-only.* In addition, Android implements SELinux's *mandatory access control* model which is configured with everything in enforcing mode since Android 5.0 (L) [9].

Apple iOS Whilst iOS is based on Apple's OS X, and so implements the same underlying permissions model as described above, the app sandboxing (i.e., *mandatory access control*) constraints are the primary limiting factor due to the configuration of iOS devices [13].

Web Applications and Cloud Services Numerous web-based application and infrastructure platforms are available to modern users, and many base their protection systems on access control lists with permissions assigned to users and groups. Some may even implement a simplified version of role-based access control (usually where the roles are pre-defined and administration-related, implemented without any hierarchy). For example:

- Atlassian Cloud – (used for software development and collaboration) supports access control lists for users, groups, and pre-defined project roles [17].
- Dropbox Business – (used for data sharing) supports sharing of folders with groups, where each group is assigned to either an "editor" or "view-only" role [71].
- Intuit QuickBooks – (used for book-keeping) also supports access control lists for users and a special "master administrator" role [153].
- Salesforce – (used for customer relationship management) supports access control lists for users and groups, and also supports role-based access control with a role hierarchy [159].

In contrast, social networks like Facebook typically employ relationship-based access control (ReBAC) models, often based on a single type of relation between user entities (commonly friend) and another between resources and the controlling user (commonly owner) [83].

3.2 Existing Issues

3.2.1 List Management

Whilst access control lists are still commonly employed (as per Section 3.1) due to their simplicity, their use continues to suffer from a significant burden of administration [77]. The introduction of new users into a system requires an associated assignment of permissions on existing protected resources. Equally, the introduction of a new resource requires the assignment of permissions to appropriate existing users. The same issues arise when users move from one part of an organisation to another.

It was these issues which, in part, motivated the proposal and development of role-based access control [80]. Nevertheless, ACLs have continued to be used. Modern implementations of ACLs commonly make use of groups and inheritance as ways to partially reduce the burden of managing file system permissions in such scenarios [13, 133]. Whilst these features remove the need to undertake a considerable number of administrative actions in the general case, the administration of ACL permissions in any other scenario remains burdensome.

Whilst a greater adoption of more easily managed models is the true solution, there are a number of likely reasons why significant use of ACLs has perpetuated: they are conceptually simple; they potentially give an illusion of greater control by directly linking users (or groups) to resources; and there are few operating systems in mass use today which are not built on the foundations of those initially created decades ago. Whatever the cause, the ongoing limitations of ACLs have led management products to be developed (e.g., [152]), wrapping more intuitive user tools around the model with the goal of easing some of the associated administrative burden.

An intuitive, easily managed access control model continues to be needed to replace access control lists. Whilst role-based access control was meant to be that model, the issue of role explosion (discussed next) has limited the perceived benefit.

3.2.2 Role Explosion

Role-based access control was motivated by the desire to "articulate and enforce enterprise-specific security policies and to streamline the typically burdensome process of security management" [77]. RBAC's ability to support a range of useful enterprise policy configurations (as discussed in Section 2.2.3) alone makes it far superior to access control lists. However, it is the simplification of administration

which perpetuated as the main motivating factor [163].

RBAC's "abstraction" of permission assignment to roles, rather than users, alleviates some of the administrative burden of ACLs. Whilst direct assignment may give an illusion of control, in reality the abstraction behind roles is a far more powerful construct. The intention of role abstraction is that the number of roles should be far less than the number of users in a system. As noted by Sandhu *et al.* in [163]:

> "Administration of RBAC is very important, and must be carefully controlled to ensure that policy does not drift from its original objectives. In large enterprise-wide systems, the number of roles can be in the hundreds or thousands, and users in the tens or hundreds of thousands."

RBAC enables policies to be defined for roles, which are fewer in number than users, thus requiring less effort to manage. However, as indicated by Sandhu *et al.* there may still be a huge number of roles in practice. In some cases an "explosion" of roles occurs due to the need to distinguish specific contextual constraints that must be considered alongside a particular base role. Hilchenbach highlighted this using the example of probation periods for bank tellers when reporting observations from early implementations in 1997 [101]. I provide two, related, motivating examples of the inadequacy of roles to support context.

Wide-Acting Roles A commonly cited example, which is related to Hilchenbach's, demonstrates the issue of roles which are too wide-acting. This example considers a Doctor role within a healthcare record management system [20, 21, 48, 62, 63, 82, 143, 157]. Whilst the general statement that a Doctor role should have access to patient healthcare records is likely considered true, in reality many security researchers do not desire that any member of the Doctor role should have access to all healthcare records. There may be specific healthcare records, such as those of heads of state or heads of government, which it may be deemed necessary to have more restricted; but more generally, for patient privacy, many would only wish access to their healthcare record by a Doctor who is directly treating them, or (legitimately) advising on such treatment. It is this relationship context which is lacking in role definitions.

Conflicting Roles The second example demonstrates an issue which arises when the same user may occupy different roles in different contexts. This example considers a PhD student enrolled as a student on a postgraduate course, and also working as a teaching assistant on an undergraduate one [63]. As a member of the university

department's Student role and Teaching-Assistant role, issues arise when the PhD student attempts to read coursework of another student. Obviously, I would wish the request to be granted if the coursework was submitted to the undergraduate course, but denied if it were for the postgraduate course. However, it is unclear from RBAC alone what should happen in the case of such a conflict. In reality, the outcome (grant or deny) would depend on the specific implementations of the model and the policy, with both outcomes possible given particular implementations. Once again it is the relationship context which is lacking.

Whilst the parameterization of roles offers a potential means of resolving such issues [88], this adds further to the role explosion already highlighted. Certainly having a Student-for-Course-X role and also a Teaching-Assistant-for-Course-Y role (along with their counterparts Student-for-Course-Y and Teaching-Assistant-for-Course-X) would avoid the conflict previously identified, but this could rapidly require a large number of roles to accommodate the necessary options in a university. It is less clear how the Doctor role should be partitioned as part of parameterization, but it is likely that significantly more roles would result whatever approach were employed.

Given the desire that RBAC should ease administrative burden, the issues associated with role explosion mean that an intuitive, context-aware access control model is needed. Whilst some would say that attribute-based access control is meant to be that model, it is not appropriate for all applications (as discussed next) [41, 63, 85].

3.2.3 Attribute Applicability

The fact that attribute-based access control is able to support multi-factor decisions is a significant development over the role-specific viewpoint of RBAC (the key features of which ABAC can implement using a role attribute). ABAC provides a versatile basis on which authorization policy may be defined, using attributes which are of particular relevance to the system in question. Specifically, ABAC enables policy to be defined with respect to one or more attribute values such that entities may be provisioned into a system without referencing or altering the protection policy. Further, this avoids the need for capabilities to be directly assigned to subjects and so allows the evaluation of attributes at request time.

Whilst the generality of ABAC makes it suitable for a wide variety of applications, it is not believed to be the most appropriate model for every environment [41, 63, 85]. Sharing in social networks can be more easily articulated through relationships than attributes, and, as indicated in Section 2.4.1, the fact that the human race is such

a social species means that we use relationships to comprehend and communicate all manner of abstract ideas. I, therefore, share the view of Fong and Siahaan that "in many emerging application domains" an access decision "depends not on the intrinsic attributes of the accessor, but on the relationship between the accessor and the owner" [85].[2] Instead, I believe that relationship-based access control has great utility – a fact further evidenced by the breadth of research in this field[3] and recently acknowledged by Ahmed *et al.* when they stated that both ABAC and ReBAC "have considerable applications in industry, and are anticipated to continue being important for the foreseeable future" [3].

Ahmed *et al.*'s recent work comparing generic types of ABAC and ReBAC models has determined that both equivalences and non-equivalences exist between these models [3]. Most notably in their comparison, they identify that ABAC models which support various types of attribute value[4] are more expressive than pure ReBAC models, which do not themselves make use of attributes (on either nodes or relationships). However, they also identify that for ABAC to express a "node dynamic" ReBAC model, i.e. one where the entities and relations are able to change through the life of the model, then the ABAC model must support attributes with values from a countably infinite set of entities.

Further work is required on ABAC to determine the real-world implications this may have. It may be that this simply impacts formal analysis, and that from a purely analytical perspective ABAC may be the more expressive model. However, given that it is the relational context which is missing in the RBAC scenarios discussed in Section 3.2.2, it seems counter-intuitive to encode these relationships within complex attributes as part of ABAC when they may be treated simply and natively in a ReBAC model. In the future ABAC and ReBAC may be seen as simply different perspectives on a single access control problem, with particular benefits of usability, intelligibility, expressiveness, computability, efficiency of storage, and efficiency of computation determining which of the perspectives (or their implementations) is particularly relevant for representing a system at any moment in time.

3.2.4 Relationship Context

As with ABAC, relationship-based access control enables authorization policies to be defined independent of the specific participating entities [105]. Therefore, the

[2] As this thesis will demonstrate I, further, believe that relationships with, and between, resources are also highly relevant to access decisions.

[3] For example, with regard to: OSN, [5, 41, 43, 44, 45, 51, 52, 83, 104, 120]; more general computing, [60, 62, 63, 64, 65, 82, 84, 85]; policy languages, [41, 85, 128, 194]; administration, [49, 64, 157, 174]; implementation, [157]; and other models, [3, 21, 50, 156].

[4] Specifically, values representing identifiable entities, non-entities, and structured combinations of the two.

addition (and deletion) of entities and relationships does not require an associated change in policy to impact the decisions which result from authorization requests. This significantly alleviates the administrative burden associated with access control lists. Further, the use of environment-specific relationships enables the appropriate interactions to be represented whilst avoiding "tedious role parameterization" [84].

Whilst there are a number of existing relationship-based access control models, each has limitations which prevent their broader application. Here I summarise the key features and limitations of five significant ReBAC models. These limitations motivate the features of the general computing relationship-based access control model which is the topic of the remainder of this thesis.

Carminati *et al.*, 2006 [44] This simple social network model uses a graph which contains users connected by relations of various types. The graph does not support resources, and so the model assumes ownership is the single (extra-model) user-resource relation. Another limitation of this model is that its policy *access conditions* may only identify direct (or indirect) chains of relationships of a single relation type. It is, therefore, not possible to define policies over multi-relation chains. Whilst access to objects is controlled by positive *access rules* (comprising disjoint sets of conjoined access conditions) which must be satisfied for access to be granted, negative rules are not supported. Finally, requests are evaluated solely between the requestor and the resource owner.

Carminati *et al.*, 2009 [43] This semantic web model is also limited to social networks, but the graph for this model supports both users and resources. The model supports various types of relationships between all combinations of users and resources. In addition, the policy *security rules* may be authorization rules or prohibition rules, with each type of rule containing a conjunction of atoms, as described in Section 2.4.1. However, as atoms can reflect at most a single relation between two object nodes, paths of relationships must be carefully constructed using distinct atoms with common object labels or variables. Whilst variable relationship type paths may be defined, the need to identify both objects involved in each direct relation (by variable, if not specifically by label) limits the policy language to pre-determined paths. Requests are evaluated between the target subject and the target object, but individual security rules are evaluated between the contained atoms' objects.

Fong (and Siahaan), 2011 [82, 85] and Bruns *et al.*, 2012 [41] The ReBAC model introduced by Fong (and Siahaan) and developed by Bruns *et al.* is mostly presented in terms of social networks; however, they have also demonstrated its use for the sharing of healthcare records. The model's graph is limited to users and, as with Carminati *et al.*'s 2006 model, it is assumed that ownership is the single (extra-model) user-resource relation. The logic-based policies (originally modal logic and later hybrid logic) support multiple chains of different relations joined using conjunction, disjunction and negation. When hybrid logic is supported, these policies also allow the binding of variables to user nodes. Whilst the grammar is able to define chains between the requestor and arbitrary third-parties, the request processing requires the policy to be satisfied between the requester and the resource owner. A further limitation of the ReBAC model is that a policy must be defined for (and then is stored with) each resource individually.

Cheng *et al.*, 2012 [51, 52] Cheng *et al.*'s U2U and U2R relationship-based access control models are, once again, focused on social networks. Whilst the U2U model graph is limited to users, the U2R model graph also supports three kinds of resources: objects; policies; and user sessions. The U2R model supports any relations amongst the users and resources, making this the most flexible graph yet. Cheng *et al.*'s policy language is based on a *path* of relations constructed using regular expressions. However, whilst the Kleene operators are supported, the lengths of satisfying paths are limited by a *hopcount*. It is claimed that such restrictions are appropriate in social networks because of the "six-degrees-of-separation" phenomenon, which means the diameter of a social network is very small (compared to the number of nodes it contains); such claims do not apply for more general applications. The *path specs* (paths and hopcounts) are joined using conjunction, disjunction and negation (much like the chains in Fong's ReBAC) to produce a *path rule* which is associated with a *starting node*. Evaluation of the resulting path rule requires satisfaction between the starting node and each of the evaluation nodes (see Section 2.4.3 for information on how these are determined). However, Cheng *et al.* do not allow cycles when satisfying path rules in the social network. There are seven distinct policies (system-specified policy, accessing user policy, target user policy, accessing session policy, target session policy, object policy, and policy for policy) which are enforced within the U2R model, which apply to a request depends on the requester and the resources targeted, as described in Section 2.4.3.

Whilst these relationship-based access control models introduce a variety of useful

concepts, each limits the graph (its nodes or relationships) or the policy language (use of multiple relationships in a chain or the length of such chains) in a manner which prevents it from being an effective alternative to ABAC in more general computing applications.

3.3 Summary

Whilst modern computer systems still make considerable use of early access control models, where newer models are available they provide significant benefits. However, whilst newer models are introduced with the intention of addressing limitations in existing models, this does not mean that they are without their own issues or are the best choice for all environments. Given the issues I have highlighted in Section 3.2, I believe (as do others) that there are opportunities for developments in relationship-based access control. I have, therefore, been motivated to develop a relationship-based access control model which is more flexible in its graph and policy construction, such that it can provide: less burdensome administration than access control lists; a range of policy configurations without the role explosion of role-based access control; more intuitive relation-driven sharing to suit applications less appropriate for attribute-based access control; and a broader range of applications than the existing relationship-based access control models. The remainder of this thesis describes that model.

Part I

RPPM's Fundamentals

Chapter 4

$\mathbf{RPPM_0}$

The base RPPM model (RPPM_0) is designed to address many of the concerns discussed in Section 3.2. As with other relationship-based access control models, the central component of the RPPM model is a labelled graph, the *system graph*, in which nodes represent the system's entities and the labelled edges represent relationships between them.

The functionality I will introduce throughout this chapter instils desirable properties in RPPM:

- Generality – RPPM's system graph supports nodes representing any concrete and logical entity types. System-specific entity types may be used as required, with logical entities able to give context to concrete entities.
- Abstraction – RPPM uses paths of edges in the system graph to identify security principals to which permissions are associated. This abstraction eases administration in a similar way to that of roles in RBAC; it also enables the content and arrangement of the system graph to be changed independently of RPPM's policies.
- Conciseness – RPPM's access control policy language and rule structure is designed to minimise RPPM's components, avoiding duplication and waste, whilst maintaining granularity and robustness. Amongst other things this allows rules to be defined which can apply to many (or even all) objects and actions, further easing the administration of the model.
- Versatility – RPPM's access control policy may be defined by requiring the existence of a path of edges between the subject and the object, by forbidding the existence of such a path of edges, or both.

- Completeness – RPPM is able to guarantee a single authorization decision when a conflict resolution strategy and default decisions are employed.

Much of the functionality of the RPPM$_0$ model described here has previously been published, in conjunction with Jason Crampton, in [62, 63].

4.1 Overview

4.1.1 Grounding

The RPPM model is designed as a general model for access control, utilising relationship information in order to make authorization decisions. The generality comes from the model's ability to support whatever entity and relationship types are necessary to describe a particular system at the desired level of detail. Abowd *et al.* define the term *context* to refer to "any information that can be used to characterize the situation of an entity" [2]. As well as concrete entities, such as users and resources, RPPM's nodes can also represent logical entities with which other entities are associated. These logical entities (and their connecting relationships) can be employed to give context, or some system-specific grounding, to the concrete entities. For example, in the case of a medical records management system we may have concrete entities representing patients, doctors, healthcare records and medicines; additionally we may have logical entities representing medical cases, healthcare teams and research projects.

4.1.2 Request Evaluation

Request evaluation in the RPPM model is a two-step process, as shown in Figure 4.1, where first I *compute principals* and subsequently *compute authorizations*. This two-step request evaluation process is inspired by UNIX, which first determines the relevant principal (from "owner", "group" and "other") and then authorizations (from the permission mask of the object) [57]. Unlike UNIX, however, RPPM does not constrain a modelled system to a particular set of principals.

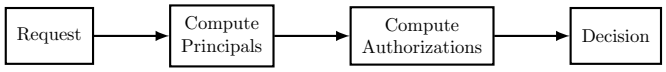


Figure 4.1: Request evaluation processing overview

The RPPM model employs two distinct policies to support the two-step request evaluation process: the principal-matching policy and the authorization policy. The purpose of the principal-matching policy is to determine which security principals

are relevant to an access request. The purpose of the authorization policy is to determine whether the principal is authorized to perform the requested action.

A security principal is, therefore, the central component in RPPM's policies and, in line with the definition given by Saltzer and Schroeder, is the "entity in a computer system to which authorizations are granted" [162]. A request is mapped to a set of principals and each principal is authorized to perform particular actions. This abstraction avoids directly assigning permissions to subjects, enabling the system graph to be updated in isolation from the policies. Further, I use principals to ease the administration burden in a way analogous to the use of roles in role-based access control. However, my association of subjects to principals is a request-time dynamic process, unlike the static membership of roles utilised in RBAC.[1]

4.2 Compute Principals

4.2.1 System Model and System Graph

The flexibility provided by allowing RPPM's nodes to represent any concrete or logical entities is powerful; however, it can also limit the checks and controls available for administration of an implementation of the model. In order to provide an underlying structure and, therefore, a basis on which to incorporate the appropriate checks and controls, I first define a system model which constrains the "shape" of the system graph.

Definition 4.1. *A* system model *comprises a set of types T, a set of relationship labels R, a set of* symmetric *relationship labels $S \subseteq R$ and a* permissible relationship graph $G_{PR} = (V_{PR}, E_{PR})$*, where $V_{PR} = T$ and $E_{PR} \subseteq T \times T \times R$.*

Definition 4.2. *Given a system model (T, R, S, G_{PR}), a* system instance *is defined by a* system graph $G = (V, E)$*, where V is the set of entities and $E \subseteq V \times V \times R$, and a function $\tau : V \to T$ which maps an entity to a type. G is* well-formed *if for each entity v in V, $\tau(v) \in T$, and for every edge $(v, v', r) \in E$, $(\tau(v), \tau(v'), r) \in E_{PR}$.*

The definition of a system graph allows for multiple edges between nodes, as multiple relationships frequently exist between entities in the real world; such a graph is sometimes called a *multigraph*. Such edges may be directed (asymmetric) or undirected (symmetric) depending on the type of relationship. I will depict an edge (v, v', s), when $s \in S$, without arrowheads, as can be seen in Figure 4.2 in the case of the Sibling-of relationship. (Due to the symmetry of s, the edge (v, v', s)

[1] RBAC does enable session-specific activation of roles, but even so the roles are then static for the session and those available for the session only come from those already (statically) authorized to the user.

implies an edge (v', v, s) and vice versa.) An edge (v, v', r), when $r \in R \backslash S$, is directed from v to v', depicted with an arrowhead pointing towards v' (see Figure 4.2a). The directed edges (v, v', r) and (v', v, r) represent two different relationships. Of course both may belong to E, in which case this will be depicted with arrowheads at both ends of the link between v and v' (see Figure 4.2b).

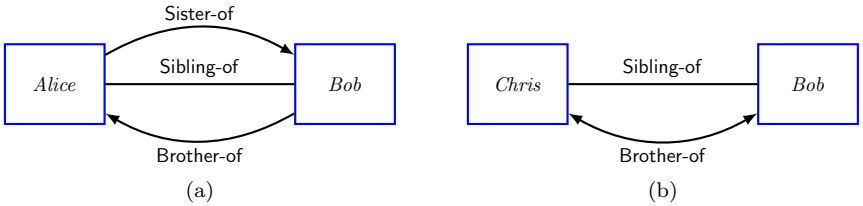


Figure 4.2: Illustrating different edges in system graph instances

The administrative interface for any implementation of the RPPM model must ensure that the system graph is always well-formed with respect to its underlying system model. As the system being modelled may well be dynamic, updates to the system graph must be controlled in order to continue to maintain its well-formedness. I will discuss administration of RPPM models and policies in detail in Chapter 8.

Remark 4.1. *It is important to note that whilst, for ease of exposition, I will regularly use examples involving entities of a "user" type, the RPPM system model and system graph definitions place no requirement on human participation. In fact, all of the RRPM functionality I will introduce in this thesis (no matter the RPPM model) operates exactly the same if there are no human entity types employed and if the participants of every request are autonomous entities, automated agents or even inanimate objects. The make-up of any instance of the RPPM model is principally driven by the system being modelled. Only where that system requires entities reflecting human "actors" should they be modelled through appropriate entity types.*

4.2.2 Path Conditions

In order to limit the administrative burden of defining access control policies in systems with many subjects, the RPPM model abstracts permission assignment to security principals (in the same way that roles simplify policy specification and maintenance in RBAC). To determine if a particular principal is relevant to a request, an associated required (positive) chain of relationships must be matched between the subject and object of the request whilst a forbidden (negative) chain must not be matched. Such chains of relationships are called path conditions, and are composed of relationship labels, organised as sequences, with support for several regular

expression-like operators.[2]

Definition 4.3. *Given a set of relationships R, a* path condition *is defined recursively as:*

- $\diamond$ *is a path condition;*
- r *is a path condition for all* $r \in R$*; and*
- *if* π *and* π' *are path conditions, then* $\overline{\pi}$, π^+, (π) *and* $\pi\,;\pi'$ *are path conditions.*

A path condition of the form r *or* $\overline{r}$*, where* $r \in R$*, is said to be an* edge condition.

Informally, $\diamond$ defines an "empty" path condition, whilst $\overline{\pi}$ represents π reversed. π^+ (the Kleene plus operator[3]) represents one or more occurrences, in sequence, of π, and (π) provides a means of clearly indicating the extent of path condition π such that use of the Kleene plus operator is unambiguous. Lastly, $\pi\,;\pi'$ represents the concatenation of two path conditions.

The satisfaction of a path condition is defined relative to a system graph G and two nodes u and v in the graph.

Definition 4.4. *Given a system graph* $G = (V, E)$ *and two nodes* $u, v \in V$*, I write* $G, u, v \models \pi$ *to denote that the path condition* π *is* satisfied *from* u *to* v *in* G*. Then, for all* G, u, v, π, π'*:*

- $G, u, v \models \diamond$ *iff* $v = u$*;*
- $G, u, v \models r$ *iff* $(u, v, r) \in E$*;*
- $G, u, v \models (\pi)$ *iff* $G, u, v \models \pi$*;*
- $G, u, v \models \overline{\pi}$ *iff* $G, v, u \models \pi$*;*
- $G, u, v \models \pi\,;\pi'$ *iff there exists* $w \in V$ *s.t.* $G, u, w \models \pi$ *and* $G, w, v \models \pi'$*; and*
- $G, u, v \models \pi^+$ *iff* $G, u, v \models \pi$ *or* $G, u, v \models \pi\,;\pi^+$.

Path condition satisfaction is a binary concept, and so conversely a path condition may not be satisfied between two nodes u and v in a system graph G.

Definition 4.5. *I use the notation* $G, u, v \not\models \pi$ *to denote that the path condition* π *is* not satisfied *between* u *and* v *in* G*. The meanings for specific* π *follow naturally from the inverse of the definitions in Definition 4.4:*

[2]The definition of path conditions employs common regular expression operators, as do [52] and [114]. I exclude several common operators from my definition for conciseness, see Remark 4.2.

[3]Recall, from Section 2.4.3, that Cheng *et al.* limit their use of Kleene operators with a hopcount in light of the "six-degrees-of-separation" phenomenon. I do not bound the Kleene plus operator as no such assumptions about path length can be made for more general systems. Such unbound paths are useful when traversing a sub-graph comprising similar types of elements that might have arbitrary diameter (as in a directory tree, for example).

- $G, u, v \not\models \diamond$ *iff* $v \neq u$;
- $G, u, v \not\models r$ *iff* $(u, v, r) \notin E$;
- $G, u, v \not\models (\pi)$ *iff* $G, u, v \not\models \pi$;
- $G, u, v \not\models \overline{\pi}$ *iff* $G, v, u \not\models \pi$;
- $G, u, v \not\models \pi\,;\pi'$ *iff there doesn't exist* $w \in V$ *s.t.* $G, u, w \models \pi$ *and* $G, w, v \models \pi'$; *and*
- $G, u, v \not\models \pi^+$ *iff* $G, u, v \not\models \pi$ *and* $G, u, v \not\models \pi\,;\pi^+$.

The compositional nature of path conditions, along with the regular expression-like operators, means that there is flexibility in how chains of relationships can be specified in a path condition. For example, the path conditions $\pi\,;\pi^+$ and $\pi^+\,;\pi$ are valid representations of the same chain of relationships – specifically, two or more instances of the relationship π. I now define what I mean by the equivalence of two path conditions, enabling me to define the concept of *simple* path conditions.

Definition 4.6. *Path conditions* π *and* π' *are said to be* equivalent*, denoted* $\pi \equiv \pi'$*, if, for all system graphs* $G = (V, E)$ *and all* $u, v \in V$*, I have*

$$G, u, v \models \pi \quad \textit{if and only if} \quad G, u, v \models \pi'.$$

Trivially, by Definition 4.4 and the definition of a symmetric relationship, I have (i) $\overline{\diamond} \equiv \diamond$; (ii) $(\pi) \equiv \pi$ for all path conditions π; and (iii) $\overline{s} \equiv s$ for all $s \in S$.

The set of all path conditions, given a set of relationships, is a *monoid* algebraic structure. $\diamond$ acts as the *identity element*, whilst ; provides the single associative binary operation.

Proposition 4.1. *For all path conditions* π_1 *and* π_2*:*

(i) $\pi_1\,;\diamond \equiv \diamond\,;\pi_1 \equiv \pi_1$;

(ii) $\overline{\pi_1^+} \equiv \overline{\pi_1}^+$;

(iii) $\overline{\overline{\pi_1}} \equiv \pi_1$;

(iv) $\overline{\pi_1\,;\pi_2} \equiv \overline{\pi_2}\,;\overline{\pi_1}$;

(v) $(\pi^+)^+ \equiv \pi^+$; *and*

(vi) $\pi_1^+\,;\pi_1 \equiv \pi_1\,;\pi_1^+$.

Proof. All results follow immediately from Definitions 4.4 and 4.6. Consider (iv), for example. By definition, $G, u, v \models \overline{\pi_1 \,;\, \pi_2}$ if and only if $G, v, u \models \pi_1 \,;\, \pi_2$. And $G, v, u \models \pi_1 \,;\, \pi_2$ if and only if there exists w such that $G, v, w \models \pi_1$ and $G, w, u \models \pi_2$. Thus I have $G, u, v \models \overline{\pi_1 \,;\, \pi_2}$ if and only if there exists w such that $G, w, v \models \overline{\pi_1}$ and $G, u, w \models \overline{\pi_2}$. That is $G, u, v \models \overline{\pi_2} \,;\, \overline{\pi_1}$. □

Definition 4.7. *Given a set of relationships R, a* simple path condition *is defined recursively as:*

- $\diamond$, r *and* $\overline{r}$, *where* $r \in R$, *are simple path conditions; and*
- *if* $\pi \neq \diamond$ *and* $\pi' \neq \diamond$ *are simple path conditions, then* (π), $\pi \,;\, \pi'$ *and* π^+ *are simple path conditions.*

In other words, $\overline{\star}$ occurs in a simple path condition if and only if $\star$ is an element of R. It follows from Proposition 4.1 that every path condition may be reduced to a simple path condition. The path condition $\overline{r_1 \,;\, r_2} \,;\, (r_1 \,;\, r_3)^+$, for example, can be transformed into the equivalent, simple path condition $(\overline{r_3} \,;\, \overline{r_1})^+ \,;\, r_1 \,;\, r_2$ using the equivalences in Proposition 4.1.

Henceforth, I assume all path conditions are simple. Thus I define the set of relationship labels to be $\widetilde{R} = R \cup \overline{R}$, where $\overline{R}$ is defined to be $\{\overline{r} : r \in R\}$. Given this formulation, the system graph must satisfy the following consistency requirements:

- $(t, t', r) \in E_{\text{PR}}$ if and only if $(t', t, \overline{r}) \in E_{\text{PR}}$;
- $(v, v', r) \in E$ if and only if $(v', v, \overline{r}) \in E$; and
- $(v, v', s) \in E$ if and only if $(v', v, s) \in E$ and $s \in S$.

Example 4.1. *Returning to my motivating issue of wide-acting roles from Section 3.2.2, I consider the example system graph shown in Figure 4.3.*[4] *It should be clear that I can define a path condition to "describe" the particular sequence of relationships between a doctor and the healthcare record of a patient they are treating. In this way, I may constrain access to such a path, rather than simply to a qualified doctor, or even a doctor simply working at the relevant institution. In the case of Figure 4.3, the path condition* Treating ; $\overline{\text{Pertains-to}}$ *could be used within appropriate policies to grant access to healthcare records by treating physicians.*

[4] Throughout this thesis I use different styles of graph node to represent the different types of entity within the system graph. Where it is relevant I will explicitly indicate what entity types the example's styles relate to using a legend. Elsewhere, such as here, it is sufficient to appreciate that different types of entity are used, without focusing on what those types are.

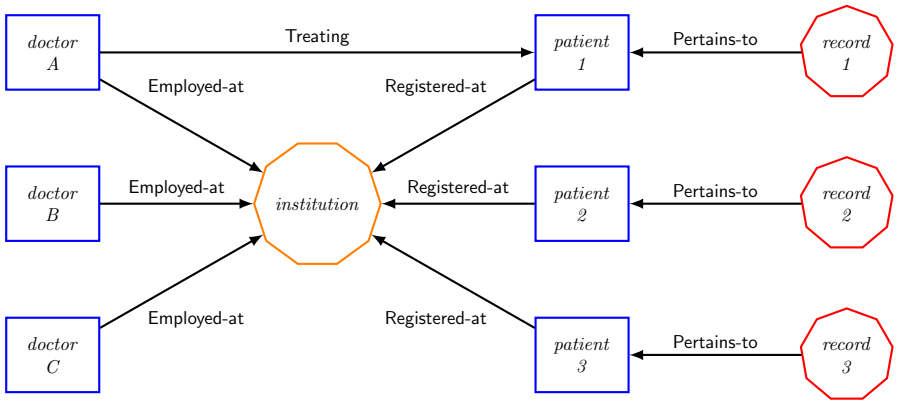


Figure 4.3: A healthcare system graph fragment

4.2.3 Principal-Matching Policy

As just introduced, a path condition allows a path of relationships to be specified such that the path condition's satisfaction indicates that that path of relationships exists between two entities in the system graph. Path conditions can, therefore, be employed within principal-matching rules to enable the compute principals step of request evaluation to identify those principals matched to a particular request.

Each principal-matching rule in the principal-matching policy specifies two *targets* – a required target and a forbidden target[5] – along with a principal. Every path condition π is a target and a request (s, o, a), where s and o are vertices in the system graph G (and $a \in A$ is a requested action), *matches* target π if $G, s, o \models \pi$. I define two special targets: all and none, where all matches every request and none matches no request. By a slight abuse of notation (all and none are not path conditions), I will write $G, s, o \models \mathsf{all}$ and $G, s, o \not\models \mathsf{none}$, for any request (s, o, a). Given a request $q = (s, o, a)$, where s and o are vertices in the system graph G, I will tend to write $G, q \models \pi$, rather than $G, s, o \models \pi$, to simplify notation.

Definition 4.8. *Let P be a set of security principals. A* principal-matching rule *has the form (ϕ, ψ, p), where $p \in P$ and ϕ and ψ are targets. A* principal-matching policy *is a set of principal-matching rules.*

Informally, targets are used to determine which rules are applicable to a given request, where ϕ specifies a required path in the system graph and ψ specifies a forbidden path.[6] The meaning of a principal-matching policy (PMP) is defined in the context of a system graph and a request.

[5]The required target is existential, it must exist at least once to be matched, whilst the forbidden target is universal, it most not exist at all to be matched.

[6]RPPM$_0$ can only consider targets defined by a single path; I will introduce an enhancement in Chapter 5 which will enable RPPM to support conjunction and thereby multiple paths of relations per target.

Definition 4.9. *A principal-matching rule* (ϕ, ψ, p) *is* applicable *to a request* $q = (s, o, a)$ *if and only if* $G, q \models \phi$ *and* $G, q \not\models \psi$. *Given a system graph* $G = (V, E)$, *a PMP* ρ *and a request* $q = (s, o, a)$, *where* $s, o \in V$, *the set of* matched principals $[\![\rho]\!]_q^G$ *is defined as*

$$[\![\rho]\!]_q^G \stackrel{\text{def}}{=} \{p \in P : (\phi, \psi, p) \in \rho \text{ is applicable to } q\}.$$

Henceforth, G will be assumed to be given, so I will simply write $[\![\rho]\!]_q$ to denote the set of matched principals for policy ρ and request q. I will further abbreviate this to $[\![\rho]\!]$ when q is obvious from context.

The matching of security principals through the satisfaction (or otherwise) of required and forbidden targets enables the "layout" of a relevant part of the system graph to be used to inform the request. This matching of principals does not take into consideration the requested action. Therefore, the notion of a principal may be seen as that of a label given to the subject of a request by the request's object, independent of the intended action. As previously indicated, such a notion is comparable to that of UNIX where, for example, the "owner" principal may apply independent of whether a read or write request is made.

Remark 4.2. *Path conditions are clearly closely related to regular expressions. However, for conciseness I do not include several common operators in my definition of path conditions: disjunction; conditional, which is equivalent to zero or one time; and the Kleene star, equivalent to zero or more times. Instead, I accommodate each of these operators through the use of two (or more) principal-matching rules.*[7]

The path condition $\pi^*;\pi'$, *for example, can be associated with a principal* p *by specifying the principal-matching rules* (π', none, p) *and* $(\pi^+ ; \pi', \mathsf{none}, p)$. *This approach is preferable to including these operators definitively as their inclusion would not increase the expressiveness of the policy language but would introduce greater complexity into the request evaluation process which will be discussed in Section 4.2.4.*[8]

Remark 4.3. *In some cases it is desirable for a PMP to specify a default principal* p_{def}, *much like the concept of "other" (also known as "world") in the UNIX access control system. To do so, I include the principal-matching rule* $(\mathsf{all}, \mathsf{none}, p_{\text{def}})$.

[7] Whilst I find no need within this thesis, these derived operators could be used as "syntactic sugar" to document policies using fewer rules.

[8] Specifically, I would require an additional non-deterministic finite automaton construction mechanism for each.

4.2.4 Principal Matching using NFA

Whilst Section 4.2.3 described the formal principal-matching policy components, I choose to implement the necessary processing in RPPM using non-deterministic finite automata (NFA). Briefly, I exploit the correspondence between path conditions and regular expressions to build a non-deterministic finite automaton M_π for path condition π; and I use the correspondence between labelled graphs and non-deterministic finite automata (NFA) to construct a non-deterministic finite automaton $M_{G,q}$ derived from the system graph G and information in a request q. For brevity, I will write M_q for $M_{G,q}$, as G will always be obvious from context. I use these non-deterministic finite automata to determine whether each principal-matching rule is applicable to a request, and whether its principal is, therefore, matched.

Non-Deterministic Finite Automata

I now describe the correspondence between path conditions and non-deterministic finite automata (NFA) in more detail. I also explain how to define the *request NFA* M_q given a system graph G and a request q. Finally, I explain how to construct an automaton that will determine whether a request q matches a path condition π (in the context of a system graph G).

A *non-deterministic finite automaton* is a 5-tuple $M = (Q, \Sigma, \delta, s, F)$ where [189]:

- Q is the set of *states*;
- Σ is the set of inputs (the *alphabet*);
- $\delta \subseteq Q \times Q \times \Sigma$ is the *transition relation*; and
- $s \in Q$ is a *start state* and $F \subseteq Q$ is the set of *accepting states*.

Let $\omega = \sigma_1 \ldots \sigma_\ell$, where $\sigma_i \in \Sigma$, be a *word* over the alphabet Σ. The automaton M *accepts* word ω if there exists a sequence of states, $q_0, \ldots, q_\ell$ such that:

- $s = q_0$;
- $(q_i, q_{i+1}, \sigma_{i+1}) \in \delta$ for $0 \leqslant i \leqslant \ell - 1$; and
- $q_\ell \in F$.

I write $L(M)$ to denote the set of words (or *language*) accepted by M.

Given two automata, $M_1 = (Q_1, \Sigma_1, \delta_1, s_1, F_1)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, s_2, F_2)$, the *intersection NFA* $M_\cap = (Q_1 \times Q_2, \Sigma_1 \cap \Sigma_2, \delta_\cap, (s_1, s_2), F_1 \times F_2)$ accepts the language $L(M_1) \cap L(M_2)$, [122, 127, 154], where

$$\delta_\cap = \{((q_1, q_2), (q_1', q_2'), \sigma) : (q_1, q_1', \sigma) \in \delta_1, (q_2, q_2', \sigma) \in \delta_2\}.$$

Path Conditions as NFA

I now explain how to construct an automaton for a path condition. The construction is straightforward and is based on standard techniques (see, for example, [4]), given the obvious similarities between path conditions and regular expressions. By construction, every automaton will have a single final state. Moreover, because I do not include disjunction, or the optional and Kleene star operators, in the definition of path conditions, there is a unique transition from the initial state.[9]

Proposition 4.2. *Let $r \in \widetilde{R}$, and let π and ϕ be path conditions with automata $M_\pi = (Q_\pi, \Sigma_\pi, \delta_\pi, s_\pi, \{f_\pi\})$ and $M_\phi = (Q_\phi, \Sigma_\phi, \delta_\phi, s_\phi, \{f_\phi\})$ accepting languages $L(M_\pi)$ and $L(M_\phi)$, respectively. Then:*

- $M_r = (\{s, f\}, \{r\}, \{(s, f, r)\}, s, \{f\})$;
- $M_{\pi;\phi} = (Q_{\pi;\phi}, \Sigma_\pi \cup \Sigma_\phi, \delta_{\pi;\phi}, s_\pi, \{f_\phi\})$, *where* $Q_{\pi;\phi} = Q_\pi \cup Q_\phi \setminus \{s_\phi\}$ *and*

  $$\delta_{\pi;\phi} = \delta_\pi \cup \delta_\phi \cup \{(f_\pi, q, r) : (s_\phi, q, r) \in \delta_\phi\} \setminus \{(x, y, z) \in \delta_\phi : x = s_\phi\};$$ *and*

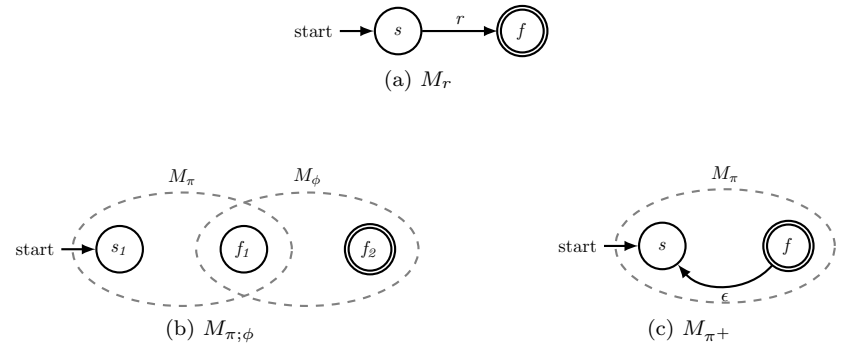
- $M_{\pi^+} = (Q_\pi, \Sigma_\pi \cup \{\epsilon\}, \delta_{\pi^+}, s_\pi, \{f_\pi\})$, *where ϵ is the empty symbol and*

  $$\delta_{\pi^+} = \delta_\pi \cup \{(f_\pi, s_\pi, \epsilon)\}.$$

The constructions of M_r, $M_{\pi;\phi}$ and M_{π^+} are illustrated in Figure 4.4.

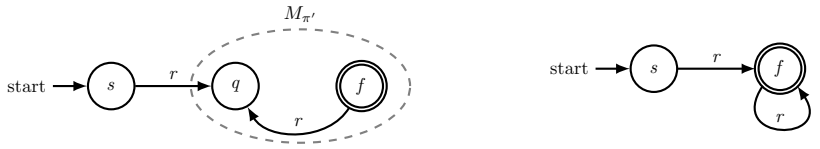
The construction of the intersection NFA is simpler if the component automata do not contain empty transitions. Accordingly, I modify the automaton for M_{π^+} to remove the empty transition. Since every automaton representing a path condition has a single transition from the initial state, I may assume that I can write any path condition $\pi \neq \diamond$ in the form $r\,;\pi'$, where $r \in \widetilde{R}$. Hence, I may represent π^+ by the automaton shown in Figure 4.5a. In the special case that $\pi = r$ for some $r \in \widetilde{R}$, $\pi' = \diamond$ and, therefore, the start and final states of π' coincide, as shown in Figure 4.5b.

[9] Recall from Remark 4.2 that whilst these three operators may be used as "syntactic sugar" they are not part of my definition of path conditions as they do not increase the expressiveness of the policy and would increase the complexity of evaluation by allowing multiple transitions from the corresponding automaton's initial state.



(a) M_r (b) $M_{\pi;\phi}$ (c) $M_{\pi+}$

Figure 4.4: Schematic representations of automata for r, $\pi\,;\phi$ and π^+



(a) $\pi = r\,;\pi'$, $\pi' \neq \diamond$ (b) $\pi = r$

Figure 4.5: Schematic representation of automata for π^+ without the empty transition

Example 4.2. *Consider the path condition*

$$\overline{\left(\overline{r_1\,;r_2^+}\right)^+\,;(r_1\,;r_3)^+}.$$

Then, I may transform this into a simple path condition using the rules in Proposition 4.1.

$$\begin{aligned}\overline{\overline{(r_1\,;r_2^+)^+\,;(r_1\,;r_3)^+} &= \overline{(r_1\,;r_3)^+}\,;\overline{\left(\overline{r_1\,;r_2^+}\right)^+}\\ &= (\overline{r_1\,;r_3})^+\,;\overline{\left(\overline{r_1\,;r_2^+}\right)}^+\\ &= (\overline{r_3}\,;\overline{r_1})^+\,;(r_1\,;r_2^+)^+\end{aligned}$$

The corresponding non-deterministic automaton is shown in Figure 4.6. Note the number of states is 5 *and the number of transitions is* 7*. In Section 4.5.2 I establish the way in which the number of states and transitions vary with the structure of* π*.*

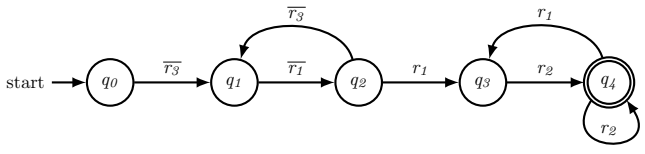


Figure 4.6: Automaton for $(\overline{r_3}\,;\overline{r_1})^+\,;(r_1\,;r_2^+)^+$

Principal Matching

The set of matched principals for a request is determined by identifying those principal-matching rules that are applicable to a given request $q = (s, o, a)$. Recall that a system graph $G = (V, E)$ contains a set of nodes V and a set of edges $E \subseteq V \times V \times \widetilde{R}$, where $\widetilde{R}$ is the set of relationship labels. Given a request $q = (s, o, a)$ and the system graph $G = (V, E)$, I define the automaton $M_q = (V, \widetilde{R}, E, s, \{o\})$. Thus, every labelled edge in G defines a transition and the start and final states are s and o, respectively.

It is trivial to decide whether a request matches the all and none targets. Hence, I focus my attention on targets that are path conditions. Informally, given a path condition π, a request q and a system graph G, I wish to find a path in the directed graph G (equivalently a word accepted by M_q) that is also a word accepted by M_π. Thus, for a principal-matching rule (ϕ, ψ, p) to be applicable to a request, where ϕ and ψ are path conditions, I require ϕ to be matched by the request and ψ to not be matched. Therefore, I compute the two intersection languages $L(M_\phi) \cap L(M_q)$ and $L(M_\psi) \cap L(M_q)$; the former must be non-empty and the latter must be empty. I test these language properties by constructing two intersection automata, one from M_ϕ and M_q, the second from M_ψ and M_q.

4.3 Compute Authorizations

4.3.1 Authorizing Principals

Having determined the set of principals which match to a request, the second step of RPPM's request evaluation process determines whether those principals are authorized to perform the requested action. This second step involves simple comparisons and set membership checks as part of evaluating the authorization policy.

Definition 4.10. *An* authorization rule *has the form* (p, X, Y, b)*, where* $p \in P$*,* $X \subseteq T \cup V$*,* $Y \subseteq A$ *and* $b \in \{0, 1\}$.[10] *Given a PMP* ρ*, an authorization rule* (p, X, Y, b) *is applicable to a request* $q = (s, o, a)$ *if all of the following conditions hold:*

- $p \in [\![\rho]\!]_q$*;*
- $X \cap \{\tau(o), o\} \neq \emptyset$*; and*
- $Y \cap \{a\} \neq \emptyset$.

[10] Recall that T is the set of entity types in the system model, V is the set of entities in the system graph and A is the set of actions.

An authorization policy *is a set of authorization rules. Given a PMP ρ, an authorization policy ϱ and a request $q = (s, o, a)$, the set of* authorization decisions $[\![\rho, \varrho]\!]_q^G$ *is defined as*

$$[\![\rho, \varrho]\!]_q^G \stackrel{\text{def}}{=} \{b \in \{0, 1\} : (p, X, Y, b) \in \varrho \text{ is applicable to } q\}.^{11}$$

The rule $(p, \{o\}, \{a\}, 1)$ indicates that principal p is authorized to perform action a on object o, while $(p, \{o\}, \{a\}, 0)$ indicates p is not authorized. A rule such as $(p, \{o\}, \{a_1, a_2\}, 1)$ can be used to authorize a principal for multiple actions, in this case a_1 and a_2, whilst the rule $(p, \{o\}, A, 1)$ can, therefore, be used to authorize a principal for all actions on a given object.

Remark 4.4. *For simplicity of notation (particularly within examples where labels may not have been assigned for the relevant sets), I will make use of the wild card character $\star$ when referring to all actions in authorization rules. Therefore, conceptually (p, X, A, b) is equivalent to $(p, X, \star, b)$, represented in notation as $(p, X, A, b) \equiv (p, X, \star, b)$.*

I can also prohibit specific actions using a negative authorization tuple. Thus, the inclusion of $(p, \{o\}, \star, 1)$ and $(p, \{o\}, \{a\}, 0)$ in the policy may authorize p for all actions on object o, *except* action a.

Should it be desired, an authorization rule may cover multiple objects. The rule $(p, \{o_1, o_2\}, \{a\}, 1)$ authorizes p for action a on objects o_1 and o_2, whilst the authorization rule $(p, \{t\}, \{a\}, 1)$ authorizes p for action a on all of the entities of type t. Broader still, the rule $(p, V, \{a\}, 1)$ authorizes p for action a on all entities within the system graph.

Remark 4.5. *Note that due to the system graph's well-formedness property (from Definition 4.2), I have $(p, V, Y, b) \equiv (p, T, Y, b)$.*

Remark 4.6. *As I did with authorization rules which control all actions (rather than a specific subset), I will make use of $\star$ when referring to all entities within authorization rules. Therefore, $(p, V, Y, b) \equiv (p, T, Y, b) \equiv (p, \star, Y, b)$.*

Example 4.3. *Returning to my motivating issue of conflicting roles from Section 3.2.2, I can envisage a higher education system that includes a PhD student, student 1, a professor, several courses (such as course 1) and some coursework answers (such as answer 1). I have arranged this mix of concrete and logical entities in the system graph fragment shown in Figure 4.7.*

[11] As with the set of matched principals, I will simply write $[\![\rho, \varrho]\!]_q$ to indicate the set of authorization decisions for policy ϱ and further abbreviate this to $[\![\rho, \varrho]\!]$ when no ambiguity will arise.

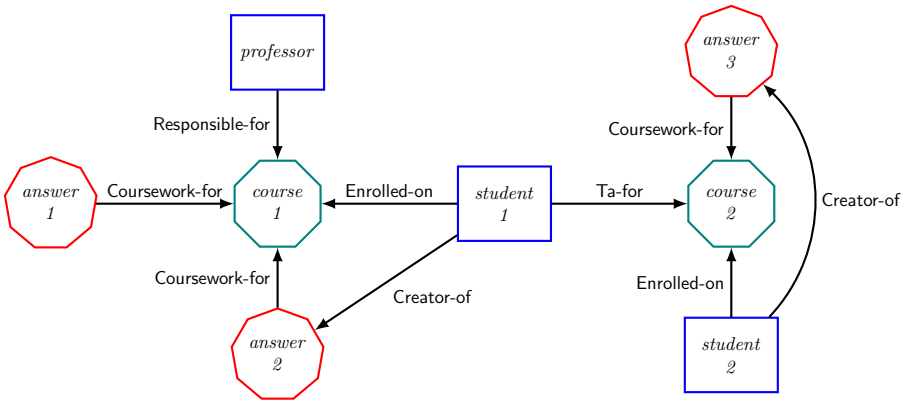


Figure 4.7: A higher education system graph fragment

It is natural that I would wish to constrain student 1 from accessing answers (other than their own) for courses on which they are enrolled as a student, whilst I would wish to grant access to those for courses for which they are a teaching assistant. To do so requires the ability to distinguish the context (in this case the course) associated with a request. I can achieve this in the RPPM model through the inclusion of logical entities in the system graph, and the specification of policies using relationships associated with those logical entities.[12]

$$\begin{aligned}\rho = \{&(\textit{Creator-of},\ \text{none},\ \textit{author}),\\ &(\textit{Ta-for}\,;\,\overline{\textit{Coursework-for}},\ \textit{Enrolled-on}\,;\,\overline{\textit{Coursework-for}},\ \textit{course-ta}),\\ &(\textit{Responsible-for}\,;\,\overline{\textit{Coursework-for}},\ \text{none},\ \textit{course-leader})\}\\ \varrho = \{&(\textit{author}, \star, \{\textit{read}, \textit{write}\}, 1),\\ &(\textit{course-ta}, \star, \{\textit{read}, \textit{grade}\}, 1),\\ &(\textit{course-leader}, \star, \{\textit{read}, \textit{review}\}, 1)\}\end{aligned}$$

Consider the request (student 1, answer 1, read), for example. There is no path in the graph between student 1 and answer 1 that matches any of the required targets in rules within ρ. Thus, the set of matched principals is empty (which will lead to the request being denied, assuming a deny-by-default discipline). On the other hand, the set of matched principals for request (student 1, answer 2, read) is {author}, since there is a path from student 1 to answer 2 with label Creator-of; hence the request will be granted (because of the first rule in ϱ). However, the set of matched principals for request (student 1, answer 3, read) is {course-ta} and the request will be permitted

[12]Note that this use of relationships and logical entities further demonstrates RPPM's ability to mitigate RBAC's wide-acting roles issue. By defining policies such as these it prevents, for example, "all" teaching assistants being able to access "all" coursework. The context of students' relationships with courses inform principal-matching through each target's path conditions.

(because of the second rule in ϱ). Note the difference in outcomes for the two requests (*student 1*, *answer 1*, *read*) *and* (*student 1*, *answer 3*, *read*) *because of the different relationships that exist between student 1 and the courses associated with the objects answer 1 and answer 3.*

Two further requests (*professor*, *answer 1*, *read*) *and* (*professor*, *answer 2*, *read*) *will result in the set of matched principals* {*course-leader*} *and these requests will be granted, whereas the set of matched principals for* (*professor*, *answer 3*, *read*) *is empty (and the request will be denied). Again, the professor's relationship with the two courses determines the principals (and thus decisions) associated with the respective requests.*

Whilst this example's requests have either resulted in an empty or singleton set of matched principals, policies may equally produce larger sets of matched principals. For example, if the system graph of Figure 4.7 included the edge (*professor*, *student 1*, *Mentor-for*) *and the principal-matching policy ρ included a rule* (*Mentor-for* ; *Creator-of*, none, *mentor*)*, then the request* (*professor*, *answer 2*, *read*) *would result in the set of matched principals* {*course-leader*, *mentor*}*.*

4.3.2 Alternative Authorization Processes

Whilst the compute authorizations step of request evaluation is able to produce a set of authorization decisions for a request, the partition between the two request evaluation steps enables alternative authorization processes to be employed, should that be desired. Specifically, any alternative mechanism of determining decisions could be used in place of compute authorizations, as long as the mechanism was able to complete it's processing using a set of matched principals as input.

For example, the set of matched principals from the compute principals step could be passed to XACML, or another ABAC model, where it could be treated as an attribute, or a collection of attributes. The relationship-derived principals would then inform an ABAC-based authorization decision, alongside other attributes within that protection system.

In Chapter 9 I will describe a similar mechanism where the set of matched principals is passed to another RPPM system graph. This enables the request evaluation in one system graph to be informed by the relationships in another.

4.4 Deterministic Authorization Policy

As each authorization rule has a binary decision outcome, the set of authorization decisions which results from compute authorizations may take one of four possible

values: $\emptyset$; $\{0\}$; $\{1\}$; or $\{0, 1\}$. Whilst the second and third of these values results in a single authorization decision, deny or allow respectively, the first and fourth value are inconclusive on their own.

A set of authorization decisions equal to $\emptyset$ indicates that the authorization policy does not specify whether or not any of the matched principals is authorized to perform the requested action. In contrast, a value of $\{0, 1\}$ indicates that there are at least two rules with conflicting decisions. For systems where a single authorization decision must be guaranteed (a deterministic policy [180]), the following conflict resolution and default decision functions are provided. Proof of the determinism of these features is given in Section 4.4.3.

4.4.1 Conflict Resolution

The authorization rule $(p, \{o\}, \{a\}, 0)$ explicitly disallows p from performing action a on object o, while $(p, \{o\}, \{a\}, 1)$ explicitly allows it. Accordingly, I define an *extended* authorization policy to be a pair (ϱ, χ), where ϱ is a set of authorization rules and χ is a *conflict resolution strategy* (CRS) which is used to reduce the set of matching decisions to a single decision.[13] That is, $[\![\rho, (\varrho, \chi)]\!] \in \{\emptyset, \{0\}, \{1\}\}$. In the interests of brevity, I will continue to write $[\![\rho, \varrho]\!]$ in preference to $[\![\rho, (\varrho, \chi)]\!]$.

I present two conflict resolution strategies which may be employed within RPPM:[14]

- DenyOverrides – reduces a value of $\{0, 1\}$ for the set of authorization decisions to $\{0\}$, leaving all other values unchanged.
- AllowOverrides – reduces a value of $\{0, 1\}$ for the set of authorization decisions to $\{1\}$, leaving all other values unchanged.

4.4.2 Default Decisions

In some cases a default access control decision (allow or deny) needs to be specified in the event that no authorization rules apply to a request. Systems may need to support allow-by-default when the system enters an emergency state, such as the opening of fire exit doors when there is a fire. Other circumstances will commonly require fail-safe handling, where a deny-by-default strategy is implemented in order

[13] For avoidance of doubt, conflict resolution strategies are distinct from (and do not compete with) the required and forbidden targets of the principal-matching rules. Those targets define which paths are required and forbidden for an individual principal to be matched to a request. These strategies determine which authorization rule outcomes, those granting or denying the requested action, should be prioritised when determining an ultimate decision.

[14] Whilst I do not define a CRS of DenyUnlessPermit (as defined in [141]) this can be achieved by using AllowOverrides with a deny-by-default strategy (see Section 4.4.2). In the case of PermitUnlessDeny this can be achieved by using DenyOverrides with an allow-by-default strategy.

to ensure no unauthorised access is allowed. Some systems may be deemed so sensitive that there may be no conditions under which allow-by-default would be enabled.

There are two circumstances in the RPPM model when default decision-making applies. The first is when no matched principals are identified ($[\![\rho]\!] = \emptyset$), whilst the second is when there are no explicit authorizations ($[\![\rho, \varrho]\!] = \emptyset$). Accordingly, I allow for a default decision to be applied at one of the following levels: default-per-subject, default-per-object, default-per-type or system-wide default. The default-per-subject decision is only applied when there are no matched principals.[15]

Definition 4.11. *Given a system graph $G = (V, E)$, $V_{so} \subseteq V \times \{\mathsf{sub}, \mathsf{obj}\}$, a set of types T, and a request $q = (s, o, a)$, a* default decision function

$$\gamma : V_{so} \cup T \cup \{\mathsf{sys}\} \to \{\bot, 0, 1\}$$

is a function which maps entities within the system graph to default decisions for requests, where $\bot$ is undefined, 0 is deny and 1 is allow.[16] *The function maps default decisions on a per subject (u, sub), per object (u, obj), per object type $t \in T$ and system-wide basis. When initialized, $\gamma(\mathsf{sys}) = 0$ and the function maps all other inputs to $\bot$ until they are configured.*

The four defaults are evaluated in order, where defined (i.e., where not set to $\bot$), with the first applicable default determining the authorization decision. In this way, if there is a default $\gamma(s, \mathsf{sub})$ defined for the subject s of the request (s, o, a), the subject's default (allow or deny) applies. If no subject default is defined for s, then the default $\gamma(o, \mathsf{obj})$ for the object o of the request shall apply, if defined. If there is no subject default for s and no object default for o, then the default $\gamma(\tau(o))$ for the type of object $\tau(o)$ shall apply, if defined. If none of these defaults are defined, then the system-wide default $\gamma(\mathsf{sys})$ shall apply.[17]

4.4.3 Proof of Determinism

The determinism of RPPM's authorization policy when a conflict resolution strategy and a default decision function are employed is simple to prove.

Proof. The compute principals step of request evaluation produces a set of matched

[15] Default-per-subject is not applied when there are no explicit authorizations: when the set of possible decisions is determined, the subject is no longer relevant, having been used to identify the appropriate matched principals.

[16] Although I do not allow $\gamma(\mathsf{sys})$ to be set to $\bot$.

[17] As an alternative, a system-wide default may be achieved using the default principal p_{def} as described in Remark 4.3. I have not relied upon this here as I introduce a comparable administrative default decision function in Chapter 8, and both system-wide defaults cannot be achieved with a single default principal.

principals

$$[\![\rho]\!]_q^G \stackrel{\text{def}}{=} \{p \in P : (\phi, \psi, p) \in \rho \text{ is applicable to } q\}.$$

Given any system graph G and any request $q = (s, o, a)$, if $[\![\rho]\!]_q^G = \emptyset$, then the default decision function processing occurs. In the trivial case, $\gamma(s, \mathsf{sub}) = \perp$, $\gamma(o, \mathsf{obj}) = \perp$, $\gamma(\tau(o)) = \perp$, and $\gamma(\mathsf{sys}) \in \{0, 1\}$. Therefore, a single authorization decision results. In the non-trivial case, one or more of $\gamma(s, \mathsf{sub})$, $\gamma(o, \mathsf{obj})$ and $\gamma(\tau(o))$ is equal to 0 or 1. Whatever combination, a single authorization decision results as these defaults are evaluated in order, where defined, with the first applicable default determining the authorization decision.

If $[\![\rho]\!]_q^G \neq \emptyset$, then compute authorization step of request evaluation is performed, producing a set of authorization decisions

$$[\![\rho, \varrho]\!]_q^G \stackrel{\text{def}}{=} \{b \in \{0, 1\} : (p, X, Y, b) \in \varrho \text{ is applicable to } q\}.$$

If $[\![\rho, \varrho]\!]_q^G = \emptyset$, then once again default decision function processing occurs. In the trivial case, $\gamma(o, \mathsf{obj}) = \perp$, $\gamma(\tau(o)) = \perp$, and $\gamma(\mathsf{sys}) \in \{0, 1\}$. Therefore, a single authorization decision results. In the non-trivial case, one or both of $\gamma(o, \mathsf{obj})$ and $\gamma(\tau(o))$ is equal to 0 or 1. Whatever combination, a single authorization decision results as these defaults are evaluated in order, where defined, with the first applicable default determining the authorization decision.

If $[\![\rho, \varrho]\!]_q^G \neq \emptyset$, then $[\![\rho, \varrho]\!]_q^G \in \{\{0\}, \{1\}, \{0, 1\}\}$. If $[\![\rho, \varrho]\!]_q^G \in \{\{0\}, \{1\}\}$, then a single authorization decision results. If $[\![\rho, \varrho]\!]_q^G = \{0, 1\}$, then conflict resolution processing occurs and $[\![\rho, \varrho]\!]_q^G$ is reduced to $\{0\}$ in the case of DenyOverrides or $\{1\}$ in the case of AllowOverrides. In either case, a single authorization decision results. □

4.5 Request Evaluation

4.5.1 End-to-End Process

Figure 4.8 provides a detailed architecture of the complete RPPM request evaluation process, indicating the inputs necessary and decisions employed. Recall, the first step of request evaluation, compute principals, is rather complex and, conceptually, requires the identification of paths within the system graph in order to determine the principals which match a request. However, the second step, compute authorizations, involves simple lookups to determine whether the matched principals for a request are authorized to perform the requested action on the object.

Pseudo-code for the entire request evaluation process is shown in Algorithm 4.1.

I determine the set of matched principals using Algorithm 4.2

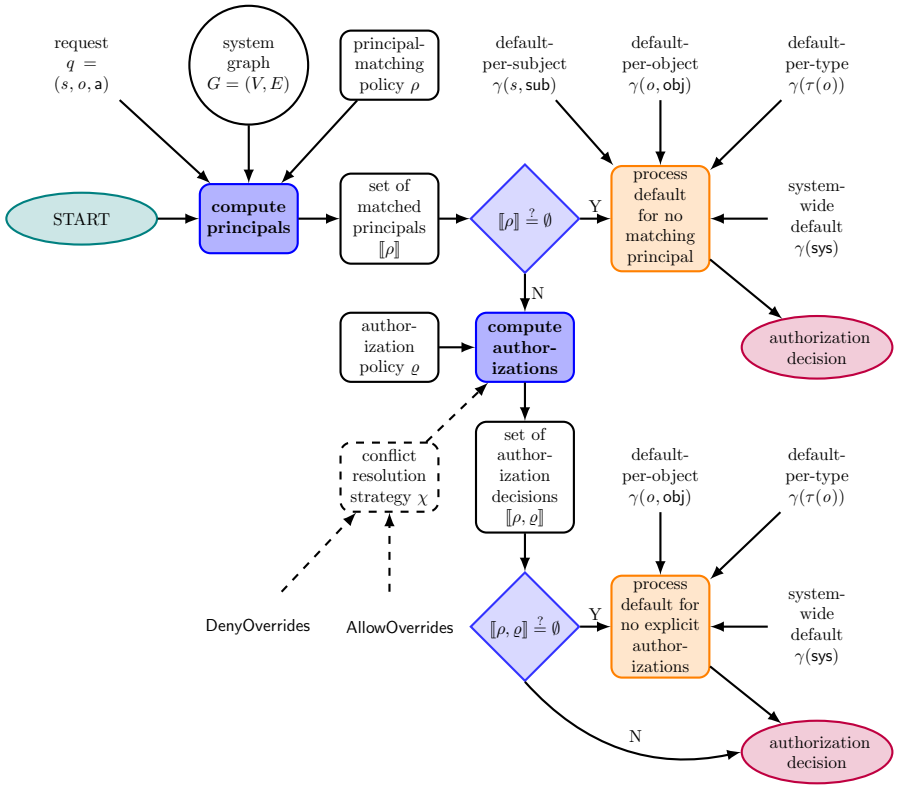


Figure 4.8: Request evaluation detailed architecture

Algorithm 4.1 RequestEvaluation

Require: System graph $G = (V, E)$, set of relationship labels $\widetilde{R}$, set of types T, request $q = (s, o, a)$, principal-matching policy ρ, default decision function γ and extended authorization policy (ϱ, χ)
Ensure: Returns authorization decision

```
1:  [[ρ]] ← ComputePrincipals(G, R̃, q, ρ)
2:  if [[ρ]] = ∅ then
3:      [[ρ, ϱ]] ← ApplyDefaults(γ, o, s)
4:  else
5:      [[ρ, ϱ]] ← ComputeAuthorizations(q, (ϱ, χ), [[ρ]])
6:      if [[ρ, ϱ]] = ∅ then
7:          [[ρ, ϱ]] ← ApplyDefaults(γ, o)
8:      end if
9:  end if
10: if [[ρ, ϱ]] = {0} then
11:     return false // deny
12: else if [[ρ, ϱ]] = {1} then
13:     return true // allow
14: end if
```

(ComputePrincipals). This automata language evaluation process described in Section 4.2.4 is used in lines 4 and 5.

Having identified the set of matched principals, the set of authorizations can be easily determined from the applicable authorization rules (as per Definition 4.10) using Algorithm 4.3 (ComputeAuthorizations). This process is far simpler than that of Algorithm 4.2, being limited to simple comparisons and set membership checks.

Algorithm 4.2 ComputePrincipals

Require: System graph $G = (V, E)$, set of relationship labels $\widetilde{R}$, request $q = (s, o, a)$ and principal-matching policy ρ
Ensure: Returns set of matched principals $[\![\rho]\!]$
1: $[\![\rho]\!] \leftarrow \emptyset$
2: $M_q \leftarrow (V, \widetilde{R}, E, s, \{o\})$
3: **for** $(\phi, \psi, p) \in \rho$ **do**
4: **if** $(\phi = \mathsf{all})$ **or** $(L(M_\phi) \cap L(M_q) \neq \emptyset)$ **then**
5: **if** $(\psi = \mathsf{none})$ **or** $(L(M_\psi) \cap L(M_q) = \emptyset)$ **then**
6: $[\![\rho]\!] \leftarrow [\![\rho]\!] \cup p$
7: **end if**
8: **end if**
9: **end for**
10: **return** $[\![\rho]\!]$

Lines 7 through 11 of Algorithm 4.3 provide for conflict resolution, should it be required, as described in Section 4.4.1.

Algorithm 4.3 ComputeAuthorizations

Require: Request $q = (s, o, a)$, extended authorization policy (ϱ, χ) and set of matched principals $[\![\rho]\!]$
Ensure: Returns set of authorization decisions $[\![\rho, \varrho]\!]$
1: $[\![\rho, \varrho]\!] \leftarrow \emptyset$
2: **for** $(p, X, Y, b) \in \varrho$ **do**
3: **if** $(p \in [\![\rho]\!])$ **and** $((o \in X)$ **or** $(\tau(o) \in X))$ **and** $(a \in Y)$ **then**
4: $[\![\rho, \varrho]\!] \leftarrow [\![\rho, \varrho]\!] \cup b$
5: **end if**
6: **end for**
7: **if** $[\![\rho, \varrho]\!] = \{0, 1\}$ **then**
8: **if** $\chi =$ DenyOverrides **then**
9: $[\![\rho, \varrho]\!] \leftarrow \{0\}$
10: **else if** $\chi =$ AllowOverrides **then**
11: $[\![\rho, \varrho]\!] \leftarrow \{1\}$
12: **end if**
13: **end if**
14: **return** $[\![\rho, \varrho]\!]$

In cases where default decisions processing is desired, a decision is determined using Algorithm 4.4 (ApplyDefaults).[18]

Algorithm 4.4 ApplyDefaults

Require: Default decision function γ, object o, and (optionally) subject s
Ensure: Returns a single element set of authorization decisions, $\{0\}$ or $\{1\}$
1: **if not** s is nothing **then**
2: **if** $\gamma(s, \mathsf{sub}) \neq \perp$ **then**
3: **return** $\emptyset \cup \gamma(s, \mathsf{sub})$
4: **end if**
5: **else if** $\gamma(o, \mathsf{obj}) \neq \perp$ **then**
6: **return** $\emptyset \cup \gamma(o, \mathsf{obj})$
7: **else if** $\gamma(\tau(o)) \neq \perp$ **then**
8: **return** $\emptyset \cup \gamma(\tau(o))$
9: **else**
10: **return** $\emptyset \cup \gamma(\mathsf{sys})$
11: **end if**

[18] Note that in Algorithm 4.4 I return the union of the default decision with the empty set in order to return a single element set containing the decision.

4.5.2 Complexity

The algorithms ComputeAuthorizations and RequestEvaluation do not involve significant computation. The worst case time complexity of request evaluation is, therefore, dominated by the complexity of ComputePrincipals, which is dependent on two things:[19]

- The number of principal-matching rules to be evaluated; and
- The complexity of determining whether the intersection NFA accept non-empty languages.

To evaluate the second of these factors, I define the *length* $\ell(\pi)$ of a (simple) path condition π to be:

- $\ell(\pi) = 1$ if $\pi = r$ for some $r \in \widetilde{R}$;
- $\ell(\pi\,;\pi') = \ell(\pi) + \ell(\pi')$; and
- $\ell(\pi^+) = \ell(\pi)$.

In other words, $\ell(\pi)$ is simply the number of occurrences of elements in $\widetilde{R}$ in π. I now consider the *size*, number of states and transitions, of the automaton M_π.

Proposition 4.3. *Let $r \in \widetilde{R}$, π and ϕ be path conditions. Then:*

- $|Q_r| = 2$ *and* $|\delta_r| = 1$ *for* $r \in \widetilde{R}$*;*
- $|Q_{\pi;\phi}| = |Q_\pi| + |Q_\phi| - 1$ *and* $|\delta_{\pi;\phi}| = |\delta_\pi| + |\delta_\phi|$*; and*
- $|Q_{\pi^+}| = |Q_\pi|$ *and* $|\delta_{\pi^+}| = |\delta_\pi| + 1$.

Proof. The proof follows immediately by inspection of the automata in Figure 4.4. □

Corollary 4.1. *Let π be a simple path condition and let $\vartheta(\pi)$ denote the number of occurrences of $+$ in π. Then for path condition π, $|Q_\pi| = \ell(\pi) + 1$ and $|\delta_\pi| = \ell(\pi) + \vartheta(\pi)$.*

Proof. The result may be proved by a straightforward induction on the structure of π. Clearly, the result for $|Q_\pi|$ holds for path condition $\pi = r$, $r \in \widetilde{R}$. Now consider path condition $\pi\,;\phi$ and assume the result holds for π and ϕ. Then

$$|Q_{\pi;\phi}| = |Q_\pi| + |Q_\phi| - 1 = (\ell(\pi) + 1) + (\ell(\phi) + 1) - 1 = \ell(\pi\,;\phi) + 1,$$

[19] The automata for path conditions contained in rules in the PMP can be pre-computed once and then used, as required, to construct the intersection automata.

as required. Finally, consider path condition π^+ and assume the result holds for π. Then

$$|Q_{\pi^+}| = |Q_\pi| = \ell(\pi) + 1 = \ell(\pi^+) + 1.$$

Similarly, the result for $|\delta_\pi|$ holds for path condition $\pi = r$. Now consider path condition $\pi\,;\phi$ and assume the result holds for π and ϕ. Then

$$|\delta_{\pi;\phi}| = |\delta_\pi| + |\delta_\phi| = \ell(\pi) + \vartheta(\pi) + \ell(\phi) + \vartheta(\phi) = \ell(\pi\,;\phi) + \vartheta(\pi\,;\phi).$$

Finally, consider π^+ and assume the result holds for π. Then

$$|\delta_{\pi^+}| = |\delta_\pi| + 1 = \ell(\pi) + \vartheta(\pi) + 1 = \ell(\pi^+) + \vartheta(\pi^+).$$

□

The complexity of computing the intersection NFA for automata M and M' is determined by the size of the respective transition relations, since I compute a product automaton whose transition relation is determined by the transition relations of the component automata. In the worst case, the size of an automaton's transition relation $\delta \subseteq Q \times Q \times \Sigma$ is $O(|Q|^2 \cdot |\Sigma|)$. The size of the transition relation in M_q is, therefore, $O(|V|^2 \cdot |\widetilde{R}|)$. However, in the case of my path condition automata, I have $|\delta_\pi| = \ell(\pi) + \vartheta(\pi)$.

Thus, the overall complexity of evaluating a path condition π with respect to a request and system graph G is $O((\ell(\pi) + \vartheta(\pi)) \cdot |V|^2 \cdot |\widetilde{R}|)$. Each principal-matching rule contains at most two path conditions as targets. And each principal-matching rule in policy ρ must be evaluated. Thus the overall complexity of evaluating a request is

$$O(|\rho| \cdot \vartheta(\rho) \cdot |V|^2 \cdot |\widetilde{R}|), \text{ where } \vartheta(\rho) = \max\{\ell(\pi) + \vartheta(\pi) : \pi \in \rho\}.$$

4.6 Model Comparisons

4.6.1 Relationship-Based Access Control

The RPPM$_0$ model is designed as a relationship-based access control model able to support general computing applications. It, therefore, does not suffer most of the limitations of other relationship-based access control models (as identified in Section 3.2.4). Table 4.1 shows a comparison of a variety of relevant features between RPPM$_0$ and the relationship-based access control models previously discussed [41, 43, 44, 51, 52, 82, 85]. Whilst this base version of RPPM limits path evaluation

to between the subject and object of the request and cannot support conjunction, RPPM$_0$ has numerous benefits over other models.

Generality Its ability to support any entity and relationship types within the system graph makes it able to describe any system desired. The policy rule targets are defined using paths which may comprise multiple distinct relationship types and may identify paths of any length satisfied through any entities; this way the traversed entities do not need to be identified during policy creation (explicitly or via variables), and paths are not restricted to avoid cycles. Further, its policies may be defined using paths which exist between any entity types, and so may control actions undertaken by any system entity on any other system entity. Specifically, there is no constraint on user involvement and so the model may equally be used to manage access by autonomous components, automated agents, or even inanimate objects should that be desired for a particular application. None of the other relationship-based access control models offer this degree of generality.

Abstraction and Conciseness Many relationship-based access control models directly bind permissions to users through paths of relationships. In contrast, RPPM$_0$ abstracts the permission assignment away from requesting entities using principals. This enables a far more flexible model, whereby changes may be made to the system graph without the need to modify policy. Further, policy rules may cover large numbers of (unidentified) entities, greatly reducing the administrative burden of managing the policy.[20] For example, a single rule may apply to all entities, all entities of a specific subset of types, all entities of a particular type, a specific subset of entities, or a single entity. Once again, none of the other relationship-based access control models offer such features.

Versatility By supporting two targets in the principal-matching rules, RPPM$_0$ may match principals when a path of relations exists between the subject and the object, or only when such a path is absent. As well as matching principals based on paths in the system graph, these rules allow the use of the special targets all and none which together are matched for all requests (thus enabling the creation of a default principal). Only Carminati *et al.*'s semantic web model offers both positive and negative rules [43]; however, they do not have concepts equivalent to my special targets as they directly assign permissions to users.

[20] As entities aren't specifically identified in policy rules, the rules may be applicable to entities present in the system graph at the time the rules were created and also entities which are added afterwards.

Completeness Whilst the rules of RPPM$_0$ enable principals to be matched and authorization decisions to be made in a wide range of circumstances, I also cater for circumstances which result in either multiple conflicting authorization rules applying or none. Conflict resolution strategies resolve conflicting rules, enabling either a DenyOverrides or AllowOverrides mode of operation to apply. If no authorization rules apply, either because no principals were matched or those that were do not lead to a decision, then the default decision function ensures that an authorize or deny outcome is determined. The default decision function's system entry enables the model to apply an allow-by-default or deny-by-default strategy.

In comparison, Carminati *et al.*'s social network model employs a deny-by-default strategy, but it doesn't support negative rules and so cannot produce conflicts [44]. The same is true of the ReBAC model of Fong (and Siahaan) and Bruns *et al.* [41, 82, 85]. Carminati *et al.*'s semantic web model implements the equivalent of DenyOverrides and deny-by-default, with access only granted if an authorization exists but no prohibitions exist [43]. Cheng *et al.*'s U2R model allows the resolution of conflicts between policies using disjunction, conjunction or prioritisation. It employs a deny-by-default strategy alongside this [51, 52].

One limitation of RPPM$_0$ is the fact that the targets within principal-matching rules do not support conjunction of multiple paths. Whilst a rule may require one path of relations and forbid another, targets cannot be constructed from multiple paths at present. I will introduce an enhancement in Chapter 5 which will enable RPPM to support conjunction. The other limitation is that the targets may only be satisfied between the subject and object of the request. I will introduce a further enhancement, in Chapter 7, which will rectify this limitation.

4.6.2 Implementing Non-ReBAC Models

The base RPPM model (RPPM$_0$) provides a sufficiently general model of access control that it may be used to implement other existing access control models. Specifically, it is possible to construct an RPPM instance with the appropriate system model, system graph, and policies such that authorization requests evaluated by that RPPM instance would be authorized, or denied, in the same manner as those within the implemented access control model. Whilst I shall provide a clear explanation of how such implementations may be constructed for popular models (so that the reader may get an intuition of the flexibility and power of RPPM), I shall not prove their equivalence formally.

	Social Network Model [44]	**Semantic Web [43]**	**ReBAC [41, 82, 85]**	**U2R [51, 52]**	**RPPM$_0$**
Graph Entities	Users	Users and resources	Users	Users, objects, policies and sessions	$\star$
Graph Relations	$\star$	$\star$	$\star$	$\star$	$\star$
Graph Limitations	No resources	None	No resources	No cycles	None
Resource Relations	Ownership	$\star$	Ownership	$\star$	$\star$
Path Evaluation Between (Types)	Users and Users	Users and Resources	Users and Users	Users and $\star$	$\star$ and $\star$
Path Evaluation Between (Entities)	Requestor and resource owner	Subject and object	Requester and resource owner	A pair from requester, target and controlling user	Subject and object
Multiple Distinct Relations in Paths	No	Yes	Yes	Yes	Yes
Policy Language	Conditions over relation, max length and min strength	Atoms over type of resource, direct relation, and same or different entity	Propositions over direct relation, variables and nominals	Regular expressions over relations	Regular expressions over relations
Paths in Rules	Positive conjunction of conditions	Positive conjunction of atoms	Positive and negative, conjunction and disjunction of propositions	Positive and negative, conjunction and disjunction of expressions	Single positive and single negative path
Rule Types	Positive only	Positive and negative	Positive only	Positive only	Positive and negative
Rule Limitations	Single relation within exact path	Exact path via pre-determined objects	Defined per resource	Path length limited by hopcount	Lack of conjunction
Evaluation Basis	CWM reasoner [29]	SweetRules reasoner [92]	Depth-first search model checker	Depth-first search with limited depth	Regular language emptiness evaluation

Key: The wild card character $\star$ is used in this table to indicate "anything" or "all".

Table 4.1: Relationship-based access control model comparisons with RPPM$_0$

UNIX

Whilst RPPM$_0$ is able to define the necessary access control model structure and policies, the set-based authorization policy evaluation of RPPM$_0$ is unable to enforce UNIX's ordered request evaluation process, whereby the "owner" permissions apply, or the "group" permissions apply, or the "other" permissions apply. RPPM$_0$ is, therefore, unable to implement the UNIX access control model. However, I shall

introduce an enhancement in Chapter 5 which will enable RPPM to fully implement UNIX's access control model (see Section 5.2.2).

Multi-Level Security

In contrast, RPPM$_0$ is able to implement and enforce basic multi-level security policies ensuring no direct compromise, or "reading up", of labelled objects. To achieve this, the model is configured such that the security levels are logical entities within the system graph, alongside concrete entities representing the users and objects. The clearances of users are represented by the presence of a relationship between each user and the security level representing their maximum clearance. Similarly, the classifications of objects are represented by the presence of an equivalent relationship between each object and the security level representing its classification. This arrangement is depicted, conceptually, in Figure 4.9.[21]

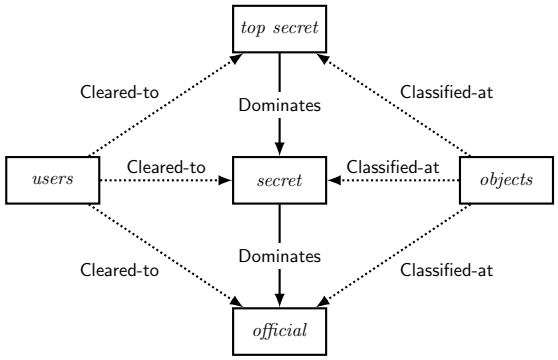


Figure 4.9: Multi-level security generalisation

More formally, an RPPM$_0$ model configured as follows would grant actions to cleared-users where the maximum clearance of the user is at least the classification of the object.

$$
\begin{aligned}
T &= \{users, levels, objects\} \\
\widetilde{R} &= \{\mathsf{Cleared\text{-}to}, \mathsf{Classified\text{-}at}, \mathsf{Dominates}, \\
&\quad \overline{\mathsf{Cleared\text{-}to}}, \overline{\mathsf{Classified\text{-}at}}, \overline{\mathsf{Dominates}}\} \\
S &= \emptyset \\
E_{\mathrm{PR}} &= \{(users, levels, \mathsf{Cleared\text{-}to}), (objects, levels, \mathsf{Classified\text{-}at}), \\
&\quad (levels, levels, \mathsf{Dominates})\} \\
G_{\mathrm{PR}} &= (T, E_{\mathrm{PR}})
\end{aligned}
$$

[21] Figure 4.9 does not show a system graph, it shows a high-level representation of the "shape" of a system graph.

$$
\begin{aligned}
V &\supseteq \{\mathit{official}, \mathit{secret}, \mathit{top\ secret}\} \\
\tau(\mathit{official}) &= \mathit{levels} \\
\tau(\mathit{secret}) &= \mathit{levels} \\
\tau(\mathit{top\ secret}) &= \mathit{levels} \\
E &\supseteq \{(\mathit{top\ secret}, \mathit{secret}, \mathsf{Dominates}), (\mathit{secret}, \mathit{official}, \mathsf{Dominates})\} \\
G &= (V, E) \\
P &= \{\mathsf{cleared\text{-}user}\} \\
\rho &= \{(\mathsf{Cleared\text{-}to}\ ;\ \overline{\mathsf{Classified\text{-}at}},\ \mathsf{none},\ \mathsf{cleared\text{-}user}), \\
&\quad\ (\mathsf{Cleared\text{-}to}\ ;\ \mathsf{Dominates}^{+}\ ;\ \overline{\mathsf{Classified\text{-}at}},\ \mathsf{none},\ \mathsf{cleared\text{-}user})\} \\
(\varrho, \chi) &= (\{(\mathsf{cleared\text{-}user}, \{\mathit{objects}\}, \star, 1)\}, \mathsf{DenyOverrides}) \\
\gamma(\mathit{objects}) &= 0 \\
\gamma(\mathsf{sys}) &= 0
\end{aligned}
$$

The model specification should be self-explanatory; there are three entity types T as previously mentioned and three relationships (and their inverses) $\widetilde{R}$ which are limited to particular pairs of those entity types G_{PR}. The system graph $G = (V, E)$ of the implementation must include the (three) security levels which are ordered using the Dominates relation.[22] The principal-matching policy employs a cleared-user principal which is matched in two rules where the required path condition is based on the possible paths of relations between users and objects for which they have clearance. The authorization policy grants all actions[23] and the default decision function is configured to deny access for requests made targeting objects if no principals are matched.[24]

Whilst, using this approach, it is possible to enforce Bell and LaPadula's simple security property using RPPM$_0$ [25], the model cannot support the dynamic session awareness required to implement the *-property which prevents potential future compromises associated with "writing down" [23]. However, I shall introduce an enhancement in Chapter 6 which will enable RPPM to implement the awareness necessary to prevent a cleared-user from writing classified material to a lower classification object (see Section 6.2.2).

[22] In reality, the relation simply links the security levels, the ordering comes about because the second principal-matching rule includes one or more instances of the relation as part of required paths.

[23] Obviously a more refined authorization policy could be employed here should it be necessary.

[24] Whilst $\gamma(\mathsf{sys})$ is sufficient here, I've included $\gamma(\mathit{objects})$ for clarity of the example.

RBAC

RPPM$_0$ may also be configured to implement core RBAC [108]. However, given that there are no generic roles applicable to all systems (unlike the consistent security levels of MLS) I shall describe the implementation procedurally rather than explicitly.

I assume the set of entities is the disjoint union of users, roles, permissions and objects. I, initially, employ two types of relationship, the User-role relationship, referred to as user assignment and abbreviated ua, along with the Role-permission relationship, referred to as permission assignment and abbreviated pa. The principal-matching policy is then defined with each rule having the form $(ua; pa, \mathsf{none}, p)$ where the principal p has the same name as the permission identified by the pa edge. The authorization policy contains elements $(p, \{ob\}, \{op\}, 1)$ which map the principals to objects, allowing them operations (as per the permission binary relation in RBAC).

Further, I can introduce the Role-role relationship (abbreviated rr) in order to extend this configuration to implement a role hierarchy. Finally, I may also introduce the User-permission relationship (abbreviated up), in order to articulate exceptions to the basic RBAC model by directly associating permissions with users.

4.7 Summary

I have introduced a base RPPM model (RPPM$_0$) which is more broadly applicable and less constrained than other existing relationship-based access control models. I have also demonstrated how it may be used to implement several popular access control models which are not based on relationships, specifically multi-level security and RBAC. Whilst I have also shown that RPPM$_0$ provides numerous features beyond those of the relationship-based access control models discussed in Section 3.2.4, there are further features which are desirable for an access control model for general computing applications. Through the remainder of this thesis I will introduce enhancements to the base RPPM model.

Chapter 5

$\mathbf{RPPM_{1a}}$

In this chapter I shall introduce RPPM_{1a}, an RPPM model containing a number of enhancements over the base model (RPPM_0). Specifically, RPPM_{1a} includes the following *request evaluation enhancements*:

- Policy graph evaluation – this enhancement enables RPPM to support principal-matching based on conjunction of paths and, as a corollary, enables *list-oriented* policies (such as those used by UNIX). Support for conjunction is based on a generic mechanism called *principal activation*, whereby the matching of one (or more) principal(s) leads to the matching of others.
- Target-based request evaluation – this enhancement is a minor optimisation of the compute principal step which enables RPPM to only process those principal-matching rules which can affect the result of the request's evaluation.
- Caching edges – this enhancement is a major optimisation which enables RPPM to completely bypass the compute principal step of request processing if a request by the same subject on the same object has previously been evaluated in full.

The basis for the policy graph evaluation and target-based request evaluation enhancements described in Sections 5.1.1 and 5.1.2 has previously been published, in conjunction with Jason Crampton, in [63]. The caching edges enhancement of Section 5.1.3 has been published likewise in [60, 61, 63].

5.1 Request Evaluation Enhancements

5.1.1 Policy Graph Evaluation

As discussed in Section 4.2.3, RPPM$_0$ matches principals to a request by evaluating rules within a principal-matching policy (PMP). Each of these principal-matching rules makes use of two targets, one of which is required and one of which is forbidden. Disjunction, where a principal is activated if at least one of several path conditions is satisfied, is supported in RPPM$_0$ through the use of multiple principal-matching rules for the same principal (as described in Remark 4.2). However, there may also be times when there is a need to match a security principal only if each one of several path conditions is satisfied. The basic RPPM policy model described in Chapter 4 does not support such conjunction requirements.

Hence, I introduce the idea of a *policy graph*. I arrange the rules in a PMP as a directed acyclic graph, making the process of matching principals more like the evaluation of XACML policies [141].[1]

Definition 5.1. *Given a system graph $G = (V, E)$ and a set of principals P, a* policy graph *$G_\rho = (V_\rho, E_\rho)$ is a directed acyclic graph with a unique root (of in-degree 0) such that each vertex is a principal-matching rule. The set of principals includes a special principal called the* null *principal and the principal-matching rule for the root is defined to be* (all, none, null).

Definition 5.2. *Given a policy graph $G_\rho = (V_\rho, E_\rho)$, if there exist edges*

$$(v_{\rho_1}, v_{\rho_2}), (v_{\rho_2}, v_{\rho_3}), \ldots, (v_{\rho_{k-1}}, v_{\rho_k})$$

in E_ρ then I say v_{ρ_1} is an ancestor *of v_{ρ_i}, for any i, $1 < i \leqslant k$, and I will write $v_{\rho_1} > v_{\rho_i}$. I also say v_{ρ_i} is the* parent *of $v_{\rho_{i+1}}$.* Descendent *and* child *are, naturally, the inverse of ancestor and parent, respectively.*

Informally, the policy graph allows the applicability of one principal-matching rule to "trigger" other descendent principal-matching rules (the edges in the policy graph determining which rules trigger other rules). More formally, the evaluation of a policy graph with respect to a request, in order to compute a set of matched principals, is performed as per Algorithm 5.1.[2] This "variant" of the ComputePrinci-

[1] Recall from Section 4.3.2 that the partition between compute principals and compute authorizations enables an alternative authorization process, such as XACML, to be employed to make the ultimate authorization decision. Even so, a policy graph is still necessary to enable the conjunction of path conditions to be used as targets when matching a principal. Additionally, I will use policy graphs in Chapter 9 when evaluating requests across multiple system graphs.

[2] Note that I do not add the null principal to the set of matched principals (line 7) so that it does not interfere with authorization decisions.

pals algorithm may be employed in place of Algorithm 4.2 with only minor changes to the calling arguments. Firstly, Algorithm 5.1 takes a policy graph rather than a principal-matching policy; secondly, it takes a new argument called the principal-matching strategy σ. I present two principal-matching strategies which may be employed within RPPM:

- AllMatch – the set of matched principals contains all of those principals matched during a complete evaluation of the policy graph.
- FirstMatch – the set of matched principals contains only the first principal matched when evaluating rules within the policy graph.[3]

Algorithm 5.1 ComputePrincipals (policy graph variant)

Require: System graph $G = (V, E)$, set of relationship labels $\widetilde{R}$, request $q = (s, o, a)$, policy graph $G_\rho = (V_\rho, E_\rho)$ and principal-matching strategy σ
Ensure: Returns set of matched principals $[\![\rho]\!]$

```
1: [[ρ]] ← ∅
2: M_q ← (V, R̃, E, s, {o})
3: while Perform breadth-first search of G_ρ starting at root vertex do
4:     Evaluate current vertex, (φ, ψ, p) ∈ V_ρ
5:     if (φ = all) or (L(M_φ) ∩ L(M_q) ≠ ∅) then
6:         if (ψ = none) or (L(M_ψ) ∩ L(M_q) = ∅) then
7:             if p ≠ null then
8:                 [[ρ]] ← [[ρ]] ∪ p
9:                 if σ = FirstMatch then
10:                    return [[ρ]]
11:                end if
12:            end if
13:        else
14:            Prune all (φ', ψ', p') from evaluation, where (φ, ψ, p) > (φ', ψ', p')
15:        end if
16:    else
17:        Prune all (φ', ψ', p') from evaluation, where (φ, ψ, p) > (φ', ψ', p')
18:    end if
19: end while
20: return [[ρ]]
```

It is possible to represent a *list-oriented* policy using the policy graph (as illustrated in Figure 5.1) by limiting the policy graph to a rooted ordered tree.[4] If I also employ the FirstMatch principal-matching strategy then not only will this policy be ordered, but the first matched principal will be all that is returned.[5]

However, this graph-based approach also makes it possible to encode *principal activation rules* of the form "if p is applicable to a given request then so is principal p'".[6] Moreover, I can insist that a principal p_{n+1} is only activated if multiple path conditions $\pi_1, \ldots, \pi_n$ are satisfied (equivalent to $\pi_1 \wedge \cdots \wedge \pi_n$). This is illustrated in

[3] Equivalent to XACML's "First-applicable" combining algorithm [141].
[4] For the avoidance of doubt, the breadth-first search of policy graphs illustrated in this thesis is performed from left to right.
[5] This approach enables RPPM to implement the UNIX access control model, as will be described in Section 5.2.2.
[6] Assuming the AllMatch principal-matching strategy is employed.

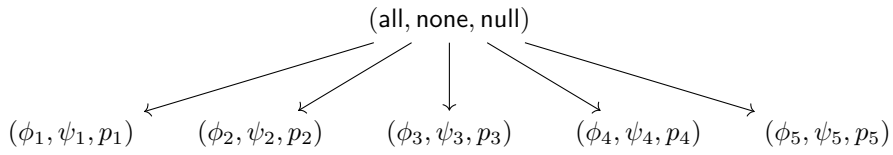


Figure 5.1: A list-oriented tree PMP

Figure 5.2, where path conditions ϕ_1 through ϕ_3 must be satisfied for principal p_4 to be matched to the request.

I believe that such arrangements are useful for combining specialisms with more general principals; for example, a Quality Engineer principal could be activated if both the Engineer and Quality Team Member principals are matched. Nevertheless, principal activation will be particularly relevant to inter-operation and the evaluation of requests in Inter-RPPM (see Section 9.2.2).

Note that principal-matching rules of the form $(\mathsf{all}, \mathsf{none}, p)$ are always applicable (as all is always satisfied and none is never satisfied) and, when evaluated, always result in principal $p \neq \mathsf{null}$ being added to the set of matched principals. Rules of this form are used to "activate" a principal p once some set of preceding principals (from parent rules) are matched.

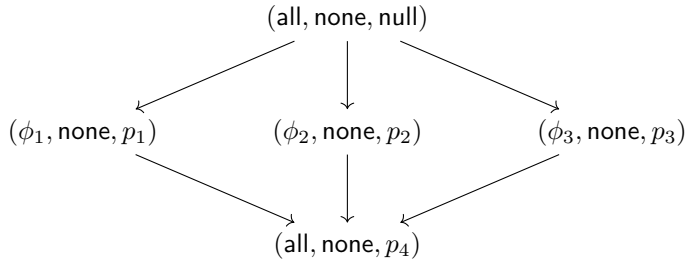


Figure 5.2: Policy conjunction

Example 5.1. *Let us consider the evaluation of the simple policy graph in Figure 5.3 using the AllMatch principal-matching strategy. When evaluating this policy graph there are four potential outcomes. The set of matched principals may be one of* $\emptyset$, $\{p_1\}$, $\{p_2, p_4\}$, *or* $\{p_1, p_2, p_3, p_4\}$.[7] *In particular,* p_4 *is activated if* p_2 *is, because the path conditions associated with* p_2 *are satisfied (and the targets associated with* p_4 *are trivially satisfied); and if both* p_1 *and* p_2 *are activated (because their respective path conditions are satisfied) then so is* p_3 *(as well as* p_4*).*

[7] Recall that the null is not added to the set of matched principals.

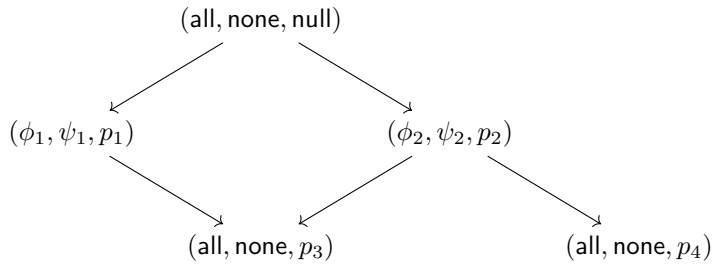


Figure 5.3: A graph-based PMP

5.1.2 Target-Based Request Evaluation

When Evaluating Principal-Matching Policy Sets

The RPPM$_0$ request evaluation process, described in Section 4.2 and Section 4.3, evaluates every principal-matching rule to identify those principals applicable to a request. It subsequently determines whether those principals are authorized to perform the requested action on the object. This process is rather inefficient, as one or more of the principals matched in this way may not appear in any authorization rules associated with the requested object. A natural optimisation is to employ "target-based" evaluation of requests, as exemplified by XACML [141] and other target-based languages [59].

Simply, rather than evaluating every (ϕ, ψ, p) in the principal-matching policy ρ to determine the matched principals for a request $q = (s, o, a)$, I only need to evaluate those rules in the subset

$$\{(\phi, \psi, p) \in \rho : (p, X, Y, b) \in \varrho, o \in X \text{ or } \tau(o) \in X, a \in Y\}.$$

By Definition 4.10, this is equivalent to

$$\{(\phi, \psi, p) \in \rho : (p, X, Y, b) \in \varrho \text{ is applicable to } q\}.$$

In the worst case, of course, I still have to evaluate the entire set of principal-matching rules and each authorization rule will now be checked twice for applicability (once before compute principals, and once during compute authorizations). However, the additional complexity from this repeated check is negligible (asymptotically), particularly when compared to the potential benefits for models where authorization rules don't regularly cover all objects and where multiple principals are employed.

When Evaluating Principal-Matching Policy Graphs

In contrast, request evaluation using the policy-graph variant of the ComputePrincipals algorithm (as described in Section 5.1.1) already includes several mechanisms by which a subset of the principal-matching rules may be evaluated. Firstly, Algorithm 5.1 prunes descendent rules where the ancestor rule is not matched; secondly, the FirstMatch principal-matching strategy, when employed, short circuits evaluation once the first principal is matched.

Whilst these mechanisms exist, a target-based approach similar to that just described may be employed. However, this approach must take into consideration principal-activation. Therefore, rather than simply evaluating principal-matching rules for principals which are part of applicable authorization rules, I must also evaluate all principal-matching rules which are ancestors of such rules. I, therefore, start out evaluating only the rules in the set

$$\{(\phi, \psi, p) \in \rho : \ (p, X, Y, b) \in \varrho \text{ is applicable to } q \text{ or} \\ (p', X', Y', b') \in \varrho \text{ is applicable to } q, (\phi, \psi, p) > (\phi', \psi', p')\}$$

and may evaluate fewer rules due to pruning and the choice of principal-matching strategy.

Once again, in the worst case I still have to evaluate the entire policy graph and each authorization rule will be checked twice for applicability. As before, the additional complexity from this repeated check is asymptotically negligible.

5.1.3 Caching Edges

Principal-Matching Optimisation

Whilst the base RPPM model can cover a wide range of systems, the compute principals step of request evaluation may be computationally intensive for very large system graphs or systems employing principal-matching policies which contain a very large number of rules. The target-based optimisation for request evaluation, described in Section 5.1.2, offers a potential reduction in the processing required during the compute principals step. However, the benefit is limited to simply excluding some of the principal-matching rules during the evaluation of individual requests. Which rules, if any, are excluded varies with each request, but the remainder must be processed as before.

However, note that the set of matched principals for a subject-object pair remains static until a change is made to the system graph or certain associated policy

components. This observation suggests a more widely applicable optimisation. Accordingly, I introduce the concept of *caching edges* and make use of the relative stability of matched principals in order to reduce the processing required for future authorization requests. In order to implement caching edges, a form of *typed edge*, I allow edges within the system graph to be labelled: from the base model's set of relationship labels $\widetilde{R}$; and with a subset of principals from P.

In particular, when I evaluate a request (s, o, a) that results in a set of matched principals $[\![\rho]\!] \subseteq P$, I add an edge $(s, o, [\![\rho]\!])$ to the system graph, directed from s to o and labelled with $[\![\rho]\!]$; this edge identifies the matching principals relevant to future requests of the form (s, o, a'). The subsequent processing of such requests can, therefore, skip the computationally expensive step of computing the matched principals and instead use $[\![\rho]\!]$ (the label of the caching edge) as input to the second (compute authorizations) step of request evaluation. Recall that the action of a request is not part of the compute principals step of request evaluation, and so caching edges are created and used independent of the requested action.

Remark 5.1. *As the target-based request evaluation described in Section 5.1.2 skips principal-matching rules which are not for the requested object* and action*, not all potential matched principals are determined. This* strong *form of target-based request evaluation is, therefore, incompatible with caching edges. I therefore* weaken *the notion of target-based evaluation when used with caching edges to evaluate principal-matching rules for principals which are part of authorization rules appropriate for the target object, no matter the action. In the case of evaluating request $q = (s, o, a)$ against a set-based PMP, the evaluated principal-matching rules are*

$$\{(\phi, \psi, p) \in \rho : (p, X, Y, b) \in \varrho, o \in X \text{ or } \tau(o) \in X\}.$$

In the case of evaluating request $q = (s, o, a)$ against a graph-based PMP, the evaluated rules are

$$\{(\phi, \psi, p) \in \rho : \begin{array}{l} (p, X, Y, b) \in \varrho, o \in X \text{ or } \tau(o) \in X, \text{ or} \\ \qquad (p', X', Y', b') \in \varrho, o \in X \text{ or } \tau(o) \in X, (\phi, \psi, p) > (\phi', \psi', p')\}. \end{array}$$

Example 5.2. *Recall from Example 4.3, that student 1 is the teaching assistant for course 2 and can (based on the rules repeated below), therefore, match the principal*

course-ta and be authorized to read and grade answer 3.

$$\rho = \{\ldots, (\textsf{Ta-for}\,;\,\overline{\textsf{Coursework-for}},\ \textsf{Enrolled-on}\,;\,\overline{\textsf{Coursework-for}},\ \textsf{course-ta}), \ldots\}$$
$$\varrho = \{\ldots, (\textsf{course-ta}, \star, \{\textsf{read}, \textsf{grade}\}, 1), \ldots\}$$

Suppose that student 1 makes the request (student 1, answer 3, read*). Then this request will be authorized because* $[\![\rho]\!] = \{\textsf{course-ta}\}$*. At this stage, I may therefore add an edge (student 1, answer 3,* $\{\textsf{course-ta}\}$*), thereby caching the outcome of the principal-matching phase of request evaluation, as illustrated in Figure 5.4. (I use the convention that caching edges have a diamond-shaped arrow head.) Then a subsequent request (student 1, answer 3,* grade*) will only need to determine that* course-ta *is associated with the subject-object pair (student 1, answer 3) and can immediately evaluate the authorization rules (and thereby authorize the request).*

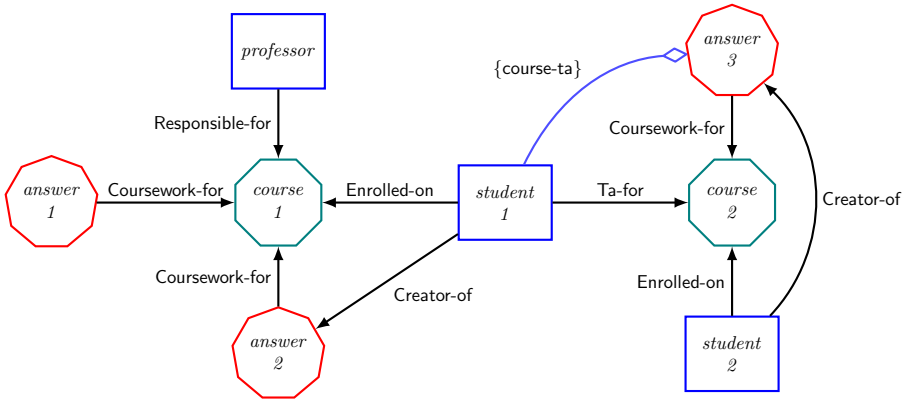


Figure 5.4: Adding the caching edge (*student 1, answer 3,* $\{\textsf{course-ta}\}$)

Whilst the benefit of caching may seem limited in such a trivial example, recall that (as per Section 4.5.2) the overall complexity of request evaluation comes solely from the compute principals step, and is

$$O(|\rho| \cdot \vartheta(\rho) \cdot |V|^2 \cdot |\widetilde{R}|), \text{ where } \vartheta(\rho) = \max\{\ell(\pi) + \vartheta(\pi) : \pi \in \rho\}.$$

For any system where repeat requests are made by a subject on a particular object, the compute principals step may be bypassed after it has been performed in full once. I propose that such repeat requests are common in many computing systems. In my experience users commonly request access to resources multiple times as they perform multiple operations on those resources (e.g., read, write, and print). Equally, users repeat access requests on resources over time as they continue to rely on or modify the content within them. For example, access requests to execute software applications, and the multitude of associated resource requests that go with

these, occur daily with users having applications which they prefer for e-mail, web browsing and writing documents. I, therefore, believe that caching can provide a significant improvement in many situations.

Cache Management

Whilst the performance improvement offered by caching edges may be significant, there is a need to carefully manage any implementation of caching edges to prevent this improvement being countered by an indiscriminate increase in the number of edges in the system graph. In the worst case, the number of caching edges directed out of a node is $O(|V|)$, where V is the set of nodes in the system graph. However, there are strategies that can be used to both prevent the system graph realizing the worst case and to reduce the impact of large numbers of caching edges. To maintain an acceptable number of caching edges, I could, for example, use some form of cache purging. I could also distinguish between relationship edges and caching edges using some flag on the edges, and index the caching edges to dramatically decrease the time taken to search the set of caching edges. Employing these techniques should enable the benefits of caching edges to be realised without incurring unacceptable costs during identification of the relevant caching edge.

More fundamentally, a change to any of the following components of the model could alter the list of matched principals which would be determined for a particular pairing of subject and object. Such changes, therefore, may affect the correctness of caching edges. (The obvious exception is a change to the system graph resulting from the addition or deletion of a caching edge.)

- The system graph;
- The principal-matching policy; and
- The principal-matching strategy (if a graph-based PMP is employed).

The most crude management technique for handling such changes involves removing all caching edges from the system graph whenever one of the above changes occurs. However, in certain specific scenarios it may be possible for a system to identify a scope of impact for a particular change and thus apply a more refined management technique. For example, if a change to the set-based principal-matching policy removes all rules which are used to match a certain principal (and nothing more), then it would be sufficient for only caching edges labelled with a set including that principal to be purged. Whilst such a refinement may further optimise the operations performed by the authorization system, its applicability will depend upon the

configuration of the authorization system in its entirety.

I have already noted that it may make sense to purge the cache in order to limit the number of caching edges in the system graph. Again, there are several possible purging strategies. I present three cache purging strategies which may be employed within RPPM:

- MaxThreshold – the number of caching edges within the system graph is limited to a maximum threshold as either a specific value or a percentage of the total number of edges.
- MaxThresholdOut – the number of caching edges out of each entity is limited to a maximum threshold value.
- Timeout – the duration for which caching edges persist is limited by a timeout value, with edges "retired" once their timeout has expired.

Numerous other strategies could be defined, as could mixed strategies, and these may be desirable for specific applications.

Pre-Emptive Caching

Any optimisation provided by the caching of matched principals relies upon the existence of a caching edge in order to reduce the authorization request processing. The first request between a subject and object must, therefore, be processed normally in order to determine the set of matched principals which will label the caching edge. If this initial evaluation were only performed when an authorization request were submitted then the benefit of caching edges would be limited to repeated subject-object interactions alone.

However, many authorization systems will experience periods of time when no authorization requests are being evaluated. The nature of many computing tasks is such that authorization is required sporadically amongst longer periods of computation by clients of the authorization system and idle time for the authorization system itself. These periods of reduced load on the authorization system can be employed for the purpose of *pre-emptive caching*.

Thus for pairs of nodes (u, v) in the system graph, I may compute $[\![\rho]\!]_{u,v}$ and insert a caching edge $(u, v, [\![\rho]\!]_{u,v})$.[8] The fact that a request's action is not employed during the principal matching process means that to perform this further optimisation an authorization system solely requires a subject and object between whom the matched principals are to be identified. There are numerous potential strategies

[8]Here I use the notation $[\![\rho]\!]_{u,v}$ as an equivalent to $[\![\rho]\!]_q$ where $q = (u, v, a)$ for any $a \in A$.

for determining which subject-object pairs should be considered for pre-emptive caching. Here I describe two simple and natural strategies.

Subject-focused. Subject-focused pre-emptive caching assumes that subjects which have recently made authorization requests are *active* and so will likely make further requests. The authorization system, therefore, prioritises determining the list of matched principals between the most recently active subjects and a set of target objects. The set of target objects could be selected at random or may be systematically chosen using an appropriate mechanism for the system defined in the system graph. This might involve the target objects being *popular*, *significant* or those whose access may be particularly *time-sensitive*. I envisage that the interpretation of these concepts may be system specific, as may be their relative worth.

As pre-emptive caching builds the number of caching edges within the system graph the number of subjects and objects under consideration could be expanded to provide greater coverage of the potential future requests (allowing for whatever cache purging strategy is employed).

Object-focused. In certain applications, there will be resources that will be used by most users, such as certain database tables. Thus, it may make sense to construct caching edges for all active users for certain resources.

No matter the strategy, pre-emptive caching makes use of available processing time in order to perform the most complex part of authorization request evaluation: principal matching. Any requests that are made utilising a subject-object pair which has already been evaluated by pre-emptive caching will be able to make use of the caching edge already established, even if that request were the first received for that pair. Once determined, caching edges resulting from pre-emptive caching are no different from those established as a result of request evaluation.

5.2 Model Comparisons

5.2.1 Request Evaluation Enhancements

The $RPPM_{1a}$ model introduces three request evaluation enhancements to the $RPPM_0$ model, increasing the expressiveness of the policy language and optimising the request evaluation process. The enhancements may be employed individually or in combination, as is appropriate for the system being modelled and the policies to be enforced.

Policy Graph Evaluation By organising the principal-matching policy within a graph, RPPM_{1a} can express more complex policies, including ordered lists, conjunction and principal activation. Principal-matching rules are vertexes within the policy graph, and the edges indicate the child rules which should be evaluated if their parent rules are matched. This arrangement allows a policy to contain dependencies and enables rules to be pruned from the evaluation if their parent rules don't match, thus reducing the number of rules that need to be evaluated. Whatever the arrangement of the policy graph, the process of evaluating individual principal-matching rules is no different to that used by RPPM_0. Whilst "policy graphs" have been used in other access control models, these uses are not directly equivalent to mine and the term has been variously used to define: separation of duty and temporal constraints between roles [35]; constraints between the types of entity [102] (much like my path condition); and, more close to my use, chains of first-order logic policies protecting a resource [167].

Target-Based Request Evaluation Strong target-based request evaluation in RPPM_{1a} only evaluates principal-matching rules which can affect the result of the current request evaluation. These are determined by identifying the principals within authorization rules defined for the target object and requested action. In contrast, weak target-based request evaluation, which can be used alongside caching edges, evaluates principal-matching rules which can affect the result of any request between the subject and object (ignoring the action). Whether strong or weak, target-based request evaluation can reduce the number of principal-matching rules which must be evaluated by RPPM_{1a} when compared to RPPM_0.[9] Whilst the mechanisms may be different, other models, such as XACML and PTaCL, employ targets to limit the requests to which the model's policy rules apply [59, 141].

Caching Edges Caching edges have direct value in the RPPM_{1a} model, allowing the computationally expensive part of the request evaluation process to be bypassed. This allows for intuitive and broadly applicable optimisation strategies to be employed. There has been some interest in recent years in reusing, recycling or caching authorization decisions at policy enforcement points in order to avoid re-computing decisions [37, 117, 118, 187]. These techniques have perceived benefit, in particular, in large-scale, distributed, systems due to demands for reduced latency and resilience to intermittent communications failures. Whilst caching in the RPPM model does not resolve connectivity issues directly, the capability has considerable

[9] This optimisation benefits any system where authorization rules don't regularly cover all objects and where multiple principals are employed.

impact on latency. Moreover, a cached edge applies to multiple requests, irrespective of whether the exact request has previously been evaluated, unlike many, if not all, proposals in the literature. Whilst many authorization recycling strategies involve the policy enforcement point maintaining its own cache, or employing a "speculative" system [115], caching in the RPPM$_{1a}$ model updates the system graph itself and thus is implemented by very simple enhancements to the base model.

5.2.2 Implementing Non-ReBAC Models

UNIX

As described in Section 4.6.2, RPPM$_0$ was unable to implement the UNIX access control model as it cannot replicate UNIX's ordered request evaluation process. The introduction of policy graph evaluation in RPPM$_{1a}$, however, resolves this limitation. To implement the UNIX access control model, RPPM$_{1a}$ is configured such that the entity types are users, groups and objects. Three relationship types are defined in the system model: the *uo* relationship is used between users and objects to identify UNIX's "owner" principal; the *ug* relationship identifies user membership of groups; and the *go* relationship identifies group assignment to objects. Together, these last two relationships are used to identify UNIX's "group" principal, whilst a default principal-matching rule is used to identify "other".

$$
\begin{aligned}
T &= \{users, groups, objects\} \\
R &= \{uo, ug, go\} \\
S &= \emptyset \\
E_{\mathrm{PR}} &= \{(users, objects, uo), (users, groups, ug), (groups, objects, go)\} \\
G_{\mathrm{PR}} &= (T, E_{\mathrm{PR}})
\end{aligned}
$$

In order to have RPPM evaluate the three principal-matching rules in order, and to ensure only one of them is matched, I employ the graph-based PMP shown in Figure 5.5 and the FirstMatch principal-matching strategy. The combination of these two elements means that only the first principal from owner, group or p_{def} (conceptually equivalent to UNIX's "other") will be matched to the request. As p_{def} is matched trivially, due to the use of the principal-matching rule $(\mathsf{all}, \mathsf{none}, p_{\mathrm{def}})$, one of the three principals will always be matched.

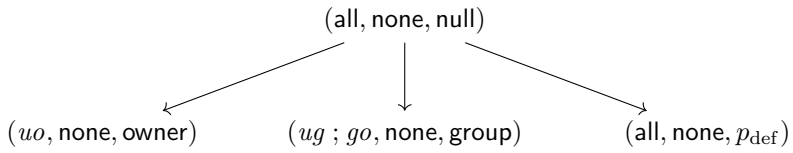


Figure 5.5: The UNIX tree policy

5.3 Summary

I have introduced three request evaluation enhancements to the base RPPM model to produce RPPM$_{1a}$. I have shown how these enhancements increase the expressiveness of the policy language, and optimise the processing of request evaluation by reducing or bypassing the processing of the most complex step (compute principals). I have also demonstrated how the policy graph enhancement enables RPPM to implement the UNIX access control model. Whilst these enhancements have made RPPM more useful and efficient, further enhancements (introduced in the next chapter) are also desirable to ensure that RPPM may be effective in a range of enterprise application environments where prior decisions are important to the current evaluation.

Chapter 6

$\mathbf{RPPM_{1b}}$

In this chapter I present $\text{RPPM}_{1\text{b}}$, a development of the RPPM_0 model which includes *audit edges*. In addition to their use for the logging of authorization decision outcomes, I use audit edges to implement the following *policy configuration enhancements*:

- History-based policies – the introduction of *decision audit edges* enables RPPM to include ad hoc principal-matching rules which inform authorization decisions based on the results of previous request evaluations.
- Separation of duty – the systematic use of decision audit edges enables policies to be defined whereby a subject is only permitted to perform one of n constrained actions on an object. I further show that RPPM can support more stringent constraints such as requiring n one-time actions to be distributed amongst n subjects.
- Binding of duty – the systematic use of decision audit edges also enables policies requiring a single subject to perform a set of n constrained actions on an object.
- Chinese Wall – the introduction of *interest audit edges* enables RPPM to track a subject's interest in objects and to, thereby, deny access requests which attempt to access objects in a distinct *conflict of interest class*.

Policy configurations for history-based policies, (standard) separation of duty and Chinese Wall, described in Section 6.1, have previously been published, in conjunction with Jason Crampton, in [60, 61, 63].

6.1 Policy Configuration Enhancements

6.1.1 History-Based Policies

The RPPM_0 and RPPM_{1a} models (introduced in Chapters 4 and 5) use paths of relationships through logical and concrete entities to inform authorization decisions. Whilst there is nothing to prevent such entities within a system from being time-related,[1] request evaluation in these models is based on context derived purely from the moment in time at which the evaluation is performed. Even where caching is employed, past evaluations do not impact the result of the current evaluation, only the complexity of determining that result. These forms of the RPPM model are, therefore, "memory-less" with respect to request evaluations. I introduce audit edges to enable RPPM_{1b} to inform the current context of the system graph based on the evaluation of previous requests.

As with caching edges (introduced in Section 5.1.3), audit edges are a form of typed edge which I introduce into the system graph. Specifically, I support two kinds of audit edge:

- *Decision audit edges* – which record the decisions from previous authorization requests and are labelled with an indication of whether a requested action a was authorized $a^{\oplus}$ or denied $a^{\ominus}$ to a subject on an object.
- *Interest audit edges* – which record a subject's active or blocked interest, $i^{\oplus}$ or $i^{\ominus}$ respectively, in an entity. (I shall return to interest audit edges in Section 6.1.4.)

At its basis, the introduction of decision audit edges allows the system to record whether previously requested actions were authorized or denied. Both authorized and denied decision audit edges are inserted, automatically, into the system graph after request evaluation completes. Specifically, if such an edge does not already exist, a decision audit edge is added between the subject and object of the evaluated request, indicating its result.[2]

Whilst RPPM_{1a}'s caching edges arise from the principal-matching part of the request evaluation process, an audit edge arises from the second phase of the evaluation process, compute authorizations. I extend the set of relationship labels by defining the relationships $a^{\oplus}$ and $a^{\ominus}$ for each action a, and upon completion of request evaluation:

[1] A system might employ logical entities representing groups, including an *ex-employee* group or a *2017-cohort* group.

[2] In some situations there may be merit in recording every occurrence of an action (by a subject on an object) being authorized or denied. Modifying the decision audit edge's label to include a count would be a simple way of achieving this if it were required.

- I add the edge $(s, o, a^{\oplus})$ to the system graph if the decision for (s, o, a) is allow; and
- I add the edge $(s, o, a^{\ominus})$ to the system graph if the decision for (s, o, a) is deny.

These edges can be utilised in a variety of ways depending on the requirements of the authorization system. At their simplest, they provide a record which may be used as input for auditing or other processing outside of the authorization system.[3] In contrast, decision audit edges may also be used to satisfy part of a path condition in the system graph, thus enabling authorization decisions to be made based on historical evidence. Reputation and history-based access control (HBAC) systems rely on just such knowledge of previous actions to inform decisions [1, 72, 121].

Principal-matching rules may, therefore, be created to make direct use of decision audit edges. Some obvious examples include:

- The principal-matching rule $(\mathsf{all}, a^{\oplus}, p)$ can be used to match the principal p to any request where the subject has not previously been granted the action a on the object;
- The rule $(\mathsf{all}, a^{\ominus}, p)$ requires that the subject has never been denied action a on the object; and
- The rule $(a^{\oplus}, \mathsf{none}, p)$ requires that the subject must have previously had a request to perform action a on the object approved.

Example 6.1. *Returning to my higher education example (introduced in Example 4.3), suppose that I have student 2 who is enrolled on course 2 and is the author of coursework answer 3. Then student 1, the teaching assistant for the course will, at some point, grade the coursework answer 3. At this point, student 2 should not be able to modify answer 3. I could enforce this requirement by modifying the principal-matching rule* (*Creator-of*, none, *author*)*—which assigns any user who is the creator of a piece of coursework to the* **author** *principal—to*

$$(\textit{Creator-of},\ \textit{Enrolled-on}\,;\,\overline{\textit{Ta-for}}\,;\,\textit{grade}^{\oplus},\ \textit{author}).$$

This rule includes a forbidden target $\textit{Enrolled-on}\,;\,\overline{\textit{Ta-for}}\,;\,\textit{grade}^{\oplus}$ *that must not be matched if the principal* **author** *is to apply to a request. This path condition traces a path from the enrolled student to the course to the teaching assistant to the (graded) coursework. Figure 6.1 illustrates the system graph once the teaching assistant has*

[3] In this form they may, for example, be used to identify potentially malicious actors. A node with a large number of, or a sudden increase in, denied decision audit edges may be considered worthy of further investigation.

graded answer 3; I represent allow audit edges using dashed lines. Note that there is a path from student 2 to answer 3 matching the prohibited path condition.

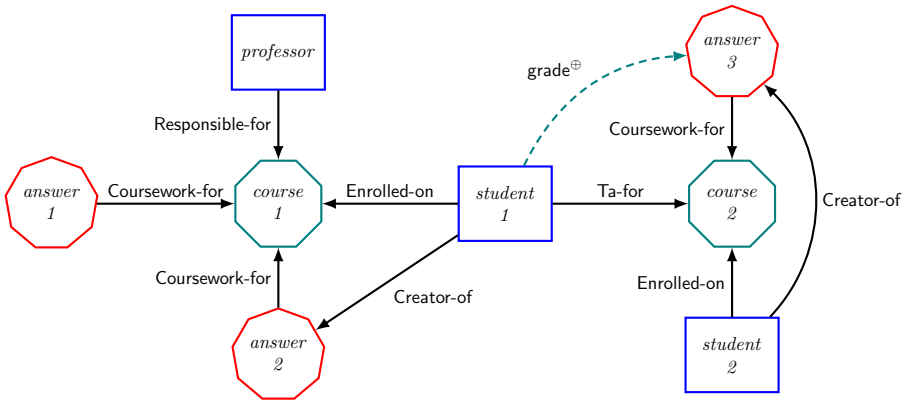


Figure 6.1: Adding the decision audit edge (*student 1*, *answer 3*, grade$^{\oplus}$)

Of course, in practice student 2 will still wish to read answer 3, so I might wish to specify a separate rule, rather than modify the existing rule. This separate rule could have the form

$$(\mathit{Enrolled\text{-}on}\,;\,\overline{\mathit{Ta\text{-}for}}\,;\,\mathit{grade}^{\oplus},\mathsf{none},\mathit{graded\text{-}student}),$$

and then I specify an additional authorization rule

$$(\mathit{graded\text{-}student},\star,\{\mathit{write}\},0)$$

which explicitly denies write *access to any object by the principal* graded-student*.*

6.1.2 Separation of Duty

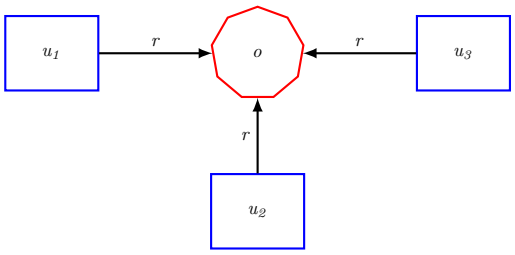
Whilst decision audit edges can be used in an ad hoc manner to enforce application-specific constraints, I can also use them to enforce separation of duty in a systematic way. Separation of duty requires that certain combinations of actions are performed by a number of distinct individuals so as to reduce the likelihood of abuse of a system. In its simplest form, separation of duty constraints require two individuals to each perform one of a pair of distinct actions so that a single individual cannot abuse the system. A common application environment for such constraints is that of a finance system, where, for example, the individual authorized to add new suppliers should not be the same individual who is authorized to approve the payment of invoices to suppliers. If a single individual were able to perform both of these actions they could set themselves up as a supplier within the finance system and then approve for payment any invoices they submitted as that supplier.

One of n Actions

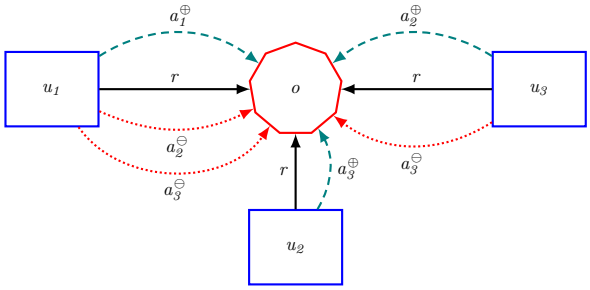
I define a mechanism here through which subjects are limited to performing one of n specific actions associated with an object.[4] Let us consider the system graph fragment shown in Figure 6.2a, a set of n actions of interest $\{a_1, \ldots, a_n\} \supseteq A$, and the policies

$$\rho = \{(r, \mathsf{none}, p)\}$$
$$(\varrho, \chi) = (\{(p, \{o\}, \star, 1)\}, \mathsf{DenyOverrides}).$$

With these policies (whether I use audit edges or not), the request $q_1 = (u_1, o, a_1)$ made by u_1 will be authorized by matching principal p, as will subsequent requests $q_2 = (u_1, o, a_2)$ and $q_3 = (u_1, o, a_3)$. A similar result would have occurred if these requests had been submitted with u_2 or u_3 as the subject.



(a) System graph fragment



(b) After request $q_6' = (u_2, o, a_3)$

Figure 6.2: Enforcing separation of duty

Now suppose I wish to restrict each user to a single interaction with o. Given n actions of interest $\{a_1, \ldots, a_n\}$ and index i, $1 \leqslant i \leqslant n$, I shall use the notation $A^n_{\not a_i} = \{a_1, \ldots, a_n\} \setminus \{a_i\}$ to indicate the set of $n - 1$ actions of interest other than

[4] Note that this doesn't force users to specific (or even distinct) actions. However, I will define a more restrictive mechanism later in Section 6.1.2.

action a_i. Then I define the policies

$$\rho' = \rho \cup \{(a_i^{\oplus}, \mathsf{none}, p_i) : 1 \leqslant i \leqslant n\}$$
$$\varrho' = \varrho \cup \{(p_i, \{o\}, A^n_{\not a_i}, 0) : 1 \leqslant i \leqslant n\}.$$

Then, starting anew with no audit edges in the system graph (as per Figure 6.2a), request $q'_1 = (u_1, o, a_1)$ matches the rule in ρ, as before, and the request is authorized by the rule in (ϱ,χ). However, assuming the audit edge $(u_1, o, a_1^{\oplus})$ is now added to the system graph, a subsequent request $q'_2 = (u_1, o, a_2)$ will match the new principal-matching rule $(a_1^{\oplus}, \mathsf{none}, p_1)$, leading to a deny decision (because of the new, blocking, authorization rule $(p_1, \{o\}, \{a_2, a_3\}, 0)$). At this point a deny audit edge $(u_1, o, a_2^{\ominus})$ will be added to the system graph. Similarly, request $q'_3 = (u_1, o, a_3)$ will be denied by the same rules. Attempts by u_2 or u_3 to perform multiple different actions will, equally, result in at most one allow decision. For example, u_3 may be granted $q'_4 = (u_3, o, a_2)$ but would subsequently be denied $q'_5 = (u_3, o, a_3)$; u_2 may be granted $q'_6 = (u_2, o, a_3)$. This case of $n = 3$ is illustrated in Figure 6.2b, where each of the three users has been permitted to perform one of the three actions. More formally, I have the following result.

Proposition 6.1. *Given an RPPM separation of duty policy, as described above, for any user u the request (u, o, a) is allowed if the request is authorized by ρ' and (ϱ', χ) and no request of the form (u, o, a') has been previously authorized where $a' \neq a$ and $a, a' \in \{a_1, \ldots, a_n\}$. The request is denied otherwise.*

Proof. Consider $E^{\oplus} = \{(s_1, o_1, a_1^{\oplus}), \ldots, (s_m, o_m, a_m^{\oplus})\}$ the set of authorized decision audit edges where $E^{\oplus} \subset E$, resulting from a sequence of m authorized requests $\{(s_1, o_1, a_1), \ldots, (s_m, o_m, a_m)\}$, and a new request (u, o, a) where $a \in \{a_1, \ldots, a_n\}$. Then:

- If there exists $(u, o, a'^{\oplus}) \in E^{\oplus}$ for some $a' \neq a$, $a' \in \{a_1, \ldots, a_n\}$ then (u, o, a) will be denied, using the DenyOverrides CRS, by matching the principal-matching rule $(a'^{\oplus}, \mathsf{none}, p_{a'})$ in $\rho' \setminus \rho$ and the authorization rule $(p_{a'}, \{o\}, A^n_{\not a'}, 0)$ in $\varrho' \setminus \varrho$ being applicable to the request.
- Otherwise, (u, o, a) is authorized by matching principal p in the principal-matching rule $(r, \mathsf{none}, p) \in \rho$ and then by authorization rule $(p, \{o\}, \star, 1) \in \varrho$ being applicable to the request.

□

Whilst I have introduced this mechanism for separation of duty employing a set-based PMP and the DenyOverrides conflict resolution strategy (as provided by

RPPM_0), it operates equally well with a graph-based list-oriented policy (as introduced in $\text{RPPM}_{1\text{a}}$) assuming the added constraint rules are inserted at the start of the PMP's list.

n One-time Actions Distributed Amongst n Subjects

The separation of duty mechanism just described prevents each subject from performing more than one action on an object. This constraint is the most commonly considered duty constraint; however, within workflow systems a wider range of constraints can be found [31, 39, 56, 114]. Whilst I leave the wider topic of workflow support within RPPM for future work, I illustrate here that RPPM is already able to support more powerful constraints than standard separation of duty. For example, $\text{RPPM}_{1\text{b}}$ can support a stronger constraint whereby n one-time actions (which may be thought of as tasks in respect of workflows) are distributed amongst n subjects.

Let us, once again, consider the system graph fragment shown in Figure 6.2a, a set of n actions of interest $\{a_1, \ldots, a_n\} \supseteq A$, and the initial policies

$$\begin{aligned} \rho &= \{(r, \mathsf{none}, p)\} \\ (\varrho, \chi) &= (\{(p, \{o\}, \star, 1)\}, \mathsf{DenyOverrides}). \end{aligned}$$

Now suppose I wish to distribute n actions amongst n users, such that each user is only able to perform a single, distinct, un-repeated task on o.[5] Once again, given n actions of interest $\{a_1, \ldots, a_n\}$ and index i, $1 \leqslant i \leqslant n$, I shall use the notation $A^n_{\not a_i} = \{a_1, \ldots, a_n\} \setminus \{a_i\}$ to indicate the set of $n-1$ actions of interest other than action a_i. Then based on policies defined for separation of duty

$$\begin{aligned} \rho' &= \rho \cup \{(a_i^{\oplus}, \mathsf{none}, p_i) : 1 \leqslant i \leqslant n\} \\ \varrho' &= \varrho \cup \{(p_i, \{o\}, A^n_{\not a_i}, 0) : 1 \leqslant i \leqslant n\}, \end{aligned}$$

I define the policies

$$\begin{aligned} \rho'' &= \rho' \cup \{(r\,;\overline{r}\,;a_i^{\oplus}, \mathsf{none}, p_{r\overline{r}i}) : 1 \leqslant i \leqslant n\} \\ \varrho'' &= \varrho' \cup \{(p_{r\overline{r}i}, \{o\}, \{a_i\}, 0) : 1 \leqslant i \leqslant n\}. \end{aligned}$$

Note that the target $r\,;\overline{r}\,;a_i^{\oplus}$ (used in the new principal-matching rules of ρ'') is satisfied if any of the users have been previously authorized to perform the action a_i, no matter which user is the subject.

[5] Such an arrangement may, for example, be relevant in the commonly cited finance system use case.

Proposition 6.2. *Given an RPPM distributed actions policy, as described above, for any user u the request (u, o, a) is allowed if the request is authorized by ρ'' and (ϱ'', χ) and no request of the form (u, o, a') has been previously authorized, where $a' \in \{a_1, \ldots, a_n\}$, and where no request of the form (u', o, a) has been previously authorized, where $u \neq u'$. The request is denied otherwise.*

Proof. Consider $E^{\oplus} = \{(s_1, o_1, a_1^{\oplus}), \ldots, (s_m, o_m, a_m^{\oplus})\}$ the set of authorized decision audit edges where $E^{\oplus} \subset E$, resulting from a sequence of m authorized requests $\{(s_1, o_1, a_1), \ldots, (s_m, o_m, a_m)\}$, and a new request (u, o, a) where $a \in \{a_1, \ldots, a_n\}$. Then:

- If there exists $(u, o, a'^{\oplus}) \in E^{\oplus}$ for some $a' \neq a$, $a' \in \{a_1, \ldots, a_n\}$ then (u, o, a) will be denied, using the DenyOverrides CRS, by matching the principal-matching rule $(a'^{\oplus}, \mathsf{none}, p_{a'})$ in $\rho' \setminus \rho$ and the authorization rule $(p_{a'}, \{o\}, A^n_{a'}, 0)$ in $\varrho' \setminus \varrho$ being applicable to the request.
- If there exists $(u', o, a^{\oplus}) \in E^{\oplus}$ then (u, o, a) will be denied, using the DenyOverrides CRS, by matching the principal-matching rule $(r\,; \overline{r}\,; a^{\oplus}, \mathsf{none}, p_{r\overline{r}i})$ in $\rho'' \setminus \rho'$ and the authorization rule $(p_{r\overline{r}i}, \{o\}, \{a\}, 0)$ in $\varrho'' \setminus \varrho'$ being applicable to the request.
- Otherwise, (u, o, a) is authorized by matching the principal p in the principal-matching rule $(r, \mathsf{none}, p) \in \rho$ and then by authorization rule $(p, \{o\}, \star, 1) \in \varrho$ being applicable to the request.

□

6.1.3 Binding of Duty

Whilst separation of duty constraints prevent a subject performing multiple actions on an object, the purpose of binding of duty constraints is the opposite. A binding of duty constraint on a set of actions specifically requires that they must be performed by a single subject [176, 184]. I define a mechanism here through which a set of n specific actions must be performed by a single subject on an object.

Let us again consider the system graph fragment shown in Figure 6.2a, a set of n actions of interest $\{a_1, \ldots, a_n\} \supseteq A$, and the initial policies

$$\rho = \{(r, \mathsf{none}, p)\}$$
$$(\varrho, \chi) = (\{(p, \{o\}, \star, 1)\}, \mathsf{DenyOverrides}).$$

Now suppose I wish to constrain n actions to a single subject. Then I define policies

$$\rho' = \rho \cup \{(r\,;\overline{r}\,;a_i^{\oplus}, a_i^{\oplus}, p_{excl}) : 1 \leqslant i \leqslant n\}$$
$$\varrho' = \varrho \cup \{(p_{excl}, \{o\}, \{a_1, \ldots, a_n\}, 0)\}.$$

Note that the target combination of $r\,;\overline{r}\,;a_i^{\oplus}$ as a required target and $a_i^{\oplus}$ as a forbidden target means that the principal matching-rule $(r\,;\overline{r}\,;a_i^{\oplus}, a_i^{\oplus}, p_{excl})$ is only matched if a user other than the requesting subject has previously been authorized to perform the action a_i.

Proposition 6.3. *Given an RPPM binding of duty policy, as described above, for any user u the request (u, o, a) is allowed if the request is authorized by ρ' and (ϱ', χ) and no request of the form (u', o, a') has been previously authorized where $u' \neq u$ and $a, a' \in \{a_1, \ldots, a_n\}$. The request is denied otherwise.*

Proof. Consider $E^{\oplus} = \{(s_1, o_1, a_1^{\oplus}), \ldots, (s_m, o_m, a_m^{\oplus})\}$ the set of authorized decision audit edges where $E^{\oplus} \subset E$, resulting from a sequence of m authorized requests $\{(s_1, o_1, a_1), \ldots, (s_m, o_m, a_m)\}$, and a new request (u, o, a) where $a \in \{a_1, \ldots, a_n\}$. Then:

- If there exists $(u', o, a'^{\oplus}) \in E^{\oplus}$ for some $a' \in \{a_1, \ldots, a_n\}$ then (u, o, a) will be denied, using the DenyOverrides CRS, by matching the principal-matching rule $(r\,;\overline{r}\,;a^{\oplus}, a^{\oplus}, p_{excl})$ in $\rho' \setminus \rho$ and the authorization rule $(p_{excl}, \{o\}, \{a_1, \ldots, a_n\}, 0)$ in $\varrho' \setminus \varrho$ being applicable to the request.
- Otherwise, (u, o, a) is authorized by matching the principal p in the principal-matching rule $(r, \mathsf{none}, p) \in \rho$ and then by authorization rule $(p, \{o\}, \star, 1) \in \varrho$ being applicable to the request.

□

6.1.4 Chinese Wall

The Chinese Wall principle may be used to control access to information in order to prevent any conflicts of interest arising. The standard use case concerns a consultancy that provides services to multiple clients, some of which are competitors of each other. It is important that a consultant does not access documents of company c if she has previously accessed documents of a competitor of c. To support the Chinese Wall policy, data is categorised using conflict of interest classes to indicate groups of competitor entities [40]. Requests to access a company's resources within a

conflict of interest class will only be authorized if no previous request was authorized accessing resources from another company in that conflict of interest class.

It should immediately be clear that audit edges have a part to play in enforcing Chinese Wall policy, with prior actions affecting the result of the current request evaluation. Whilst decision audit edges could be used to achieve the desired conflict of interest protection, the fact that these are added between the subject and object of the decided request limits their usefulness in the kinds of large entity arrangements that Chinese Wall is intended for. Specifically, in a large system graph representing multiple companies and their many resources organised in various arrangements, the Chinese Wall principal-matching rules expected to traverse the audit edges would become complex and numerous. This is because there would be an interest in the existence of paths from a user through a resource belonging to one organisation, through a competing organisation, and through that competing organisation's resources and back to the user. In order to conceptually simplify the policy arrangement for Chinese Wall, I introduce interest audit edges as a second type of audit edge.

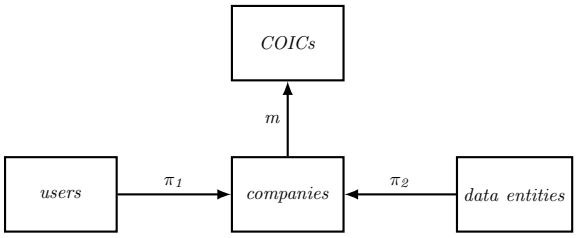
In a similar way to decision audit edges, interest audit edges record information related to previous requests which can be utilised to make future decisions. However, whilst the decision audit edges record the direct result of a previous request evaluation, interest audit edges record the higher level notion of "interest" associated with those requests. A subject which requests to perform an action on an object can be considered to be showing an interest in that object (or an entity which that object is related to). An authorization system may be configured to use the record of this interest to determine whether a future request on that, or another, object should be approved or denied.

Let us suppose that for a given conflict of interest class c and every user u, I have $G, u, c \models \pi_1$, and for every resource o, I have $G, o, c \models \pi_2$. Then for this conflict of interest class I have $G, u, o \models \pi_1; \overline{\pi_2}$ for every user and resource. This arrangement is depicted, conceptually, in Figure 6.3a.[6] I introduce a logical entity into the system graph to represent a conflict of interest class and the relationship m, where an edge (c, i, m) indicates company c is a member of conflict of interest class i. (I assume here that membership of conflict of interest classes is determined when the system graph is initially populated and remains fixed through the lifetime of the system.)

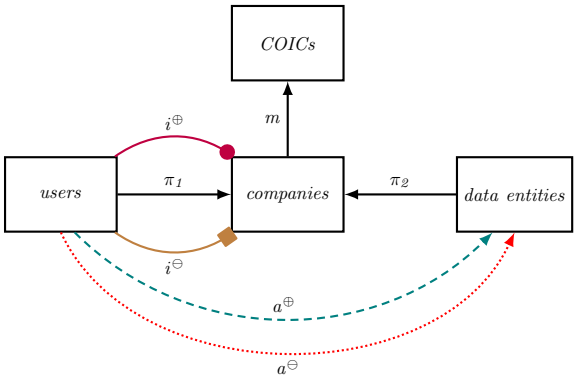
I now introduce interest audit edges into the system graph which are added between users and companies (see Figure 6.3b).[7] Active interest audit edges are

[6] Figure 6.3 does not show a system graph, it shows high-level representations of the "shape" of a system graph.

[7] As the interest audit edge connects to a company, rather than it's resources, this abstracts some of the complexity which would have existed if decision audit edges had been employed.



(a) Basic layout with COICs



(b) With audit edges

Figure 6.3: Chinese Wall generalisation

labelled with $i^{\oplus}$, whilst blocked interest audit edges are labelled with $i^{\ominus}$. I denote edges of the form $(u, c, i^{\oplus})$ with a filled circle head and those of the form $(u, c, i^{\ominus})$ with a filled square head. I, therefore, extend the set of relationships $\widetilde{R}$ to include the set $\{i^{\oplus}, i^{\ominus}\}$, thus allowing the system graph to support these new edges.[8] When users are authorized (or denied) access to particular data entities, authorized (or denied) decision audit edges will result for these requests as shown in Figure 6.3b.

Given a system graph $G = (V, E)$ such that $G, u, c \models \pi_1$ and $G, o, c \models \pi_2$ for all users u, all objects o and all companies c with membership of a given conflict-of-interest class i, I assume an initial policy

$$\rho = \{(\pi_1 \,;\, \overline{\pi_2}, \mathsf{none}, p)\}$$
$$\varrho = \{(p, \star, \{\mathsf{read}\}, 1)\}.$$

This policy ensures that every request of the form (u, o, read) is matched to principal p which is authorized to read every o.

Now suppose that I wish to extend this basic policy and enforce a Chinese Wall policy, which requires that if a user u reads a document belonging to company c

[8] Whilst I do not rely upon decision audit edges to enforce Chinese Wall policies they, equally, do not interfere with interest audit edges. My discussion of Chinese Wall will consider an authorization system which is also supporting decision audit edges so as to provide a more complete picture of an RPPM$_{1b}$ model.

where $(c, i, m) \in E$ then u must not read any document belonging to c', where $c' \neq c$ and $(c', i, m) \in E$. Then I redefine the policy to[9]

$$\rho' = \{(\pi_1 \,;\, \overline{\pi_2}, i^{\ominus} \,;\, \overline{\pi_2}, p)\}$$
$$\varrho' = \varrho.$$

Consider an initial request (u, o, read), where o is a document owned by c a member of conflict-of-interest class i. If principal p is matched, that is

$$G, u, o \models \pi_1 \,;\, \overline{\pi_2} \quad \text{and} \quad G, u, o \not\models i^{\ominus} \,;\, \overline{\pi_2},$$

then the request is authorized (since the principal p is matched) and the following edges are added to G:[10]

- $(u, c, i^{\oplus})$;
- $(u, c', i^{\ominus})$ for all $c' \neq c$ where $(c, i, m) \in E$ and $(c', i, m) \in E$; and
- $(u, o, read^{\oplus})$.

Consider a subsequent request (u, o', read), where o' is owned by $c' \neq c$ and c' belongs to the same conflict-of-interest class as c. Then $G, u, o' \models i^{\ominus} \,;\, \overline{\pi_2}$, principal p is no longer matched and the request will be denied (assuming the default decision function is populated to ensure a denial in such cases).

This is illustrated in Figure 6.4, where $\pi_1 = wf \,;\, cf$ and $\pi_2 = df$ (where wf is short for Works-for, cf is short for Consults-for and df is short for Document-for). A member of staff u_1 works for a consultancy firm e_1 that acts on behalf of clients (c_1, c_2 and c_3) and stores data about the commercial interests of those clients in the form of files (f_1, f_2, f_3 and f_4). The figure illustrates the system graph before (Figure 6.4a) and after (Figure 6.4b) requests $(u_1, f_1, \mathsf{read})$ and $(u_1, f_4, \mathsf{read})$ have been authorized (note both f_1 and f_4 are files of the same company c_1). This results in additional edges in the system graph (shown in Figure 6.4b), notably $(u_1, c_2, i^{\ominus})$, which means that $G, u_1, f_2 \models i^{\ominus} \,;\, \overline{\pi_2}$ and the request $(u_1, f_2, \mathsf{read})$ would be denied (since p would not be matched). Note that request $(u_1, f_3, \mathsf{read})$ would be permitted because $G, u_1, f_3 \not\models i^{\ominus} \,;\, \overline{\pi_2}$. The audit and interest edges added through evaluation of these two further requests are also shown in Figure 6.4c.

My discussion has so far considered the case where a single path of relationships exists between users and companies and between objects and companies; in reality

[9] If my base principal-matching rule employed a prohibited target already then I could still enforce the Chinese Wall policy by inserting an additional principal-matching rule of the form $(i^{\ominus} \,;\, \overline{\pi_2}, \mathsf{none}, p_{block})$ where the principal p_{block} is denied all actions on all objects [60].

[10] Note that interest audit edges are only added when a request is authorized.

there may be multiple alternative paths and the basic layout can be adjusted to enable this.[11] To support such multi-path scenarios, the approach described above can be generalised as follows. Let $\pi_1, \ldots, \pi_n$ where $n \geqslant 1$ be paths between users and companies, and let $\pi'_1, \ldots, \pi'_m$ where $m \geqslant 1$ be paths between objects and companies. The set of principal-matching rules required to authorise users to access objects (through all possible combinations of these paths) is

$$\{(\pi_i \,;\, \overline{\pi'_j},\ \mathsf{none},\ p) : 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}.$$

In order to support Chinese Wall policies, these rules are modified to

$$\{(\pi_i \,;\, \overline{\pi'_j},\ i^{\ominus} \,;\, \overline{\pi'_j},\ p) : 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant m\}.$$

More generally, suppose I have a principal-matching policy ρ and an extended authorization policy (ϱ, χ) where $\chi = \mathsf{DenyOverrides}$. In order to enforce the Chinese Wall constraint using the basic layout shown in Figure 6.3 I define the policies

$$\rho' = \rho \cup \{(i^{\ominus} \,;\, \overline{\pi_2}, \mathsf{none}, p_{cw})\}$$
$$\varrho' = \varrho \cup \{(p_{cw}, \star, \star, 0)\}.$$

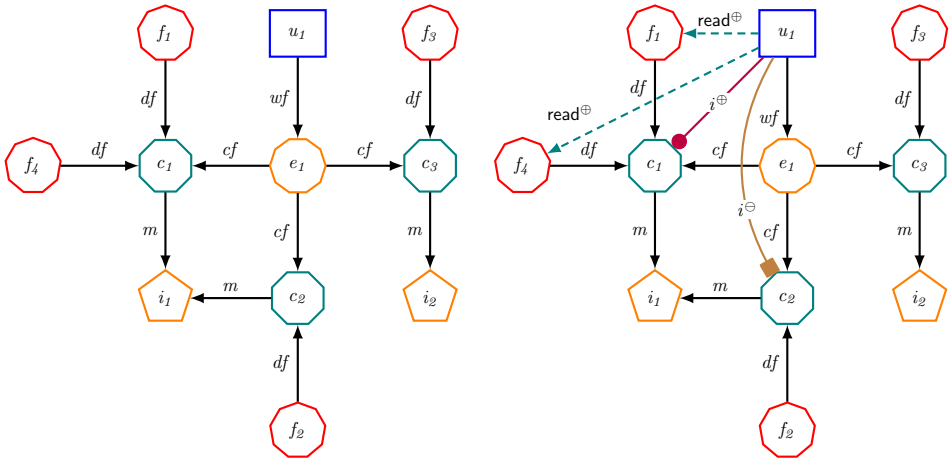
Proposition 6.4. *Given an RPPM Chinese Wall constraint, as described above, for any user u the request (u, o, a) is allowed if the request is authorized by ρ' and (ϱ', χ) and the user u does not have an active interest in any company c' which is a member of the same conflict of interest class as the company $c \neq c'$ responsible for o. In all other cases the request is denied.*

Proof. Let $G = (V, E)$ be a system graph, $E_{i^{\oplus}} = \{(u_1, c_1, i^{\oplus}), \ldots, (u_j, c_j, i^{\oplus})\}$ be a set of active interest edges and $E_{i^{\ominus}} = \{(u_1, c_1, i^{\ominus}), \ldots, (u_k, c_k, i^{\ominus})\}$ be a set of blocked interest audit edges, where $E_{i^{\oplus}} \cup E_{i^{\ominus}} \subset E$ and where for all edges $(u_l, c_l, i^{\ominus}) \in E_{i^{\ominus}}$, $(c_l, i, m) \in E$, $(c_n, i, m) \in E$ and $(u_l, c_n, i^{\oplus}) \in E_{i^{\oplus}}$. Consider a new request (u, o, a), where $G, o, c \models \pi_2$ for some company c. Then:

- If there exists $(u, c, i^{\ominus}) \in E_{i^{\ominus}}$ then (u, o, a) will be denied, using the DenyOverrides CRS, by matching the principal-matching rule $(i^{\ominus} \,;\, \overline{\pi_2}, \mathsf{none}, p_{cw})$ in $\rho' \setminus \rho$ and the authorization rule $(p_{cw}, \star, \star, 0)$ in $\varrho' \setminus \varrho$ being applicable to the request.
- Otherwise, (u, o, a) is authorized if it is authorized by ρ and ϱ.

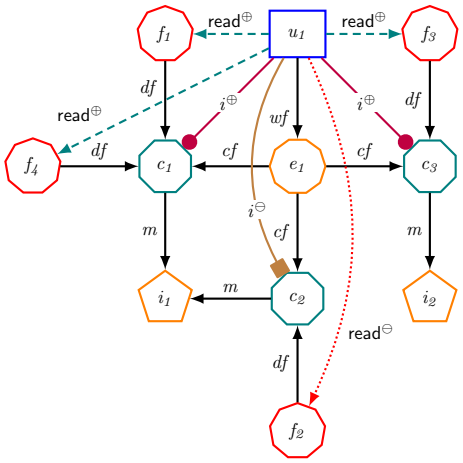
□

[11] The key components of the basic layout are the existence of a subset of system graph entities $C \subset V$ (in my example companies) connecting users to objects, the fact that each $c \in C$ is a member of at most one conflict of interest classes i, and the fact that C is the range for interest audit edges.



(a) System graph fragment

(b) After two requests



(c) After four requests

Figure 6.4: Enforcing the Chinese Wall policy

6.2 Model Comparisons

6.2.1 Policy Configuration Enhancements

The introduction of audit edges into the RPPM$_{1b}$ model enables RPPM to support a range of useful policy frameworks. Whilst audit edges enable a record of previous request evaluations to be kept, some systems may use previous activity to enforce constraint policies such as separation of duty [89, 171] and Chinese Wall [40].

History-Based Policies Audit edges enable RPPM$_{1b}$ to evaluate authorization decisions based on the results of previous request evaluations. Decision audit edges are added automatically to the system graph once request evaluation is completed. In so doing, these edges identify whether the evaluated request was authorized or

denied. Principal-matching rules may employ such edges, within the definition of targets, to require or forbid the results of particular previous actions during the current evaluation. Fong *et al.* have similarly enabled a relationship-based model to enforce history-based policies, in their case by incorporating temporal operators to the logic of Fong's ReBAC model [84]. As I have, they document historical interactions through binary relations, but they choose to track a sequence (or "trace") of graph states, where each state transition is associated with the addition of a single "event" edge representing one those interactions. Whilst this (heavier-weight method) enables them to evaluate policy based on combinations of relationships and events (as I do) their work is developed in the context of social networks, and is thus unsuitable for the more generic access control applications for which the RPPM model was designed.

Separation of Duty and Binding of Duty By using audit edges systematically $RPPM_{1b}$ is able to implement separation of duty constraints, as are commonly employed in role-based access control and workflow systems [31, 39, 56, 114, 163]. The kinds of policy construction used to define separation of duty policies can equally be applied in $RPPM_{1b}$ to produce binding of duty constraints. In the case of role-based access control, separation of duty is commonly divided into two types: static and dynamic (as described in Section 2.2.3). Whilst RBAC's distinction based on role membership and activation doesn't apply to RPPM, in reality the separation of duty mechanism described in Section 6.1.2 is more akin to dynamic separation of duty; this is due to the fact that RPPM's principal-matching determines principals, and thereby applicable constraints, at request-time. Whilst separation of duty has been briefly considered in Fong's ReBAC model [82], binding of duty constraints are not supported by any other relationship-based access control model.

Chinese Wall As with decision audit edges, $RPPM_{1b}$ supports the addition of interest audit edges once request evaluation is completed and the request is authorized. These edges identify a subject's interest in another entity within the system graph (which does not have to be the object of the request, but must be directly or indirectly related to that object). Once again, principal-matching rules may employ these edges when defining targets to ensure conflicting interest requests are denied. Brewer and Nash's seminal paper on the Chinese Wall policy employed an enforcement mechanism based on a history matrix, which records what requests have previously been allowed [40]. It is very natural to record such information as audit edges in the system graph and to use these edges to define and enforce history-

based policies. Chinese Wall has only been given brief consideration in another relationship-based access control model [14].

6.2.2 Implementing Non-ReBAC Models

Multi-Level Security

Section 4.6.2 described how the RPPM$_0$ model may be used to enforce the simple security property of multi-level security. However, RPPM$_0$ is unable to support the dynamic session awareness required to implement the *-property which blocks "writing down". The interest audit edges discussed in Section 6.1.4 offer the mechanism through which this may be achieved in RPPM$_{1b}$.

To use interest audit edges to enforce the *-property, the principal-matching policy of RPPM$_{1b}$ is changed from

$$\begin{aligned}\rho = \ & \{(\mathsf{Cleared\text{-}to}\,;\,\overline{\mathsf{Classified\text{-}at}},\ \mathsf{none},\ \mathsf{cleared\text{-}user}),\\ & (\mathsf{Cleared\text{-}to}\,;\,\mathsf{Dominates}^{+}\,;\,\overline{\mathsf{Classified\text{-}at}},\ \mathsf{none},\ \mathsf{cleared\text{-}user})\}\end{aligned}$$

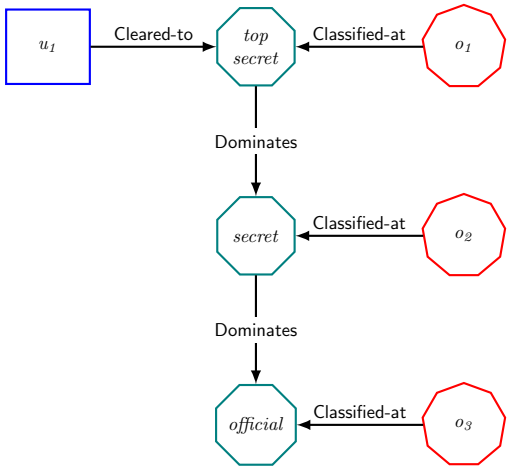
to

$$\begin{aligned}\rho' = \ & \{(\mathsf{Cleared\text{-}to}\,;\,\overline{\mathsf{Classified\text{-}at}},\ i^{\ominus}\,;\,\overline{\mathsf{Classified\text{-}at}},\ \mathsf{cleared\text{-}user}),\\ & (\mathsf{Cleared\text{-}to}\,;\,\mathsf{Dominates}^{+}\,;\,\overline{\mathsf{Classified\text{-}at}},\ i^{\ominus}\,;\,\overline{\mathsf{Classified\text{-}at}},\ \mathsf{cleared\text{-}user})\}.\end{aligned}$$

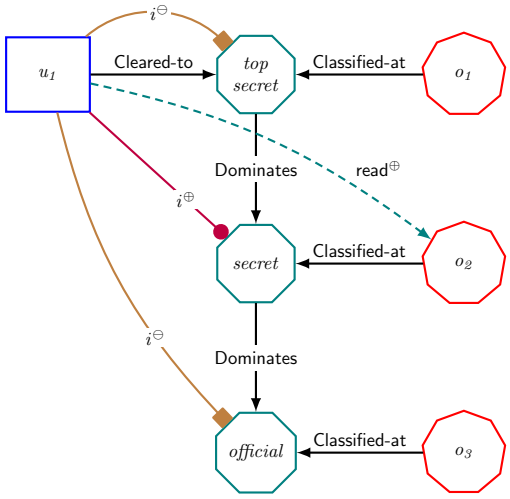
RPPM$_{1b}$ is then configured such that when a request (u, o, a) is authorized, interest audit edges are added between the subject u and each of the security levels (*official*, *secret* and *top secret*). Specifically, an active interest edge $(u, l, i^{\oplus})$ is inserted between the subject u and the level l where $(o, l, \mathsf{Classified\text{-}at}) \in E$. Additionally, blocked interest edges $(u, l', i^{\ominus})$ are inserted between the subject u and all levels $l' \neq l$. (In this way, the security levels are used like the companies of Figure 6.4.) The sole difference from the Chinese Wall policy is that the interest audit edges directed out of a subject must be purged when the subject ends their session with the system. This is because the *-property is a transient, session-specific constraint, unlike Chinese Wall which is permanent.

Let us consider the system graph fragment shown in Figure 6.5a and an initial request $q_1 = (u_1, o_2, \mathsf{read})$. This request will be authorized using the second principal-matching rule of ρ' as no interest edges exist, u_1 is cleared to top secret, and the object o_2 is classified at a level dominated by top secret. The successful authorization results in the insertion of interest audit edges between u_1 and each of the three security levels (and an authorized decision audit edge), as shown in Fig-

ure 6.5b. The presence of the $(u_1, \mathit{official}, i^{\ominus})$ edge means that a subsequent request $q_2 = (u_1, o_3, \mathsf{write})$ would be denied, as neither rule in ρ' would now match.



(a) System graph fragment



(b) After request $q_1 = (u_1, o_2, \mathsf{read})$

Figure 6.5: Enforcing the *-property

It is worth noting that this policy construction not only prevents "writing down" but also prevents "reading down" once a session is active. I assume that a new session is used to request reading at a different security level. If, however, it was desired that users should be able to read down within the same session, the principal-matching

policy required is[12]

$$\begin{aligned}\rho'' = \ & \{(\mathsf{Cleared\text{-}to}\,;\,\overline{\mathsf{Classified\text{-}at}},\ i^{\ominus}\,;\,\overline{\mathsf{Classified\text{-}at}},\ \mathsf{cleared\text{-}user}),\\ & (\mathsf{Cleared\text{-}to}\,;\,\mathsf{Dominates}^{+}\,;\,\overline{\mathsf{Classified\text{-}at}},\ i^{\ominus}\,;\,\overline{\mathsf{Classified\text{-}at}},\ \mathsf{cleared\text{-}user}),\\ & (i^{\oplus}\,;\,\mathsf{Dominates}^{+}\,;\,\overline{\mathsf{Classified\text{-}at}},\mathsf{none},\ \mathsf{cleared\text{-}user})\}.\end{aligned}$$

6.3 Summary

I have introduced audit edges to the base RPPM model to produce RPPM$_{1a}$. I have shown how these edges can support four policy configuration enhancements commonly considered within authorization systems: history-based access control; separation of duty; binding of duty; and Chinese Wall. These enhancements have made RPPM able to handle specific policy frameworks; however, another enhancement, enabling relationships between arbitrary entities to be involved in the evaluation, is desirable to broaden the expressiveness of RPPM's policies. This enhancement is introduced in the next chapter.

[12]Note that the third rule of ρ'' has a required target which begins with $i^{\oplus}$ rather than Cleared-to. This is necessary to prevent reading above the current session level.

Chapter 7

$\mathbf{RPPM_{1c}}$

In this chapter I shall introduce RPPM_{1c} as an RPPM model with a *targeting enhancement* over the base model (RPPM_0):

- Path expressions – this enhancement enables RPPM to support evaluation of authorization requests through the consideration of paths between arbitrary entities within the system graph, rather than just between the subject and object. This ultimately enables RPPM to require or forbid a (limited) *subgraph pattern* when matching a principal.

The basis for this enhancement has previously been published, in conjunction with Jason Crampton, in [64].

7.1 Targeting Enhancement

7.1.1 Path Expressions

The targets within RPPM_0's principal-matching rules comprise path conditions which are required or forbidden between the subject and the object of the request. However, it can also be desirable to consider paths involving or between other entities. (For example, Cheng *et al.* consider paths between any pair from the requester, target(s) and controlling user [51], whilst Stoller supports paths between arbitrary entities [174].)

In some cases these other entities may be individually significant (although not the subject or object of the request), such as when having a relationship with an entity representing a particular team or board conveys "supervisory" or "senior" status in some way. In other cases these other entities may be generally interesting for particular policies, such as when general policies in a university may be oriented around entities of a course or department type, thus enabling a single policy to be

employed across all courses and departments even when evaluating requests between users and coursework papers. As I will demonstrate in ARPPM, see Chapter 8, the ability to consider paths involving entities other than the subject and object is also particularly useful for administrative requests.

To support paths between arbitrary entities in an RPPM$_{1c}$ system graph I introduce the concept of an *extended path condition*, constructed from a path condition with two typed *entity conditions*, one at each end. The entity conditions allow arbitrary entities within the system graph to be specifically identified by their name or their role in the request, or more generally identified by their type. One or more extended path conditions may be grouped together within a set called a *path expression*.[1] Where a path expression contains multiple extended path conditions, the entity conditions within these may use variables to ensure the same entity (or type of entity) is present when satisfying more than one extended path condition.

Definition 7.1. *An* untyped entity condition *is either an entity variable x, an entity v in V, or an entity label from the set $RL = \{\mathsf{subject}, \mathsf{object}\}$.*

The set of entities that *satisfies* an untyped entity condition e with respect to a request $q = (s, o, a)$, denoted $[\![e, q]\!]$, is defined as

$$\begin{aligned} \text{for any entity variable } x,\ [\![x, q]\!] &= V \\ \text{for any entity } v \text{ in } V,\ [\![v, q]\!] &= \{v\} \\ [\![\mathsf{subject}, (s, o, a)]\!] &= \{s\} \\ [\![\mathsf{object}, (s, o, a)]\!] &= \{o\}. \end{aligned}$$

Remark 7.1. *Whilst it may be desirable to identify specific entities within a system graph using an untyped entity condition, it may be more useful to identify entities generally by their type. For example, in the case of my higher education example, I may require a path of relationships exist between the subject of the request (specifically identified using the untyped entity condition* subject*) and any entity of the course type. Such an approach enables policies to be defined independent of the specific courses being run by the university. Therefore, it is desirable to define the concept of a* typed entity condition*, but before I do so it is important to note that the entities represented by an untyped entity condition still have a type themselves, and this is easily identified where an untyped entity condition is satisfied by a singleton set with respect to a request.*

Given a request q, for an untyped entity condition $e = v$, $\tau(e) = \tau(v)$. For

[1] This approach is inspired by that outlined by Stoller; however, I provide a complete formal definition of the components and of their processing [174].

$e \in RL$, $\tau(e) = \tau(u)$ *where* $u \in [\![e, q]\!]$. *For example, if* $q = (s, o, a)$ *and* $e =$ *subject*, $\tau(e) = \tau(s)$.

Definition 7.2. *A* typed entity condition *has the form* (e, f)*, where* e *is an untyped entity condition and* f *either belongs to* T *or is a type variable.*

Let Var_V denote the set of entity variables and Var_T denote the set of type variables. I assume that $Var_V \cap Var_T = \emptyset$. The set of entities that *satisfies* a typed entity condition (e, f) with respect to a request q, denoted $[\![(e, f), q]\!]$, is defined as

$$[\![(e, f), q]\!] = \begin{cases} V & \text{if } f \in Var_T,\ e \in Var_V \\ \{v : \tau(v) = f\} & \text{if } f \in T,\ e \in Var_V \\ \emptyset & \text{if } f \in T,\ e \notin Var_V,\ \tau(e) \neq f \\ [\![e, q]\!] & \text{otherwise.} \end{cases}$$

A typed entity condition (e, f) in which e is not an entity variable is said to be a *simple* typed entity condition. Note that for any simple typed entity condition (e, f) and any request q, $[\![(e, f), q]\!]$ is either the empty set or $[\![e, q]\!]$ (which, in turn, is either a singleton set or the empty set). For example, the simple typed entity condition $(u, \tau(u))$, where $u \in V$, is satisfied by $\{u\}$ for any request. Henceforth, I will be working exclusively with typed entity conditions and I will, therefore, refer to entity conditions and simple entity conditions.

Definition 7.3. *An* extended path condition *has the form* $\eta \cdot \pi \cdot \eta'$*, where* η *is a simple entity condition,* η' *is an entity condition, and* π *is a path condition.*

I define satisfaction of an extended path condition $\eta \cdot \pi \cdot \eta'$ based on the satisfaction of the contained path condition π between entities meeting the specified entity conditions, η and η'. Let

$$\nu : Var_V \cup Var_T \to V \cup T,$$
$$\text{where } \nu(e) \in V \text{ for } e \in Var_V \text{ and } \nu(f) \in T \text{ for } f \in Var_T,$$

be a *variable mapping function* which maps entity variables to entities and type variables to types. I write $\nu(\eta) = (\nu(x), \nu(y))$ to denote the entity condition $\eta = (x, y)$ once variables have been replaced by entities and types.

Definition 7.4. *An extended path condition* $\eta \cdot \pi \cdot \eta'$ *is* satisfied *by system graph* G *and request* q*, denoted* $G, q \models \eta \cdot \pi \cdot \eta'$*, if and only if there exists a variable mapping function* ν *such that:*

- $[\![\nu(\eta), q]\!] \neq \emptyset$, *and*
- *There exists* $v \in [\![\nu(\eta'), q]\!]$ *such that* $\exists u \in [\![\nu(\eta), q]\!]$, $G, u, v \models \pi$.[2]

I write $G, q \not\models \eta \cdot \pi \cdot \eta'$ *otherwise.*

An extended path condition may not be satisfied even if it contains no variables. In particular, given the extended path condition $(e, f) \cdot \pi \cdot (e', f')$,

- It may be the case that $e, e' \in V$ but $G, e, e' \not\models \pi$; or
- It may be the case that $e \in RL$ and $f \neq \tau(e)$; or
- It may be the case that $e' \in RL$ and $f' \neq \tau(e')$.

In each of these cases, there can be no mapping ν such that the extended path condition is satisfied.

It should be clear that extended path conditions could be used to define the targets within a principal-matching rule; however, then entity variables and type variables could only parameterize the entity conditions at either end of a single required or forbidden path. I wish to make use of such variables across multiple required or forbidden paths; I, therefore, define a *path expression* as a set of extended path conditions, and I enable path expressions to be used as targets within principal-matching rules (alongside the special targets all and none).

Definition 7.5. *Given a principal-matching rule* (ϕ, ψ, p)*, where the* required target ϕ *is a path expression* $\{\eta_1 \cdot \pi_1 \cdot \eta'_1, \ldots, \eta_j \cdot \pi_j \cdot \eta'_j\}$*,* ϕ *is* satisfied *by system graph* G *and request* q*, denoted* $G, q \models \phi$*, iff there is a function* $\nu_\phi : Var_V \cup Var_T \to V \cup T$ *such that for all* $i \in \{1, \ldots, j\}$*:*

- $[\![\nu_\phi(\eta_i), q]\!] \neq \emptyset$ *and*
- *There exists* $v \in [\![\nu_\phi(\eta'_i), q]\!]$ *such that* $G, [\![\nu_\phi(\eta_i), q]\!], v \models \pi_i$.

The required target ϕ *is not satisfied by* G *and* q*, denoted* $G, q \not\models \phi$*, otherwise.*

Definition 7.6. *Given a principal-matching rule* (ϕ, ψ, p)*, where the* forbidden target ψ *is a path expression* $\{\eta_1 \cdot \pi_1 \cdot \eta'_1, \ldots, \eta_k \cdot \pi_k \cdot \eta'_k\}$*,* ψ *is* not satisfied *by system graph* G *and request* q*, denoted* $G, q \not\models \psi$*, iff for all functions* $\nu_\psi : Var_V \cup Var_T \to V \cup T$ *I have:*

- $[\![\nu_\psi(\eta_i), q]\!] = \emptyset$ *or*
- $G, [\![\nu_\psi(\eta_i), q]\!], v \not\models \pi_i$ *for all* $v \in [\![\nu_\psi(\eta'_i), q]\!]$,

[2]Note that as η is required to be a simple entity condition in Definition 7.3, $[\![\nu(\eta), q]\!]$ is either a singleton set or the empty set. Therefore, if there does exist $u \in [\![\nu(\eta), q]\!]$, then $[\![\nu(\eta), q]\!] = \{u\}$.

for all $i \in \{1, \ldots, k\}$. *The forbidden target* ψ *is satisfied by* G *and* q, *denoted* $G, q \models \psi$, *otherwise.*

Informally, these definitions ensure that a principal-matching rule constructed using path expression targets is conceptually equivalent to a sequence of rules in a policy graph fragment, where a single variable mapping function maps variables to entities and types in that fragment for each target.[3] So, for example, the principal-matching rule

$$(\{\eta_1 \cdot \pi_1 \cdot \eta_1', \eta_2 \cdot \pi_2 \cdot \eta_2', \eta_3 \cdot \pi_3 \cdot \eta_3'\}, \{\eta_4 \cdot \pi_4 \cdot \eta_4', \eta_5 \cdot \pi_5 \cdot \eta_5', \eta_6 \cdot \pi_6 \cdot \eta_6'\}, p)$$

is conceptually equivalent to the policy graph fragment shown in Figure 7.1. The principal p is only matched to the request if all of the rules are matched (i.e., all three required extended path conditions are satisfied and all three forbidden extended path conditions are not satisfied).

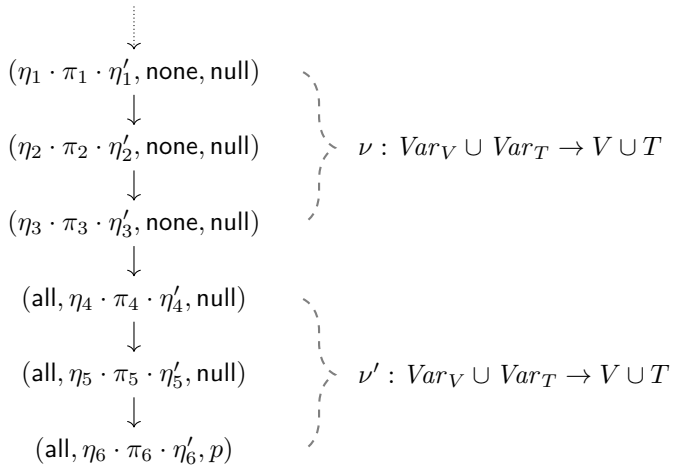


Figure 7.1: Policy graph equivalent of path expressions

Example 7.1. *Returning once again to my higher education example (introduced in Example 4.3), Figure 7.2 shows a slightly more detailed system graph including the department, dept. 1, within which the example is set. Recall the original principal-matching rule through which the* course-ta *principal is matched.*

$$\rho = \{\ldots, (\mathit{Ta\text{-}for}\,;\, \overline{\mathit{Coursework\text{-}for}},\ \mathit{Enrolled\text{-}on}\,;\, \overline{\mathit{Coursework\text{-}for}},\ \mathit{course\text{-}ta}), \ldots\}$$

It may be desirable to define a more specific policy for this principal using targets

[3] Note that whilst this conceptual equivalency exists, path expression targets may be used in a set-based principal-matching policy as well as a graph-based one.

based on path expressions.

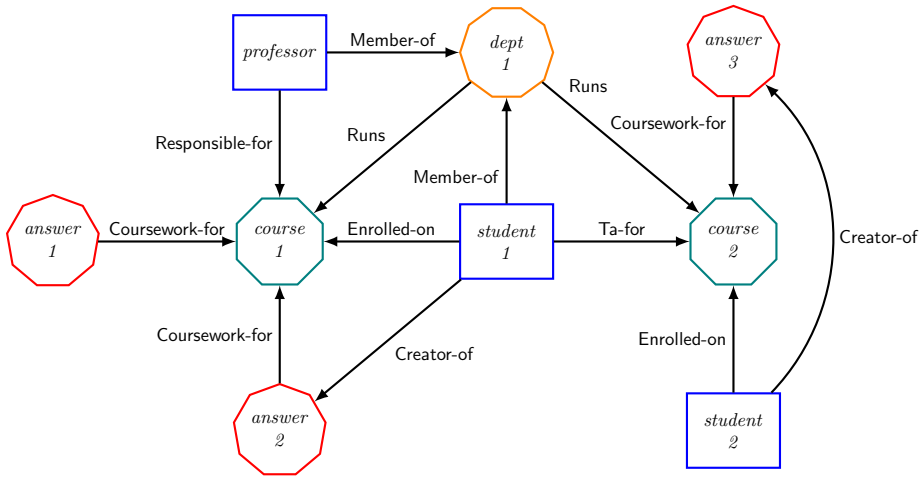
$$\begin{aligned}\rho' = \{\ldots, (\{&(\textsf{subject}, user) \cdot \textsf{Member-of} \cdot (dept\ 1, department),\\ &(\textsf{subject}, user) \cdot \textsf{Ta-for} \cdot (ev_1, course),\\ &(dept\ 1, department) \cdot \textsf{Runs} \cdot (ev_1, course),\\ &(\textsf{object}, coursework) \cdot \textsf{Coursework-for} \cdot (ev_1, course)\},\\ &\{(\textsf{subject}, tv_1) \cdot \textsf{Enrolled-on}\,;\, \overline{\textsf{Coursework-for}} \cdot (\textsf{object}, tv_2)\},\\ &\textsf{course-ta}), \ldots\}\end{aligned}$$

If we, once again, consider the request $(student\ 1, answer\ 1, \textsf{read})$*, then the* course-ta *principal will be matched with*

$$\begin{aligned}[\![(\textsf{subject}, user)]\!] &= \{student\ 1\},\\ [\![(dept\ 1, department)]\!] &= \{dept\ 1\},\\ [\![(ev_1, course)]\!] &= \{course\ 1\},\\ [\![(\textsf{object}, coursework)]\!] &= \{answer\ 1\},\\ [\![(\textsf{subject}, tv_1)]\!] &= \{student\ 1\},\\ [\![(\textsf{object}, tv_2)]\!] &= \{answer\ 1\},\\ G, student\ 1, dept\ 1 &\models \textsf{Member-of},\\ G, student\ 1, course\ 1 &\models \textsf{Ta-for},\\ G, dept\ 1, course\ 1 &\models \textsf{Runs},\\ G, coursework\ 1, course\ 1 &\models \textsf{Coursework-for},\\ G, student\ 1, coursework\ 1 &\not\models \textsf{Enrolled-on}\,;\, \overline{\textsf{Coursework-for}}.\end{aligned}$$

Remark 7.2. *Note that a variable occurring only once in a path expression is essentially a "free" variable and can be assigned to any entity (or type, depending on the nature of the variable). Hence,* $\{(\textsf{subject}, f) \cdot \pi \cdot (\textsf{object}, f')\}$*, where* $f, f' \in Var_T$ *and* $f \neq f'$*, is equivalent to the RPPM path condition* π*. In other words, path expressions of* $RPPM_{1c}$ *are backwards compatible with path conditions of* $RPPM_0$*.*

Remark 7.3. *It is also important to note the requirement that an extended path condition begin with a* simple *entity condition. This means that an extended path condition may only be satisfied by a path starting from a single specific entity within the system graph, which significantly reduces the complexity of request evaluation. I do not believe that such a constraint will significantly reduce the applicability of the* $RPPM_{1c}$ *model, especially as* $\overline{\pi}$ *enables all extended path conditions to be reversed.*



(a) System graph fragment

(b) Type legend

Figure 7.2: Another higher education system graph fragment

Through judicious specification of extended path conditions with multiple occurrences of appropriate entity and type variables (as demonstrated in Example 7.1), it is possible to define targets which result in the evaluation of authorization decisions based on the matching of *subgraph patterns*, rather than simply a path between two arbitrary entities. Whilst highly complex patterns are precluded due to the requirement for a simple entity condition at the start of each extended path condition, path expressions can be used to specify tractable "subgraph isomorphism" tests within the RPPM$_{1c}$ authorization evaluation.

Request Evaluation

Recall from Section 4.5.1 that RPPM$_0$ employs non-deterministic finite automata (NFA) as the core component of its request evaluation process. Specifically, in the ComputePrincipals algorithm (Algorithm 4.2) I use the existence of intersection NFA accepting languages to match principals, where each intersection NFA combines the request NFA $M_q = (V, \widetilde{R}, E, s, \{o\})$ with a target path condition automaton.[4] As RPPM$_{1c}$'s principal-matching rule targets are path expressions, and these may contain multiple extended path conditions each of which must be satisfied (or not) between entities identified by their entity conditions, I may need to consider multiple intersection NFA for each target. I achieve this for each extended path condition

[4]This underlying process was unchanged by the introduction of policy graph evaluation in RPPM$_{1a}$'s variant ComputePrincipals algorithm, as shown in Algorithm 5.1.

$(e_i, f_i) \cdot \pi_i \cdot (e'_i, f'_i)$ by:[5]

- Using the simple entity condition (e_i, f_i) to determine the single start state for M_{q_i};
- Using the entity condition (e'_i, f'_i) to determine the set of accepting states for M_{q_i};
- Retrieving the path condition automaton M_{π_i}; and
- Producing the intersection NFA $M_{\cap_i}$ of M_{q_i} and M_{π_i}.

I then ensure that across all of the intersection NFA for the path expression target, entity conditions using the same entity variable identify the same entity and those using the same type variable identify the same type of entity. Satisfaction of each extended path condition (and, thereby, the path expression) is then determined by the existence of an accepting language for each intersection NFA constrained by the entity conditions.

Formally, RPPM$_{1c}$ determines the set of matched principals using a path expression variant of the ComputePrincipals algorithm, as shown in Algorithm 7.1. This path expression variant requires several changes to the calling arguments when compared to the original ComputePrincipals algorithm. Specifically, Algorithm 7.1 includes three arguments necessary to determine satisfaction of entity conditions: the set of types T; the set of entity variables Var_V; and the set of type variables Var_T. The algorithm employs a new DetermineTargetSatisfaction algorithm (Algorithm 7.2) to perform the key NFA and path expression-related processing required to determine whether a path expression target is satisfied within the system graph, in the context of the current request.

Algorithm 7.1 ComputePrincipals (path expression variant)

Require: System graph $G = (V, E)$, set of relationship labels $\widetilde{R}$, set of types T, request $q = (s, o, a)$, set of entity variables Var_V, set of type variables Var_T and principal-matching policy ρ
Ensure: Returns set of matched principals $[\![\rho]\!]$

```
1: [[ρ]] ← ∅
2: M_q ← (V, R̃, E, s, {o})
3: for all (φ, ψ, p) ∈ ρ do
4:     if DetermineTargetSatisfaction(G, R̃, T, q, Var_V, Var_T, φ, true) then
5:         if DetermineTargetSatisfaction(G, R̃, T, q, Var_V, Var_T, ψ, false) then
6:             [[ρ]] ← [[ρ]] ∪ p
7:         end if
8:     end if
9: end for
10: return [[ρ]]
```

[5] I advocate pre-computing, as far as possible, the generic request NFA M_q (which forms the basis for each extended path condition-specific automaton M_{q_i}) and the path condition automata for all path expressions in the principal-matching policy so as to reduce the processing involved during request evaluation.

Specifically, DetermineTargetSatisfaction performs four distinct activities:

1. It short-circuits evaluation if the target being tested is a special target;
2. It uses the DeterminePossibleECMF algorithm (Algorithm 7.3) to determine the possible entity condition mappings which produce non-empty accepting languages for the intersection NFA;
3. It uses the ValidateECMF algorithm (Algorithm 7.4) to determine the valid entity condition mappings which (as well as being possible) are consistent in variable usage; and
4. It uses the validated mappings to determine whether a required target is satisfied or a forbidden target is not satisfied.

Algorithm 7.2 DetermineTargetSatisfaction

Require: System graph $G = (V, E)$, set of relationship labels $\widetilde{R}$, set of types T, request $q = (s, o, a)$, set of entity variables Var_V, set of type variables Var_T, target φ (one of the special targets all or none, or a path expression containing $|\varphi|$ extended path conditions), and required target indicator rti
Ensure: Returns **true** if the target satisfaction matches the required target indicator and **false** otherwise

```
// short-circuit evaluation for special targets
if (φ = all) and (rti = true) then
    return true
else if (φ = none) and (rti = false) then
    return true
else if (φ = all) or (φ = none) then
    return false
else
    // instantiate entity condition mapping function (initially all values are ∅)
    f_EC : ℕ_{≤|φ|} × {start-node, end-node} → 2^V
    // update entity condition mapping function with possible entities
    f_EC ← DeterminePossibleECMF(G, R̃, q, φ, f_EC)
    // update entity condition mapping function with valid entities
    f_EC ← ValidateECMF(R̃, T, Var_V, Var_T, φ, f_EC)
    if rti = true then
        // required target satisfied if all extended path conditions are satisfied between valid entities for both their entity conditions
        if ∀(e_n, f_n) · π_n · (e'_n, f'_n) ∈ φ: (f_EC(n, start-node) ≠ ∅) and (f_EC(n, end-node) ≠ ∅) then
            return true
        end if
    else
        // forbidden target not satisfied if there aren't pairs of valid entities between which all extended path conditions are satisfied
        if ∀(e_w, f_w) · π_w · (e'_w, f'_w) ∈ φ: (f_EC(w, start-node) = ∅) or (f_EC(w, end-node) = ∅) then
            return true
        end if
    end if
    return false
end if
```

The DeterminePossibleECMF algorithm (shown in Algorithm 7.3) iterates through each extended path condition identifying possible start and accepting states which contribute to each accepting intersection NFA. The entity condition mapping function is updated to reflect these possible entities from the system graph.

Algorithm 7.3 DeterminePossibleECMF

Require: System graph $G = (V, E)$, set of relationship labels $\widetilde{R}$, request $q = (s, o, a)$, path expression target φ containing $|\varphi|$ extended path conditions, and an entity condition mapping function f_{EC}
Ensure: Returns an updated entity condition mapping function f_{EC}

for all $(e_i, f_i) \cdot \pi_i \cdot (e'_i, f'_i) \in \varphi$ **do**
  $f_{EC}(i, \text{start-node}) \leftarrow \emptyset$
  $f_{EC}(i, \text{end-node}) \leftarrow \emptyset$
  $s_{M_{q_i}} \leftarrow [\![(e_i, f_i), q]\!]$
  $F_{M_{q_i}} \leftarrow [\![(e'_i, f'_i), q]\!]$
  if $(s_{M_{q_i}} = \emptyset)$ **or** $(F_{M_{q_i}} = \emptyset)$ **then**
    continue
  end if
  $M_{\pi_i} \leftarrow \text{PathConditionNFA}(\pi_i)$ // produce $M_{\pi_i} = (Q_{\pi_i}, \Sigma_{\pi_i}, \delta_{\pi_i}, s_{\pi_i}, F_{\pi_i})$ as per Section 4.2.4
  $M_{q_i} \leftarrow (V, \widetilde{R}, E, s_{M_{q_i}}, F_{M_{q_i}})$
  $M_{\cap_i} \leftarrow M_{\pi_i} \cap M_{q_i}$ // note $M_{\cap_i} = (V \times Q_{\pi_i}, \widetilde{R} \cap \Sigma_{\pi_i}, \delta_{\cap_i}, (s_{M_{q_i}}, s_{\pi_i}), F_{M_{q_i}} \times F_{\pi_i})$
  if $L(M_{\cap_i}) = \emptyset$ **then**
    continue
  else
    $f_{EC}(i, \text{start-node}) \leftarrow s_{M_{q_i}}$
    $AP \leftarrow \{h_0, \ldots, h_x : h_0 = (s_{M_{q_i}}, s_{\pi_i}), h_y \in V \times Q_{\pi_i}$ *for* $y > 0, h_x \in F_{M_{q_i}} \times F_{\pi_i},$
      $(h_z, h_{z+1}, \sigma_{z+1}) \in \delta_{\cap_i}$ *for* $0 \leqslant z \leqslant x - 1\}$
    $f_{EC}(i, \text{end-node}) \leftarrow \{u_{q_i} \in V : h_0, \ldots, h_x \in AP, h_x = (u_{\pi_i}, u_{q_i})\}$
  end if
end for
return f_{EC}

The ValidateECMF algorithm (shown in Algorithm 7.4) also iterates through each extended path condition; however, it refines the entity condition mapping function values based on variable use within the entity conditions of all the extended path conditions. Only entities which are consistently possible for all uses of a variable remain in the mapping function.

7.2 Model Comparisons

7.2.1 Targeting Enhancement

Path Expressions Path expressions in the RPPM$_{1c}$ model enable RPPM to evaluate principal-matching rule targets between arbitrary entities, and can even be used to evaluate authorization decisions in respect of (limited) subgraph patterns. Most other relationship-based access control models are far less expressive. Whilst Bruns *et al.*'s ReBAC hybrid logic grammar is able to define chains between the requestor and arbitrary third-parties, their request processing requires the policy to be satisfied between the requestor and the resource owner [41].[6] Cheng *et al.* offer some improvement by enabling their graph rules (described in Section 2.4.3) to be evaluated between a pair from the requester, target(s) and controlling user, depending in which of their policies the rule is defined [51]. Whilst this is more expressive

[6] Although their hybrid operators allow the path to 'jump' during that chain.

Algorithm 7.4 ValidateECMF

Require: Set of relationship labels $\widetilde{R}$, set of type T, set of entity variables Var_V, set of type variables Var_T, path expression target φ containing $|\varphi|$ extended path conditions, and an entity condition mapping function f_{EC}
Ensure: Returns an updated entity condition mapping function f_{EC}

for all $(e_j, f_j) \cdot \pi_j \cdot (e'_j, f'_j) \in \varphi$ **do**
  if $f_j \in Var_T$ **then**
    // simple entity condition containing a type variable
    $T_{j_{\mathsf{sn}}} \leftarrow \{t \in T : \exists u_{j_{\mathsf{sn}}} \in f_{EC}(j, \mathsf{start\text{-}node}), \tau(u_{j_{\mathsf{sn}}}) = t\}$
    if $f'_j = f_j$ **then**
      $f_{EC}(j, \mathsf{end\text{-}node}) \leftarrow \{u_{j_{\mathsf{en}}} \in f_{EC}(j, \mathsf{end\text{-}node}) : \tau(u_{j_{\mathsf{en}}}) \in T_{j_{\mathsf{sn}}}\}$
    end if
    for all $(e_k, f_k) \cdot \pi_k \cdot (e'_k, f'_k) \in \varphi$ **do**
      if $(k \neq j)$ **and** $(f_k = f_j)$ **then**
        $f_{EC}(k, \mathsf{start\text{-}node}) \leftarrow \{u_{k_{\mathsf{sn}}} \in f_{EC}(k, \mathsf{start\text{-}node}) : \tau(u_{k_{\mathsf{sn}}}) \in T_{j_{\mathsf{sn}}}\}$
      else if $(k \neq j)$ **and** $(f'_k = f_j)$ **then**
        $f_{EC}(k, \mathsf{end\text{-}node}) \leftarrow \{u_{k_{\mathsf{en}}} \in f_{EC}(k, \mathsf{end\text{-}node}) : \tau(u_{k_{\mathsf{en}}}) \in T_{j_{\mathsf{sn}}}\}$
      end if
    end for
  else if $f'_j \in Var_T$ **then**
    // entity condition containing a type variable
    $T_{j_{\mathsf{en}}} \leftarrow \{t \in T : \exists u_{j_{\mathsf{en}}} \in f_{EC}(j, \mathsf{end\text{-}node}), \tau(u_{j_{\mathsf{en}}}) = t\}$
    for all $(e_l, f_l) \cdot \pi_l \cdot (e'_l, f'_l) \in \varphi$ **do**
      if $(l \neq j)$ **and** $(f_l = f'_j)$ **then**
        $f_{EC}(l, \mathsf{start\text{-}node}) \leftarrow \{u_{l_{\mathsf{sn}}} \in f_{EC}(l, \mathsf{start\text{-}node}) : \tau(u_{l_{\mathsf{sn}}}) \in T_{j_{\mathsf{en}}}\}$
      else if $(l \neq j)$ **and** $(f'_l = f'_j)$ **then**
        $f_{EC}(l, \mathsf{end\text{-}node}) \leftarrow \{u_{l_{\mathsf{en}}} \in f_{EC}(l, \mathsf{end\text{-}node}) : \tau(u_{l_{\mathsf{en}}}) \in T_{j_{\mathsf{en}}}\}$
      end if
    end for
  else if $e'_j \in Var_V$ **then**
    // entity condition containing an entity variable
    for all $(e_m, f_m) \cdot \pi_m \cdot (e'_m, f'_m) \in \varphi$ **do**
      if $(m \neq j)$ **and** $(e'_m = e'_j)$ **then**
        $f_{EC}(m, \mathsf{end\text{-}node}) \leftarrow \{u_{m_{\mathsf{en}}} \in f_{EC}(m, \mathsf{end\text{-}node}) : u_{m_{\mathsf{en}}} \in f_{EC}(j, \mathsf{end\text{-}node})\}$
      end if
    end for
  end if
end for
return f_{EC}

than approaches which limit the evaluation of paths to between a particular pair of entities (frequently the subject and the object or the subject and the object's "owner") [41, 43, 44, 52, 82, 85], their limited extension is focused on supporting the various sources of policy in their model. The approach outlined by Stoller, in his RPPM^2 model, is the only one other than mine (which it inspired) able to support paths between arbitrary entities [174]. However, mine is less restrictive:[7]

- Firstly, RPPM$_{1c}$ enables me to constrain path conditions to entities by their label, their type, their role in the request, or one of two kinds of variable (which can be used to constrain multiple paths to involving the same entity, without specifying beforehand the identity of that entity). In contrast Stoller constrains them to entities by their labels or a variable.

[7] As an aside, Stoller also replaced RPPM's principal-matching policy with a path-expression naming mechanism. I believe that the two-step request evaluation process is far more powerful a concept than simply a shorthand naming mechanism (as demonstrated, for example, by policy graph evaluation in Chapter 5).

- Secondly, the RPPM^2 authorization checking algorithm entirely rejects rules which begin with "unbound variables" whilst I only preclude rules which begin with a constraint which may be satisfied by more than one entity in the system graph.

Authorization Rule Targets

In [64] Jason Crampton and I introduced required and forbidden path expression targets into RPPM's authorization rules (in place of the element X identifying the objects to which the rule applied). The intention of this modification was to mirror target use in RPPM's principal-matching rules. In this thesis, however, I do not modify my existing authorization rule structure in this way, as this change would reduce the expressiveness of RPPM$_{1c}$'s policy, rather than enhance it. Specifically, any policy which may be created using the definition of principal-matching rules and authorization rules in [64] may be defined in RPPM$_{1c}$ whilst limiting the authorization rules (p, X, Y, b) so that $X = \star$.[8]

It should be clear that a policy, defined as per [64],

$$\rho_{[64]} = \{(\phi, \psi, p)\}$$
$$\varrho_{[64]} = \{(p, \phi', \psi', y, d)\}$$

is equivalent to the RPPM$_{1c}$ policy[9], where conjunction in ρ is achieved using a graph-based policy from RPPM$_{1a}$,

$$\rho = \{(\phi \cup \phi', \psi \cup \psi', p)\}$$
$$\varrho = \{(p, \star, \{y\}, d)\}.$$

Further, if multiple authorization rules existed in $\varrho_{[64]}$ for the principal p, these could be accommodated in RPPM$_{1c}$ by utilising multiple distinct principals. So, for example, the policy

$$\rho_{[64]} = \{(\phi, \psi, p)\}$$
$$\varrho_{[64]} = \{(p, \phi', \psi', y', 0), (p, \phi'', \psi'', y'', 1)\}$$

[8] Without limiting X to $\star$, RPPM$_{1c}$'s policies may be more expressive than those of [64].
[9] Although if $y = \star$ in $\varrho_{[64]}$, then $\{y\}$ is replaced with $\star$ in ϱ.

is equivalent to the RPPM$_{1c}$ policy[10]

$$\rho = \{(\phi \cup \phi', \psi \cup \psi', p'), (\phi \cup \phi'', \psi \cup \psi'', p'')\}$$
$$\varrho = \{(p', \star, \{y'\}, 0), (p'', \star, \{y''\}, 1)\}.$$

7.3 Summary

I have introduced path expressions to the targets of the base RPPM model's principal-matching rules to produce RPPM$_{1c}$. I have also shown how these enable request evaluation to consider paths of relationships between arbitrary entities within the system graph. Whilst this enhancement has made RPPM's policy language more expressive, a clear approach to administration is required (and introduced in the next chapter) to enable RPPM to be implemented and operated in a controlled, secure manner.

[10] Although, if $y' = \star$ in $\varrho_{[64]}$, then $\{y'\}$ is replaced with $\star$ in ϱ. The same applies to y''.

Part II

Applying RPPM

Chapter 8

ARPPM

For any access control model to be effective it must keep track of the current state of the system it models. However, most systems are not static and, therefore, the associated access control model instance will not be static either. Even in the case of systems which are static, initial configuration will normally be required before the model can be used. Therefore, no matter the reason behind a change to a model instance, there is a need for that change to be controlled. Without this control, unauthorized changes could cause legitimate authorization requests to be denied (type I errors), or allow illegitimate authorization requests to be approved (type II errors). Uncontrolled changes could also cause the model instance to become inconsistent with the model's underlying assumptions or requirements. For all of these reasons, the administration of an access control model instance is a critical aspect of the model's ongoing effectiveness.

In this chapter I present ARPPM, a model based on RPPM_{1c} and containing a number of *administrative enhancements* to support the management of RPPM model instances:

- Administrative requests and policies – this enhancement introduces an administrative form of authorization request and updates RPPM_{1c}'s entity conditions so that they may be satisfied by participants of such requests. It also defines applicability of authorization rules in respect of administration requests and defines a default administrative decision function.

- Initial system graph – the specification of an initial system graph state ensures that an ARPPM model instance may be managed completely and without extra-model authorization. I provide a recommended initial state and also detail the requirements for the generalised initial state.

The basis for these administrative enhancements has previously been published, in

conjunction with Jason Crampton, in [64]. As motivation for these enhancements I first detail administrative requirements necessary for any RPPM administrative model and highlight that these requirements are not fully met by existing administrative features of relationship-based access control models.

8.1 High-Level Approach

There are various high-level approaches to the administration of access control models. Models supporting mandatory access control (MAC) employ security labels (as discussed in Section 2.1.4) assigned to subjects by a security controller. The ability of those subjects to perform particular operations is then determined by a comparison of their label with that of the object with which they are interacting [23, 24]. The policy which defines the acceptable combinations of subject and object labels for a particular operation is system-wide. Usually only the security controller can modify labels for subjects, and so the administrative policies are reasonably simple [166]. In contrast, models supporting discretionary access control (DAC) employ names for subjects and objects. Particular operations on an object are then controlled at a named individual (or named group) level [69]. The definition of policies granting permission to named individuals or groups is carried out by any authorized subject within the system. This allows for a wider variety of administrative architectures and leads to the notions of a hierarchy of administrators, administrative scope and resource ownership [166].

Whatever the high-level approach to administration, the mechanism through which the access control policies are administered has a significant impact on the security of the protection system. Recall from Section 2.1.2 the eight design principles identified by Saltzer and Schroeder [162]. Whilst these principles were not specifically targeted at securing the administration of the access control model, it is clear that the "complete mediation" principle requires that such changes be managed and that the "economy of mechanism", "fail-safe defaults", "separation of privilege" and "least privilege" principles may all be relevant to such management.

Given that access control models are themselves designed to evaluate authorization requests, it is common for access control models to mediate both operational and administrative requests. For the access control model designer such an approach is instinctive and preserves an *economy of concept*, thus easing human comprehension in a manner in line with Saltzer and Schroeder's "psychological acceptability" principle. Whilst this approach is commonly employed [64, 110, 140, 157, 163, 174], it has, however, been identified by some as being in line with the "economy of

mechanism" design principle [124, 174]. In actuality that principle requires that a design be "simple and small as possible" [162], which may not, necessarily, always align with the natural desire to protect the protection mechanism with its own capabilities. There may be particular circumstances when such a meta-model approach leads to significant complexities which warrant an alternative arrangement. Therefore, preserving economy of concept may not always be compatible with preserving "economy of mechanism". That said, in the case of access control model administration I believe such arrangements to be the exception.

8.1.1 RPPM Administration Requirements

Whilst administration has been given some, limited, consideration with respect to relationship-based access control, I believe a requirements-driven approach, as used by Li and Mao for UARBAC [124], will avoid the existing issues of proposed solutions (identified in Section 8.1.2). I, therefore, first identify requirements for any administrative RPPM model. When specifying these requirements I consider commonly applied best practice as well as desirable properties relevant to the practicality, robustness and usability of implementations of relationship-based access control.[1]

AR1: Use the model's concepts to administer itself As discussed above, it is only natural that authorization of administrative requests in respect of an access control model should be determined using the model itself (see ARBAC [163] for RBAC [79], for example). My first requirement is, therefore, to *use the model's concepts to administer itself.*

AR2: Control the addition and deletion of model components Given that an update action may be modelled through the combined use of deletion and addition actions, my second requirement is that the model *control the addition and deletion of model components.* As RPPM employs a graph-based model of a system, I chose to perform (and control) administration using two actions: addEdge and deleteEdge.

AR3: Support multiple administrators It is widely accepted that the use of a single "global" administrator account to perform all administrative tasks is undesirable from organisational, auditing and "least privilege" perspectives [162]. My third requirement is, therefore, that the model be able to *support multiple administrators.*

[1] Since originally presenting these requirements in [64] I have removed two requirements. The requirement to "Support multiple controlling paths" has been removed as I have chosen to introduce path expressions as a separate, targeting enhancement, described in Chapter 7. The requirement to "Perform request evaluation efficiently" has also been removed as this was a vague and imprecise requirement (as noted by one of the anonymous reviewers) which isn't focused, specifically, on administration.

In the case of RPPM, I require that any authorized entity be able to perform administrative actions where the relationships necessary to satisfy the specified administrative policies exist. If the only authorized entity is a central security controller then RPPM may be managed in a mandatory manner. However, the creation of these relationships through administrative requests allows administrators to delegate their control to others, thus producing a hierarchy of administrators. Further, the specifics of these relationships enable administrative scopes to be created. In this way, a discretionary approach to administration may be taken with an administrative request made by an entity only authorized when made in a context to which they have the necessary relationships.

AR4: Exclude extra-model authorization Many administrative models rely heavily on a benevolent, correct and complete extra-model administrator in order to boot-strap and manage the intra-model administrative capabilities. I require that the model should *exclude extra-model authorization* and be self-contained, ensuring "complete mediation" [162]. Specifically, I require the model to be used to authorize all changes to the model instance such that all necessary administrative requests are evaluated within the bounds of the instance.

AR5: Control changes to all parts of the model In order to be a complete model of administration it is necessary for administrative requests targeting types, permissible relationships, entities, relationships, and access control policies and their elements to be managed. The fifth requirement to *control changes to all parts of the model* increases the utility of the administrative model and is in line with the "complete mediation" design principle [162], drawing parallels with AR4. As the RPPM model utilises paths in the system graph to evaluate and control actions, I require that the system graph contain entities representing the various model components. This enables complete mediation of administrative requests.

8.1.2 Existing Issues

The matter of administration for relationship-based access control is not solved at present. Whilst various models make some mention of administration (see Section 8.3.1), none of these is able to meet all of my requirements. Of the two most administratively-proficient relationship-based access control models, both Rizvi *et al.*'s implementation of ReBAC [157] and Stoller's RPPM^2 [174] satisfy three of my five requirements. Neither satisfy AR4 and AR5, and whilst they both do manage the addition and deletion of models elements (AR2) this is only with regard

to certain elements. There is, therefore, need for a more complete approach to administration in relationship-based access control.

AR1 Rizvi *et al.* use ReBAC's hybrid logic to determine whether administrative actions are enabled (between the primary participants) and applicable (between all participants) [157]. Similarly, Stoller's RPPM^2 model, which modifies an early version of RPPM [60, 62], controls administrative actions using the mechanisms of RPPM [174].

AR2 Rizvi *et al.* limit themselves to just adding and deleting edges. In contrast, RPPM^2 also uses specific actions for adding and deleting entities and rules, and for setting system-wide default decisions and conflict resolution strategies.

AR3 As both Rizvi *et al.* and Stoller make use of relationship-based access control models to evaluate their administrative request, they also implicitly support multiple administrators (as multiple entities may have the same relationships with other entities).

AR4 and AR5 Rizvi *et al.* only consider the administration of relationship edges and so fail to allow and control changes to other model components which are left, presumably, to some extra-model administrator. Similarly, whilst Stoller states that RPPM^2 "is comprehensive in the sense that it allows and controls changes to all aspects of the ReBAC policy", it fails to allow and control changes to RPPM's system model or actions which are also left, presumably, to some extra-model administrator.

8.2 Administrative Enhancements

8.2.1 Administrative Requests and Policies

Four minor request and policy changes are required in ARPPM to support administration of RPPM model instances using the model itself (as required by AR1):

- Firstly, there is a need for entities to be able to issue administrative requests in addition to the operational requests so far considered.
- Secondly, there is a need for principal-matching rules to be matched in light of these requests, and hence for the participants of the request to be identifiable in the same way as the subject and object of operational requests.

- Thirdly, there is a need for authorization rules to be applicable to administrative requests, but only if the rule is not limited to specific objects in the system graph.
- Fourthly, there is a need for a default decision function specific to administrative requests.

As indicated in the discussion of AR2, I choose to model all administrative requests through the addition and deletion of edges to the system graph. An entity is automatically added to the system graph upon the addition of its first incident edge, and is automatically deleted upon deletion of its last incident edge. I, therefore, extend RPPM_{1c}'s definition of a request to enable actions to target edges as well as entities.[2]

Definition 8.1. *An* administrative request *has the form* $(s, (v_1, t_1, v_2, t_2, r), a)$*, where* $s, v_1, v_2 \in V$*,* $t_1, t_2 \in T$*,* $r \in R$ *and* $a \in \{\mathsf{addEdge}, \mathsf{deleteEdge}\}$*. In order to be* well-formed *any administrative request requires* $\tau(v_1) = t_1$*,* $\tau(v_2) = t_2$ *and* $(t_1, t_2, r) \in E_{PR}$*. Additionally, a well-formed* addEdge *request requires* $V \cap \{v_1, v_2\} \neq \emptyset$ *and* $(v_1, v_2, r) \notin E$*, whilst a well-formed* deleteEdge *request requires* $(v_1, v_2, r) \in E$*.*

The effect of granting the well-formed request $(s, (v_1, t_1, v_2, t_2, r), \mathsf{addEdge})$ is to add (v_1, v_2, r) to E (having added either v_1 or v_2 to V, as necessary).[3] The effect of granting the well-formed request $(s, (v_1, t_1, v_2, t_2, r), \mathsf{deleteEdge})$ is to remove (v_1, v_2, r) from E; any vertex that is no longer connected to any other vertex as a result of the edge deletion will also be deleted. Henceforth, I assume all administrative requests are well-formed.

Remark 8.1. *Whilst it may be desirable, in some circumstances, to have an entity in the system graph which is not connected by a particular relationship* r *(for example, so that that relationship may be utilised in a forbidden target), I do not allow disconnected entities to remain in the system graph. Doing so would introduce a requirement for additional actions* addEntity *and* deleteEntity*; however, such actions are unnecessary as some connecting relationship* $r' \neq r$ *can always be present without interfering with the use of* r *in forbidden targets.*

Remark 8.2. *RPPM$_0$'s first step for deciding whether to grant an operational request* (s, o, a) *attempts to match principals to the request by evaluating principal-matching rules. The targets of each principal-matching rule are evaluated in respect*

[2] I explain why I extend RPPM_{1c} and not RPPM_0 in Remark 8.2.

[3] The types t_1 and t_2 are required in the request such that either entity v_1 or v_2 may be created in the system graph.

of paths between s and o in the system graph. This is insufficient when deciding whether to grant the administrative request $(s, (v_1, t_1, v_2, t_2, r), a)$ for two reasons. Firstly, I believe, as Stoller's healthcare example illustrates [174], that multiple paths of relationships may be relevant to the authorization of administrative requests. Crucially, there is no node in the system graph representing the object edge (v_1, t_1, v_2, t_2, r) and, even if such entities were introduced, logically one would not be present at the time a request to first add the edge is made. Evaluating paths between s and (v_1, t_1, v_2, t_2, r) is, therefore, impractical and insufficient.

I, therefore, base ARPPM on the $RPPM_{1c}$ model in order to make use of its path expression targets. In this way, administrative requests may match principals based on multiple paths of relationships between arbitrary entities in the system graph.

In order to enable the participants of an administrative request to be identified in the same way as for operational requests, I update RPPM_{1c}'s definitions of an untyped entity condition and its satisfaction (from Section 7.1.1).

Definition 8.2. *An* untyped entity condition *is either an entity variable x, an entity v in V, or an entity label from the set $RL = \{\textsf{subject}, \textsf{object}, \textsf{object-start}, \textsf{object-end}\}$.*

The set of entities that *satisfies* an untyped entity condition e with respect to a request q, denoted $[\![e, q]\!]$, is defined as

$$\begin{aligned}
\text{for any entity variable } x,\ [\![x, q]\!] &= V \\
\text{for any entity } v \text{ in } V,\ [\![v, q]\!] &= \{v\} \\
[\![\textsf{subject}, (s, o, a)]\!] = [\![\textsf{subject}, (s, (v_1, t_1, v_2, t_2, r), a)]\!] &= \{s\} \\
[\![\textsf{object}, (s, o, a)]\!] &= \{o\} \\
[\![\textsf{object-start}, (s, (v_1, t_1, v_2, t_2, r), a)]\!] &= \{v_1\} \\
[\![\textsf{object-end}, (s, (v_1, t_1, v_2, t_2, r), a)]\!] &= \{v_2\} \\
[\![\textsf{object}, (s, (v_1, t_1, v_2, t_2, r), a)]\!] &= \emptyset \\
[\![\textsf{object-start}, (s, o, a)]\!] = [\![\textsf{object-end}, (s, o, a)]\!] &= \emptyset.
\end{aligned}$$

Other than the change to untyped entity conditions, the principal-matching policy and its evaluation in ARPPM remains unchanged from RPPM_{1c}, and the processing of operational requests is unaffected.

In light of the fact that an administrative request does not identify a specific object entity within the system graph ($[\![\textsf{object}, (s, (v_1, t_1, v_2, t_2, r), a)]\!] = \emptyset$), I extend RPPM_{1c}'s definition of authorization rule applicability (Definition 4.10) to encompass administrative requests.[4]

[4]Recall from Section 7.2.1 that, unlike [64], I do not introduce path expression targets to authorization rules as

Definition 8.3. *An* authorization rule *has the form* (p, X, Y, b)*, where* $p \in P$*,* $X \subseteq T \cup V$*,* $Y \subseteq A$ *and* $b \in \{0, 1\}$*. Given a PMP* ρ*, an authorization rule* (p, X, Y, b) *is applicable to an administrative request* $q = (s, (v_1, t_1, v_2, t_2, r), a)$ *if all of the following conditions hold:*

- $p \in [\![\rho]\!]$*;*
- $X = V$ *or* $X = T$*;*[5] *and*
- $Y \cap \{a\} \neq \emptyset$*.*

Other than the introduction of authorization rule applicability in respect of administrative requests, the computing of the set of authorization decisions from a set of matched principals in ARPPM remains unchanged from RPPM_{1c}, and the processing of operational requests is unaffected.

Having determined the set of authorization decisions, I may determine whether a request is authorized in the same manner as RPPM_{1c}, employing default decisions and a conflict resolution strategy where appropriate. I believe that default decision-making associated with administrative requests should be distinct from that of operational requests. The default-per-object and default-per-type are certainly redundant as administrative actions are performed on edges rather than single entities. I, therefore, allow for specific administrative default decisions at the default-per-subject and system-wide default levels only.

Definition 8.4. *Given a system graph* $G = (V, E)$*, a set of types* T*, and an administrative request* $q = (s, (v_1, t_1, v_2, t_2, r), a)$*, a* default administrative decision function

$$\gamma_\alpha : V \cup \{\mathsf{sys}\} \to \{\perp, 0, 1\}$$

is a function which maps entities within the system graph to default decisions for administrative requests, where $\perp$ *is undefined,* 0 *is deny and* 1 *is allow.*[6] *The function maps default decisions on a per subject* $u \in V$ *and* sys*tem-wide basis. When initialized,* $\gamma_\alpha(\mathsf{sys}) = 0$ *and the function maps all other inputs to* $\perp$ *until they are configured.*

It should be clear that the four minor request and policy changes required to support administration do not significantly alter the process of (operational) request evaluation used by RPPM_{1c}. In the same way that a principal may be matched to multiple operational requests by distinct subjects (as long as the required target is

this reduces the rules' general expressiveness.

[5] Recall from Remark 4.6 that I may use $\star$ to indicate all entities.

[6] Although I do not allow $\gamma_\alpha(\mathsf{sys})$ to be set to $\perp$.

satisfied and the forbidden target is not satisfied), a principal may be matched to multiple administrative requests also. It is by virtue of RPPM's principal-matching process and the abstraction of authorizations away from subjects that multiple administrators are naturally supported by ARPPM as required by AR3.

Example 8.1. *Returning to my higher education example (introduced in Example 4.3), Figure 8.1a shows a pair of courses in dept. 1 without students. The policy*

$$\begin{aligned}\rho = \{\ldots, (\{&(\textit{subject}, user) \cdot \textit{Member-of} \cdot (dept\ 1, department),\\ &(dept\ 1, department) \cdot \textit{Runs} \cdot (\textit{object-end}, course),\\ &(\textit{subject}, user) \cdot \textit{Responsible-for} \cdot (\textit{object-end}, course)\},\\ &\text{none},\ \textit{course-admin}), \ldots\}\\ \varrho = \{\ldots, (&\textit{course-admin}, \star, \{\textit{addEdge}, \textit{deleteEdge}\}, 1), \ldots\}\end{aligned}$$

would enable the successful authorization of the administrative requests

$$q_1 = (professor\ 1, (student\ 1, user, course\ 1, course, \textit{Enrolled-on}), \textit{addEdge})\ \textit{and}$$
$$q_2 = (professor\ 2, (student\ 2, user, course\ 2, course, \textit{Enrolled-on}), \textit{addEdge}),$$

whereby two students are enrolled on the two courses by the administering professors, leading to Figure 8.1b. However, the administrative request

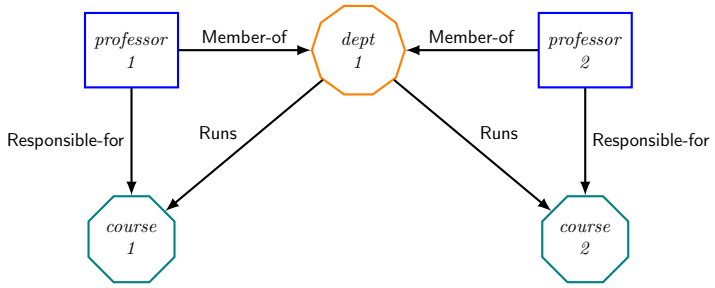
$$q_3 = (professor\ 1, (student\ 1, user, course\ 2, course, \textit{Ta-for}), \textit{addEdge})$$

would be denied as professor 1 does not have the relationships to have authority over course 2.

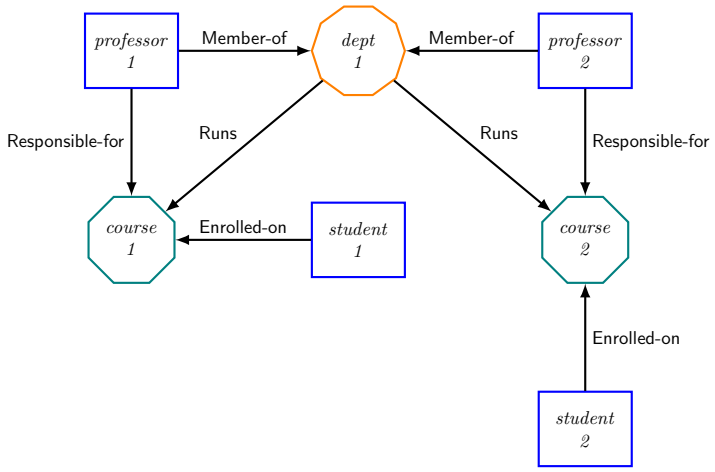
Remark 8.3. *It is worth noting that Example 8.1 clearly demonstrates how logical entities (in this case courses), and the relationships entities have with them, provide context to RPPM's request evaluation. A single principal-matching rule is, thereby, able to manage access based on this context such that multiple administrators are able to perform administrative actions whilst being confined to their appropriate scopes.*

8.2.2 Initial System Graph

The ARPPM model described so far is able to meet all but two of my user requirements. In order to exclude extra-model administration (AR4) and to control changes to all parts of the model (AR5) I must define an initial state for the system graph, populating it with entities reflecting model components. The minimum requirements for such an initial state involve an interplay of the entities, relation-



(a) System graph fragment



(b) After two requests

(c) Type legend

Figure 8.1: Administering a higher education system graph fragment

ship labels, system graph layout, and the model's policies. The requirements result naturally from (conceptually) adding entities, relationships and policy rules to an empty system graph up to such point that an initial administrative entity is able to subsequently make any and all changes that they may choose. I list these minimum requirements in Section 8.2.2.

In order to enable a discussion of how administration works in practice, I will first detail a recommended arrangement for the system graph initial state. This recommendation is a minimal viable arrangement which meets the requirements of Section 8.2.2.

Recommended Initial State

In this recommended arrangement I require all system graphs to be *initialized* as per Figure 8.2a. For ease of exposition, I omit from Figure 8.2a, and the following discussion and example, reversed edges (i.e., the existence of an edge $(T, root, \overline{r_\alpha}) \in E$ where the edge $(root, T, r_\alpha) \in E$), caching edges and audit edges. In addition, to allow a comprehensible diagram, I omit from Figure 8.2a edges from E_{PR_I} (I use the suffix I to indicate initialization) and also edges related to default decision entries mapped to $\perp$.

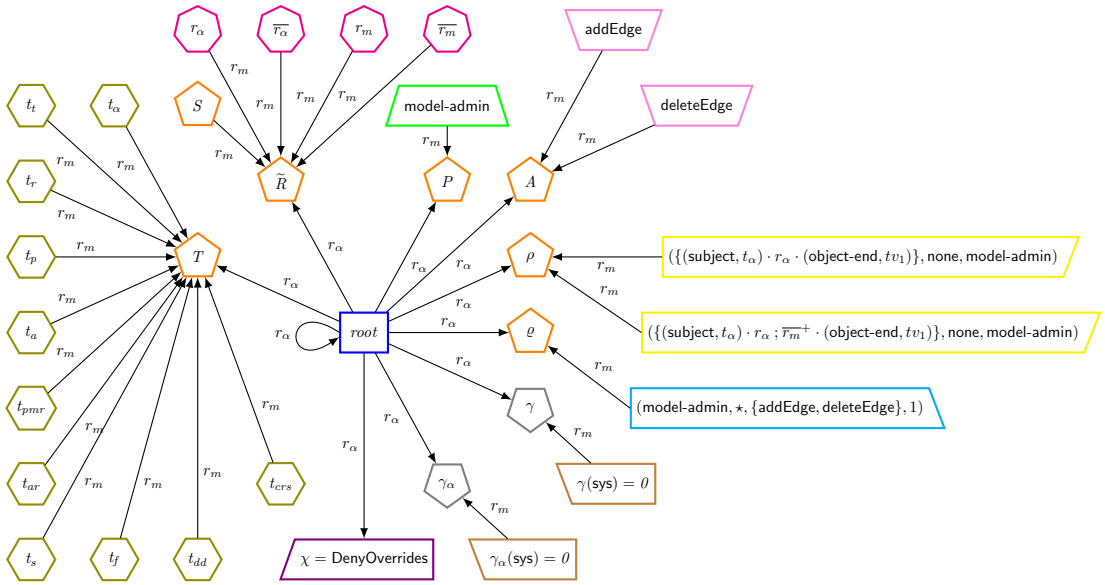
The initial underlying system model, principals, actions, principal-matching policy and extended authorization policy for this system graph are

$$
\begin{aligned}
T_I &= \{t_\alpha, t_t, t_r, t_p, t_a, t_{pmr}, t_{ar}, t_s, t_f, t_{dd}, t_{crs}\} \\
\widetilde{R}_I &= \{r_m, r_\alpha, \overline{r_m}, \overline{r_\alpha}\} \\
S_I &= \emptyset \\
E_{\mathrm{PR}_I} &= \{(t, t_s, r_m) : t \in T_I \setminus \{t_f, t_{dd}, t_{crs}, t_\alpha\}\} \cup \\
&\qquad \{(t_\alpha, t', r_\alpha) : t' \in T_I\} \cup \{(t_{dd}, t_f, r_m)\} \\
G_{\mathrm{PR}_I} &= (T_I, E_{\mathrm{PR}_I}) \\
P_I &= \{\mathsf{model\text{-}admin}\} \\
A_I &= \{\mathsf{addEdge}, \mathsf{deleteEdge}\} \\
\rho_I &= \{(\{(\mathsf{subject}, t_\alpha) \cdot r_\alpha \cdot (\mathsf{object\text{-}end}, tv_1)\}, \mathsf{none}, \mathsf{model\text{-}admin}), \\
&\qquad (\{(\mathsf{subject}, t_\alpha) \cdot r_\alpha \,;\, \overline{r_m}^{+} \cdot (\mathsf{object\text{-}end}, tv_1)\}, \mathsf{none}, \mathsf{model\text{-}admin})\} \\
(\varrho_I, \chi_I) &= (\{(\mathsf{model\text{-}admin}, \star, \{\mathsf{addEdge}, \mathsf{deleteEdge}\}, 1), \mathsf{DenyOverrides}).
\end{aligned}
$$

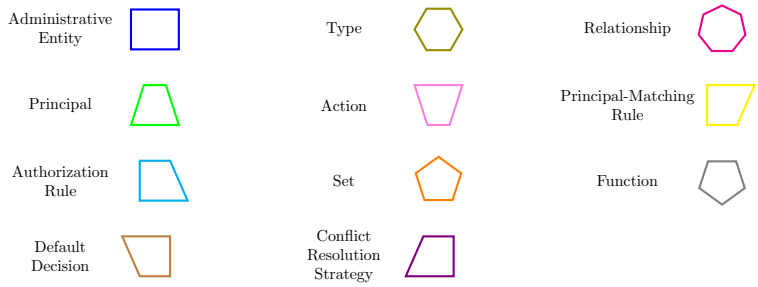
At its core, the initial system graph contains an administrative entity *root*, of type t_α[7], and entities representing the initial sets of types, relationship labels, principals, actions, principal-matching rules and authorization rules. The initial relationship entities are those necessary to construct this initial system graph: namely entities representing an Administrate relationship (which I will also refer to as r_α), a Member-of relationship (which I will also refer to as r_m), and their reversed counterparts, $\overline{r_\alpha}$ and $\overline{r_m}$ respectively. These have Member-of relationships with an entity $\widetilde{R}$ which represents the set of relationship labels. There are no initial symmetric relationship labels, although an entity S representing the set of symmetric relationship labels is present. The initial type entities represent the administrative entity's type t_α, the types type t_t, the relationships type t_r, the principals type t_p, the actions type t_a,

[7]This could be a human administrator of type *user*, or an autonomous administrator of type *auto*.

the principal-matching rules type t_{pmr}, the authorization rules type t_{ar}, the sets type t_s, the functions type t_f, the default decisions type t_{dd}, and the conflict resolution strategy type t_{crs}. These have Member-of relationships with T which represents the set of types.



(a) System graph fragment



(b) Type legend

Figure 8.2: Initial system graph (simplified)

There is a single initial principal entity representing a model-admin principal which has a Member-of relationship with the entity representing the set of principals P. The administrative actions addEdge and deleteEdge are represented by similarly named entities which have Member-of relationships with the entity representing the set of actions A. There are two initial principal-matching rules and an authorization rule which have Member-of relationships with the entities representing the policies, ρ and ϱ respectively. The initial principal-matching rules enable the model-admin principal to be matched to administrative requests made in respect of edges directed towards entities with which the subject has an Administrate (r_α) relationship, as

identified by the path expression

$$(\mathsf{subject}, t_\alpha) \cdot r_\alpha \cdot (\mathsf{object\text{-}end}, tv_1),$$

or edges directed towards entities with which the subject can satisfy the path expression

$$\{(\mathsf{subject}, t_\alpha) \cdot r_\alpha \,;\, \overline{r_m}^+ \cdot (\mathsf{object\text{-}end}, tv_1)\}.^8$$

The initial authorization rule grants the addEdge and deleteEdge actions to the model-admin principal without constraint.

In addition, the initial system-wide default decisions, $\gamma(\mathsf{sys}) = \mathit{0}$ and $\gamma_\alpha(\mathsf{sys}) = \mathit{0}$, have Member-of relationships with the entities representing the default decision function γ and the default administrative decision function γ_α, respectively. Finally, an entity representing the initial conflict resolution strategy, DenyOverrides is also present. *root* has an Administrate relationship with the conflict resolution strategy entity, the entities representing the default decision functions, all of the set entities previously described and with itself. From this initial system graph, changes are made through the addition and deletion of edges using administrative requests. (Recall that an entity is automatically added to the system graph upon the addition of its first incident edge, and is automatically deleted upon deletion of its last incident edge.)

In order to configure an ARPPM system graph for an actual system the *root* entity would add entities, by adding edges, to the newly initialized system graph (as I will illustrate in Example 8.2). In so doing, *root* would be able to modify the underlying system model to support the desired entity types and relationships, and could then introduce concrete and logical system entities into the graph by adding edges such as $(root, u, r_\alpha)$. How the structure of the system graph develops will be up to the *root* entity and, subsequently, to any entities granted the ability to administer the system graph through the existence (or otherwise) of relationships satisfying the requisite paths. Such design choices are likely to be instance, if not system, specific and are not constrained by ARPPM.

To prevent an ARPPM model from becoming unusable I do not allow any of the initial system graph's edges to be deleted except those involving entities of the types t_{dd} and t_{crs} (which may need to be deleted in order to change the system-wide default decisions or conflict resolution strategy). All other changes to the system graph are managed by the mechanisms of ARPPM. Whilst changes to the system

[8] Whilst these initial principal-matching rules only apply to requests directed towards one of the initial entities, any object edge directed away may be converted to an object edge directed towards by simply switching the entities and replacing the path condition π with $\overline{\pi}$.

model $(T, R, S, G_{\mathrm{PR}})$, where $G_{\mathrm{PR}} = (T, E_{\mathrm{PR}})$, are triggered through the system graph, I require that once granted they are enacted by an authorized and correct "system administrator".[9]

I require an ARPPM implementation's PDP to enforce the following restrictions to changes to the initial system graph:

1. Changes to $t \in T$, $s \in S$ or $r \in \widetilde{R} \setminus S$ are triggered, respectively, by administrative requests of the form $(u, (t, t_t, T, t_s, r_m), a')$, $(u, (s, t_r, S, t_s, r_m), a')$, and $(u, (r, t_r, \widetilde{R}, t_s, r_m), a')$ where u is the administrative entity (initially *root*) and $a' \in \{\mathsf{addEdge}, \mathsf{deleteEdge}\}$. Changes to E_{PR} are triggered by administrative requests adding or deleting edges between the entities representing types in G. In each case the authorized administrative request is only considered complete once the system administrator has updated the system model to reflect any desired changes.

2. $\{(t_\alpha, t, r_\alpha) : t \in T\} \subset E_{\mathrm{PR}}$.

3. $E_{\mathrm{PR}} \cap \{(t, t'', r) : t \in T, t'' \in \{t_s, t_f\}, r \notin \{r_m, r_\alpha\}\} = \emptyset$.

4. Given $T'' = \{t_t, t_r, t_p, t_a, t_{pmr}, t_{ar}, t_{dd}\}$,
   $E_{\mathrm{PR}} \cap \{(t, t'', r) : t \in T, t'' \in T'', r \neq r_\alpha\} = \emptyset$.

5. Given $T' = \{t_s, t_t, t_r, t_p, t_a, t_{pmr}, t_{ar}, t_{dd}\}$,
   $E_{\mathrm{PR}} \cap \{(t', t'', r) : t' \in T', t'' \neq t_s, r \neq r_m\} = \emptyset$.

The last three restrictions require that:

3. Edges connecting to entities of the sets type t_s or functions type t_f must be either Member-of (r_m) or Administrate (r_α) relationships;

4. Edges connecting to entities of the types found in the set T'' must be Administrate (r_α) relationships; and

5. Edges connecting from entities of the types found in the set T' can only connect to the sets type t_s and must do so using a Member-of (r_m) relationship.

Example 8.2. *Having initialized a new ARPPM model instance, as per my recommended arrangement, it is possible to construct Example 8.1's system graph fragment*

[9] Note that changes to the system model are authorized within ARPPM in the same way that all other administrative requests are. It is only the resulting manipulation of the elements which must occur outside of the system graph to which they relate.

(as shown in Figure 8.1a) within that new instance by issuing: the sequence of requests

$$q_1 = (root, (course, t_t, T, t_s, r_m), \textsf{addEdge})$$
$$q_2 = (root, (department, t_t, T, t_s, r_m), \textsf{addEdge})$$
$$q_3 = (root, (user, t_t, T, t_s, r_m), \textsf{addEdge})$$

to create the necessary entity types; the sequence of requests

$$q_4 = (root, (Member\text{-}of, t_r, \widetilde{R}, t_s, r_m), \textsf{addEdge})$$
$$q_5 = (root, (Responsible\text{-}for, t_r, \widetilde{R}, t_s, r_m), \textsf{addEdge})$$
$$q_6 = (root, (Runs, t_r, \widetilde{R}, t_s, r_m), \textsf{addEdge})$$

to create the necessary relationship labels; the sequence of requests

$$q_7 = (root, (department, t_t, course, t_t, \textsf{Runs}), \textsf{addEdge})$$
$$q_8 = (root, (user, t_t, course, t_t, \textsf{Responsible-for}), \textsf{addEdge})$$
$$q_9 = (root, (user, t_t, department, t_t, \textsf{Member-of}), \textsf{addEdge})$$

to create the necessary permissible edges; the sequence of requests

$$q_{10} = (root, (course\ 1, course, root, t_\alpha, \overline{r_\alpha}), \textsf{addEdge})$$
$$q_{11} = (root, (course\ 2, course, root, t_\alpha, \overline{r_\alpha}), \textsf{addEdge})$$
$$q_{12} = (root, (dept\ 1, department, root, t_\alpha, \overline{r_\alpha}), \textsf{addEdge})$$
$$q_{13} = (root, (professor\ 1, user, root, t_\alpha, \overline{r_\alpha}), \textsf{addEdge})$$
$$q_{14} = (root, (professor\ 2, user, root, t_\alpha, \overline{r_\alpha}), \textsf{addEdge})$$

to create the necessary entities; and the sequence of requests

$$q_{15} = (root, (professor\ 1, user, course\ 1, course, \textsf{Responsible-for}), \textsf{addEdge})$$
$$q_{16} = (root, (professor\ 1, user, dept\ 1, department, \textsf{Member-of}), \textsf{addEdge})$$
$$q_{17} = (root, (dept\ 1, department, course\ 1, course, \textsf{Runs}), \textsf{addEdge})$$
$$q_{18} = (root, (professor\ 2, user, course\ 2, course, \textsf{Responsible-for}), \textsf{addEdge})$$
$$q_{19} = (root, (professor\ 2, user, dept\ 1, department, \textsf{Member-of}), \textsf{addEdge})$$
$$q_{20} = (root, (dept\ 1, department, course\ 2, course, \textsf{Runs}), \textsf{addEdge})$$

to create the necessary relationships between those entities.

The policies associated with Example 8.1 may be created using a similar process. For conciseness, I will refer to the principal-matching rule used in the example as PMR_1; and so

$$\begin{aligned} PMR_1 = (\{&(\textsf{subject}, user) \cdot \textsf{Member-of} \cdot (dept\ 1, department), \\ &(dept\ 1, department) \cdot \textsf{Runs} \cdot (\textsf{object-end}, course), \\ &(\textsf{subject}, user) \cdot \textsf{Responsible-for} \cdot (\textsf{object-end}, course)\}, \\ &\textsf{none},\ \textit{course-admin}). \end{aligned}$$

Then the policies may be created using the requests

$$q_{21} = (root, (course\text{-}admin, t_p, P, t_s, r_m), \textsf{addEdge})$$
$$q_{22} = (root, (PMR_1, t_{pmr}, \rho, t_s, r_m), \textsf{addEdge})$$
$$q_{23} = (root, ((\textsf{course-admin}, \star, \{\textsf{addEdge}, \textsf{deleteEdge}\}, 1), t_{ar}, \varrho, t_s, r_m), \textsf{addEdge}).$$

This ARPPM model instance is now configured and ready for the requests to add students, as discussed in Example 8.1.

Generalised Initial State

Whilst I have outlined a specific recommended initial state above, my specific arrangement is just one of the infinite viable initial states. A system graph arrangement is a viable initial state if it contains entities representing all of the model components, and it contains appropriate relationships (reflected amongst those entities) to enable the policies (also reflected amongst those entities) to be used to authorize administrative requests by some initial administrative entity to add incident edges to any of those entities. Specifically, it must contain:[10]

1. An initial administrative entity to submit requests in the initialized model.
2. Entities representing each of the model's element containers (i.e., sets and functions): types; relationships; principals; actions; principal-matching rules; authorization rules; and default decision functions.
3. An entity representing the initial value for the conflict resolution strategy.
4. Paths of (one or more) relationships connecting the initial administrative entity and the entities from items 2 and 3.

[10] I work on the assumption that the initial state should not rely on default decisions to authorize administrative requests as such a model is likely inherently prone to entities being overprivileged until it is further "locked down".

5. Entities representing at least one principal-matching rule to match at least one principal to those paths of relationships.
6. Entities representing at least one authorization rule to authorize administrative actions to those matched principals.
7. Paths of (one or more) relationships connecting the entities from items 5 and 6 to the entities representing their containers in item 2.
8. Entities representing the initial values for the default decision functions.
9. Entities representing each of the types, relationships, principals, and actions necessary to create the elements from items 1 through 8.
10. Relationships representing the permissible relationship edges between the types added in 9.
11. Paths of (one or more) relationships connecting the entities from items 8 and 9 to the entities representing their containers in item 2.
12. Entities representing at least one principal-matching rule to match at least one principal to the paths of relationships from the initial administrative entity, via the entities representing the model's element containers (from item 2), to the entities in items 8 and 9.
13. Entities representing at least one authorization rule to authorize administrative actions to those matched principals.
14. Paths of (one or more) relationships connecting the entities from items 12 and 13 to the entities representing their containers in item 2.

Whilst there are infinite possible such arrangements (there are infinite paths of relationships which could be employed), in reality it is desirable to limit the size and complexity of the initial state so as to avoid negatively impacting the performance of request evaluation. Hence, my recommended initial state uses direct Administrates relationships wherever possible, and thereby requires only two principal-matching rules and a single authorization rule to function.[11]

As well as excluding extra-model administration (AR4) and enabling the control of changes to all parts of the model (AR5), the introduction of an initial system graph also has a direct impact on the "safety" of the ARPPM model [96]. The initial

[11] One could argue that rather than having types for each individual model element, a single model entity type t_m could have been employed to simplify the recommended initial state. I chose to utilise distinct types in order to provide a more robust starting point which could benefit from specific constraints for the permissible edges and the restrictions required to ensure the model remains usable.

system graph is arranged such that the initial administrator is able to perform the initial addEdge and deleteEdge actions on any entity, and may subsequently introduce new entities for which it will have the same control. The initial administrator is also able to introduce relationships which will enable any of these new entities to have similar or equivalent privileges within the system. Therefore, the safety of ARPPM is trivially decidable. The same cannot be said without the initial system graph (i.e. in $RPPM_0$, $RPPM_{1a}$, $RPPM_{1b}$ or $RPPM_{1c}$) where the decidability of safety depends on the specific configuration of the system model, system graph and policies at the time. In the general case of large, complex arrangements of entities and lengthy policies safety will be undecidable.

8.3 Model Comparisons

8.3.1 Existing Administrative Models

Given the importance of administration to access control models, it is a topic commonly covered in the literature as part of model designs. However, the level of detail at which administration is considered varies greatly. At the two extremes, some models only make limited mention of administration [76, 77, 79, 105, 144] (although they may indicate its importance), whilst others are specifically tailored to defining its functionality [35, 58, 64, 124, 140, 160, 163, 174]. In between, are models with a range of partial administrative capabilities of one form or other [43, 51, 103, 108, 110, 125, 157].

Role-Based Access Control

RBAC Whilst the early RBAC papers by Ferraiolo *et al.* advocate the administrative benefits of RBAC, they do not articulate the mechanisms through which administration may be performed [77, 79]. Instead they simply indicate that a *security administrator* is required to perform tasks such as assigning users to roles. In contrast, the specific administrative functions required of an RBAC implementation are listed in detail in the INCITS ANSI standard (along with the constraints that apply to these functions) [108]. However, no indication is given of the process for authorizing the use of those administrative functions.

ARBAC97 The popularity of role-based access control has led to a wide range of discussions about its administration. The most significant work on administration of roles is Sandhu *et al.*'s ARBAC97 model [163], as introduced in Section 2.2.2.

This administration model employs three component models to manage role-role assignment (RRA97), permission-role assignment (PRA97) and user-role assignment (URA97). The concepts of ARBAC97 have been layered on top off, but grounded in the functionality of, RBAC, thus making it a prime example of using an access control model to administer itself.

RBAC/Web Not long before ARBAC97, NIST created an implementation of RBAC for the web (RBAC/Web) and an Admin Tool was produced as part of this implementation [76]. This reference implementation was specifically designed for corporate intranet use and clearly explains how RBAC may be of benefit in such environments. Whilst this implementation highlights the benefits of RBAC to administrators in particular, it provides minimal information about the processes by which these benefits are realised. Further, Ferraiolo *et al.* direct the reader to ARBAC97 for a "richer, more flexible, model", and the URA97 model was used to extend NIST's implementation of RBAC/Web [163].

SARBAC The scoped administration of role-based access control (SARBAC) model was designed as a more robust and flexible approach to the management of RBAC than ARBAC97 [58]. Specifically, SARBAC is constructed from three components, in a manner similar to ARBAC97; however, the role hierarchy component is the foundation of SARBAC whilst it is the capstone of ARBAC97. Crampton and Loizou highlight that some components of an access control model are *static components* whist others are *dynamic*. They then identify a *complete* access control model as one in which a state transition may be defined to track all of the dynamic components from one change to the next. SARBAC is a complete model which uses partially ordered sets to define *administrative scopes*, which, in turn, identify the roles which may be managed by an administrator. As administrative scopes are dynamic, changes to the role hierarchy amend the scope within which an administrator may operate. This offers SARBAC a flexibility not present in ARBAC97, and, due to this, hierarchy changes which had to be blocked in ARBAC97 are supported in SARBAC.

X-GTRBAC Admin In order to support policy administration for multi-domain environments, Bhatti *et al.* extend various existing works to define X-GTRBAC Admin [34, 35, 113, 169]. Within this model they define Admin Domains which are partially ordered into a domain hierarchy for an enterprise. Each Admin Domain has an Admin Role which is authorized to manage policy in that domain and subordinate

domains. Membership of Admin Roles is managed by the *system security officer (SSO)* [33].[12]

UARBAC A further administrative RBAC model is the UARBAC model of Li and Mao [124]. Before describing their model they clearly articulate a set of six design requirements (motivated by scalability and flexibility, "psychological acceptability", and "economy of mechanism"):[13]

1. Support decentralized administration and scale to large RBAC systems.
2. Be policy neutral in defining administrative domains.
3. Apparently equivalent sequences of operations should have the same effect.
4. Support reversibility.[14]
5. Predictability.
6. Using RBAC to administer RBAC.

It seems that reversibility is the key driver of UARBAC's distinct features. UARBAC adds to RBAC the concept of *object classes* (equivalent to RPPM's entity types) with user and role classes predefined. UARBAC then splits the granting of permissions into two: an administrator must have the "admin" access mode on an object to give out permissions about it, and must have the "empower" access mode on an object to give permissions about another object to it. Further, UARBAC encompasses revocation into the granting permissions – thus ensuring an administrator can always reverse their own actions. Together these changes maximise reversibility by enabling revocation to be performed either from the perspective of the object targeted by a permission or that of the object granted a permission. Li and Mao also alter the role hierarchy of RBAC to support reversibility, making it irreflexive, acyclic and explicit. This ensures that no changes made to the hierarchy can have side-effects, and so all changes are efficient to reverse through use of the appropriate counter-operation.

Attribute-Based Access Control

XACML The core XACML standard does not detail the processes of administration, instead that information is separated into an associated Administration and

[12]The SSO seems to be a super-administrator constrained by the model yet not clearly identified. I hypothesise that the SSO is the initial member of the Admin Role within the first Admin Domain in a hierarchy.

[13]I used a similar approach when describing the motivation for ARPPM's design in Section 8.1.1.

[14]Where Li and Mao consider two aspects: the need for operations to delete objects where they may be added; and the need for the system state to be returned to the recent past state if an object is added and then deleted.

Delegation Profile specification [110, 140, 141].[15] This profile specification works on the basis that policies may be either administrative or access, where administrative policies delegate the right to issue access to others. Policies have *issuers* and the authority of an issuer must be verified back to a *trusted policy* before the issuer's (initially un-trusted) policies may be considered.[16] Specifically, a *reduction* process is employed to search a graph of the policies within a policy set; the goal being to determine which of those policies should be considered. Ordered pairs of policies (within the initially disconnected graph) are evaluated in respect of an access request to determine if the delegating policy authorizes the delegate for the requested *situation* (i.e., whether the issuer of the delegating policy has delegated to the other the relevant permissions, whether access or administrative). The results of these evaluations may create directed edges between the adjacent policies, where edges are labelled with the (non-deny) result determined by the source policy. Once the reduction graph is produced, a chain of result labels between an issuer's policy and a trusted policy determines whether the issuer's policy will be considered as part of evaluating the request.

NGAC As indicated in Section 2.3.4, NGAC supports administrative requests which are evaluated by the PDP and then enacted by the PAP [110]. The PAP performs the requested administrative operation by executing primitive administrative commands against the PIP where the policy-related data are stored. These commands include creating and deleting basic elements and containers (such as users, objects, attributes, policy classes and access rights), as well as creating and deleting relations and obligations. The policies which govern administration may be centralized or decentralized but begin with a single administrator [78].

UCON Whilst Park and Sandhu only briefly mention administration when presenting UCON (particularly to distinguish *immutable attributes* which may only be changed by administrative action compared with *mutable attributes* which may be changed by access), subsequent work by Salim *et al.* has focused on UCON's administration [144, 160]. They introduce a two layer structure, comprising a *peer model* and an *authoriser model*. Subjects then may assert beliefs about other subjects and objects in the peer model; however, initially those assertions are not trusted and so not considered during request evaluations. An *authority root* must identify, in the authoriser model, the assertions that they wish to be considered. Control of

[15] The first committee standard was published in October 2010 [140], and the latest, fourth, committee draft was published for public review in November 2014 [142].

[16] Trusted policies by their nature do not have an issuer specified and do not need verifying.

such acceptance is managed by a system-wide administrative policy which is defined outside of Salim *et al.*'s structure. This policy may be explicit, identifying a specific authority root, or may be algorithmic, determining subjects able to accept assertions based on attributes (and expressions on attributes) the subject and object may hold.

Relationship-Based Access Control

OSNs Models applied to social networking have focused on how users specify policies controlling access to existing resources. Who is authorized to define such policies is determined in one of three ways:[17]

(i) The existence of an "ownership" relation [44, 51];

(ii) The existence of more senior users within the system [51, 103]; or

(iii) The existence of an extra-model administrator (of some kind) [43].

Cheng *et al.* make use of a *controlling user* to define policies for entities under their control [51]. Whilst they use an ownership relation as an example indicator of this control, alternative relationships are acceptable. Hu and Ahn allow for a variable number of controllers (owner, contributor, stakeholders and disseminator) contributing to the policy [103]. Whilst these controllers are identified with implementation specific constructs (based on a "space" within a social network) it is easy to imagine that they could be determined by the existence of relationships with a shared data item entity. Carminati *et al.* introduced *admin policies* as one of three policy types within their semantic web model for social network access control [43]. The admin policies are determined by the *Security Administrator* of a social network. A policy of the form $antecedent \Rightarrow consequent$ is based on relationships between a user and a resource such that if the combination of properties and relationships identified in the *antecedent* are true then the *consequent* results. The *consequent* indicates the actions the user can grant or prohibit on that resource.

ReBAC The original work by Fong and Siahaan, later extended by Bruns *et al.*, on the ReBAC model does not consider administration of the logic-based policies they define [41, 85]. Whilst Fong's ReBAC model includes state transitions there is no discussion of how these are triggered or authorized [82]. However, more recent practical work on ReBAC by Rizvi *et al.* does consider administration [157]. In their healthcare implementation, Rizvi *et al.* identify specific administrative actions which

[17] As these models do not abstract authorization policy in the manner that RPPM does, the policies in question are approximately equivalent to a combination of RPPM's principal-matching policy and authorization policy.

are associated with two types of precondition, *enabling* and *applicability*, and have *effects* which add or delete edges. Their enabling and applicability preconditions are equivalent to path conditions which must be satisfied between the "participants" of the action (including the subject and object). Whilst Rizvi *et al.* do not refer to state transitions, their modifications to the protection state fulfil single relation variants of Fong's PUSH and POP transitions [82].

RPPM Stoller's RPPM^2 model is based on an early version of RPPM with a focus on the administration of the system graph and major policy components [60, 174]. RPPM^2 provides the ability to add and delete edges, entities, and authorization rules (which are a combination of RPPM's principal-matching policy and authorization policy), as well as the ability to set defaults and a conflict resolution strategy.

AReBAC Inspired by requirements for multi-tenancy in cloud computing environments, Cheng *et al.* combine Rizvi *et al.*'s applicability precondition with the policy language of Stoller's RPPM^2 to define the AReBAC$_1$ model [49]. They further extend that model to include: cascading revocation (AReBAC$_2$); and support for provenance (AReBAC$_3$). Cascading revocation requires that dependent edges are removed when the edge they depend on is removed; this allows the removal of tenant trust relationships to trigger removal of edges previously authorized using that trust. Support for provenance information then enables these dependencies to be tracked (using typed edges) in more complex multi-tenancy arrangements.

8.3.2 Administrative Enhancements

The ARPPM model introduces administrative enhancements to the RPPM$_{1c}$ model, enabling an RPPM model instance to be administered by the model itself.

Administrative Requests and Policies Whilst Carminati *et al.* use relationships between a user and a resource to determine the actions the user can grant or prohibit on that resource, they do not manage the creation and deletion of resources, and they are, thus, limited to actions performed on resources that already exist [43]. Cheng *et al.* use relations to indicate control over objects, and thereby the permissions for users to specify administrative policy; however, they give no indication as to how policy specification is authorized to these controlling users [51]. Further, no consideration is given as to how the necessary relationships are initially acquired. In contrast, Stoller includes actions which enable edges, rules, default decisions and conflict resolution strategies to be defined within his RPPM^2 model [174]. However,

Stoller introduces a complex "strictness order" on authorization rules in order to prevent undesirable ramifications of changes to his authorization policy (a combination of RPPM's principal-matching policy and authorization policy). Whilst I agree that it should not be possible to circumvent the access control model, I believe that a far simpler approach is to explicitly deny undesirable actions by subjects. I assume that the *root* entity is trustworthy and correct, and I believe that such restrictions can be implemented by *root* through the use of denial authorizations within the ARPPM authorization policy. In their healthcare implementation of ReBAC, Rizvi *et al.* [157] identify specific administrative actions to add or delete edges based on relationship-derived enabling and applicability preconditions. Whilst this construction is not dissimilar to mine, ARPPM is more generic still as it does not bind graph modifications to implementation specific actions. It should be clear from my discussion of ARPPM's functionality that it is a *complete* model as defined by Crampton and Loizou [58]. The state of the model is represented by the values of the system model, system graph, policies and policy elements. Each transition involves either the addition or deletion of a system graph edge, which, in turn, may modify the system model, policies and policy elements in a deterministic manner.

Initial System Graph No other relationship-based access control model has so completely and formally defined administrative features as ARPPM. As part of this definition, ARPPM requires a grounding upon which the administrative model and process may assuredly operate. Whilst the initial system graph is that grounding in ARPPM, such a construct is not seen in other models and each has some aspect of their finer workings skipped over as a result.

8.4 Summary

I have introduced four minor request and policy changes to RPPM_{1c} and have defined an initial system graph state that enables the resulting ARPPM model to be used to administer a model instance fully. I have shown how a newly initialized model instance may be used as the grounding for a specific system graph, where that system graph is constructed in the model instance by administrative requests made by the initial administrative entity. Now that RPPM may be implemented and operated in a controlled, secure manner I shall describe (in the next chapter) an enhancement to enable the authorization of inter-domain access control requests.

Chapter 9

Inter-RPPM

Whilst administration (as discussed in Chapter 8) is, probably, the most commonly discussed aspect of access control model application, the ability of models to support multi-domain environments is another prominent topic. All access control models have some scope of enforcement which identifies the *security domain* over which an instance has control. The entities and policies associated with a particular instance are used to enforce the will of the domain's authorities, be that a single security controller or some subset of the contained (human or autonomous) entities. However, that will (and those policies) does not have effect outside of the bounds of the instance of the model (i.e., on other entities) and has no knowledge of external parties also. Therefore, the protection system may only protect the system within which it operates.

In this chapter I present Inter-RPPM, a model based on RPPM_{1a} and containing a number of *inter-operation enhancements* to support the authorization of inter-domain access control requests:

- Bridged system group – this enhancement enables system graphs to be connected together via *bridges* between *hub* entities.
- Inter-operation request evaluation – this enhancement introduces two new request types to enable system graphs to evaluate remote requests, whether they originate in that system graph or not. A minor extension to the targets of principal-matching rules enables principals matched in one system graph to activate principals in another.

Much of the functionality of the Inter-RPPM model has previously been published, in conjunction with Jason Crampton, in [65].

9.1 Motivation

Originally, electronic data sharing began within individual systems and so access control models focused on managing authorization within a single, local security domain. As computer systems became more interconnected there became a greater need to accommodate the processing of requests made by remote subjects. Such a multi-domain environment is becoming more commonplace (most recently in wireless sensor networks (WSN) and Internet of Things (IoT), see Section 10.1 and Chapter 11) and so models have been tailored to support this functionality. However, inter-operation has not been considered in relationship-based access control prior to RPPM. I believe that inter-operation is an important area for investigation for two key reasons.

Firstly, systems are frequently connected together in order to support a wider range of services. In such cases, a subject in one system may request to perform an action on an object in a remote system. Currently such requests cannot be supported in relationship-based access control models unless a single *super-graph*[1] model captures every entity and relationship within the two, and all intermediary, systems. This requires global policies, outside of the "authority" of any one component system, putting at risk the autonomy of all. The interconnection of discrete autonomous subgraphs is, therefore, desirable.

Secondly, relationship-based access control models containing very large system graphs (stand-alone, or super-graph as just discussed) will be impacted by the fact that request evaluation in such models has a time complexity linked to the number of nodes in the graph, whether those nodes are relevant to the request being evaluated or not. It is, therefore, desirable to decompose such very large system graphs into smaller discrete autonomous subgraphs. Whilst request evaluation complexity in each of these subgraphs will still be linked to the number of nodes, the practical overall complexity is expected to be greatly reduced as many requests will only involve a subset of subgraphs (assuming appropriate routing is available). Therefore, the distributed evaluation of the request both reduces the load for any one subgraph and reduces the percentage of the total entity space which must be considered when matching paths to principals.

I am, therefore, motivated to develop a framework by which two system graphs may be connected and requests initiated in one may be authorized in the other. I believe RPPM is naturally suited to this task as it supports the modelling of general computing environments, to which inter-operation is highly relevant. Further, its

[1] I use the term super-graph to identify a large system graph which models multiple systems which might otherwise be modelled by distinct stand-alone system graphs.

two step request evaluation process and use of security principals provide a natural "break point" at which a request may be transferred from one system to another (particularly when employing graph-based policies). RPPM provides a convenient basis for combining paths from multiple graphs without searching end-to-end across the, potentially numerous, inter-connected system graphs.

It is important to note the distinction between multi-domain environments and those that are simply "decentralized". The term "decentralized" is commonly used to refer to any (general) environment where administration tasks are distributed amongst multiple administrators [45, 58, 124, 163]. Obviously a multi-domain environment meets that definition by virtue of there being distinct administrators managing distinct policies in distinct domains. In the more general perspective, whilst the decentralization of administration sets up administrative scopes for individual administrators, this does not necessarily imply separate security domains as I have described them. Specifically, whilst administrative scopes for individual administrators may be distinct, the underlying model elements may be consistent across those administrators. Whilst administrators may not individually be able to act on all entities within an instance, if all requests are evaluated within the context of the instance then that is not a multi-domain environment.

9.2 Inter-Operation Enhancements

9.2.1 Bridged System Group

The goal of my inter-operation framework is to connect distinct and autonomous system graphs in such a way that I am not required to define and evaluate relationship paths traversing a single super-graph. I, therefore, introduce an inter-operation framework which maintains the autonomy of individual system graphs by preserving their individual distinct request evaluation scopes, system models and policies. This way, the PDP of each individual system graph is able to evaluate "local" requests as usual, but may also contribute to the evaluation of remote requests crossing multiple system graphs.

In order to provide connectivity between system graphs, a construct external to the system graphs is required. I, therefore, introduce the concept of a *bridged system group* and employ bridging relationships[2], called *bridges*, between *hub* entities within distinct component system graphs. A request to access a remote resource

[2]I require an extra-model administrator to be responsible for the management of these bridging relationships as no single intra-model administrator has authority. However, I can also envisage an approach whereby the bridging relationships are "negotiated" by the authorities of connected system graphs (although I do not consider the mechanisms by which such an approach may be fulfilled.

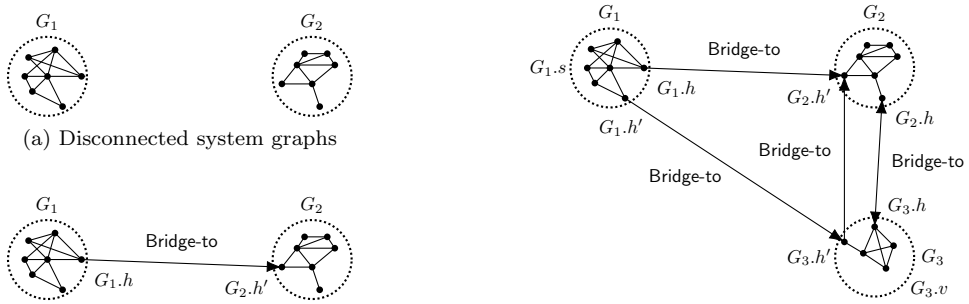
will need to be evaluated by a sequence of RPPM systems. Bridges provide the link through which I propagate information about the outcome of each system's local request evaluation to the next system in the sequence, where it subsequently informs another local evaluation.

I employ a hierarchical dot notation to label elements: that is $G.v$ and $G'.v$ represent different nodes (with the same label v) when G and G' are distinct system graphs.[3] When all of the elements of a tuple, such as (v, v', r), belong to the same system graph G, I will write $G.(v, v', r)$ to simplify notation.

Definition 9.1. *Let $\mathcal{G} = \{G_1 = (V_1, E_1), \ldots, G_n = (V_n, E_n)\}$ be a set of system graphs. Then, a* bridge *is an edge of the form $(G_i.u, G_j.v, \textsf{Bridge-to})$ such that $G_i \neq G_j$. A* bridged system group *is a pair $(\mathcal{G}, \beta)$, where β is a non-empty set of bridges and for all $(G_i.u, G_j.v, \textsf{Bridge-to}) \in \beta$, $G_i, G_j \in \mathcal{G}$.*

Informally, a bridged system group comprises two or more RPPM model instances whose system graphs are connected by one or more bridges. Each bridge connects two hub entities, one each from two distinct component system graphs.

Example 9.1. *I illustrate this framework with a simple example which I construct from two, initially disconnected, model instances, identified by their system graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, shown schematically in Figure 9.1a.*



(a) Disconnected system graphs

(b) Bridged system group (1,2)

(c) Bridged system group (1,2,3)

Figure 9.1: Constructing a bridged system group

In Figure 9.1b I illustrate the inter-connection of system graphs G_1 and G_2 through the bridge $(G_1.h, G_2.h', \textsf{Bridge-to})$; I develop the example further in Figure 9.1c to incorporate three system graphs inter-connected via five bridges.[4]

Bridges are directed – $(G_2.h, G_3.h, \textsf{Bridge-to})$ connects $G_2.h$ to $G_3.h$, but not vice versa – and are represented by an arrow. However, there may also exist a bridge

[3] Where there is no ambiguity through context or element naming I continue to leave out the prefix for convenience and clarity. So I do not prefix V_1 and E_1 in $G_1 = (V_1, E_1)$, for example.

[4] Note that the example of Figure 9.1c could equally represent three distinct system graphs which have been connected together, or a large stand-alone system graph which has been decomposed into three subsystems.

$(G_3.h, G_2.h, \textsf{Bridge-to})$ *in which case I will use a double-headed arrow to represent the pair of bridges between* $G_2.h$ *and* $G_3.h$*.*

Remark 9.1. *It is important to note that I place no specific requirements on the system models of system graphs which participate in a bridged system group. Two system graphs may be connected by a bridge, yet may have wholly different sets of entity types and relationship labels. The entities in those system graphs may, thereby, represent very different systems whose local policies are incomparable. For example, one system graph may represent a higher education department, much like that introduced in Example 4.3, whilst another may represent an IoT test lab used by faculty and students of that department.*

Any bridged system group defines inter-system paths, obtained by traversing the bridges. A *system graph sequence* defines the sequence of system graphs along such a path, where no system graph may be repeated and the directionality of the bridging relationships constrain the sequence. System graph sequences are a key component of the request evaluation process as they identify a path from the system originating a request to the system graph containing the target object. Multiple such paths may exist; I require that the PDP within a system be able to determine the single, least cost path.

To achieve this I require that an extra-model administrator[5] assign costs to bridges, and that hub entities maintain and exchange system path information using a modified path-vector routing protocol. Each hub communicates with adjacent hubs to which it directs a bridging relationship, retrieving the cost of their total (least) cost path to every other hub, and therefore every other system graph, in the bridged system group.[6] (I use an infinite cost to indicate an unreachable hub.) Upon receiving these path costs the hub adds to each the cost of the bridge connecting it to that neighbour; it will then update its local system path table to reflect the least cost path (if it didn't already know it) to every hub, along with which adjacent (or local) hub each least cost path passes through. A system's PDP can collate this information from each of its local hub entities in order to determine the single, least cost path to a target system graph, or to determine that no such path exists. Henceforth, when identifying a system graph sequence between two system graphs I assume the least cost such sequence.

Example 9.2. *Returning to my example bridged system group, let us consider the least cost paths which result when the bridge costs are as per Figure 9.2a. For ease*

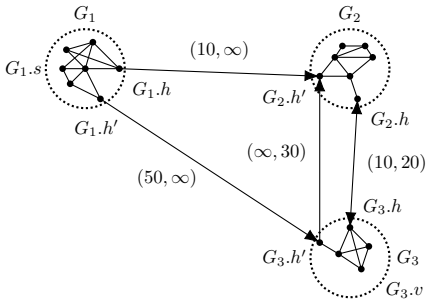
[5] Or negotiating domain authorities.

[6] The directionality of bridges is enforced by hub entities not exchanging system path information against the direction of an incident bridge.

of comprehension I have replaced the Bridge-to *bridge labels in Figure 9.2b with ordered pairs of bridge costs (where the ordering is derived from the graph number, i.e., $G_1 < G_2 < G_3$). The label $(10, \infty)$ between $G_1.h$ and $G_2.h'$ indicates that the cost is 10 from $G_1.h$ to $G_2.h'$ and the cost is ∞ (due to there being no bridge) from $G_2.h'$ to $G_1.h$.*

Bridge	Cost
$(G_1.h, G_2.h', \text{Bridge-to})$	10
$(G_1.h', G_3.h', \text{Bridge-to})$	50
$(G_2.h, G_3.h, \text{Bridge-to})$	10
$(G_3.h, G_2.h, \text{Bridge-to})$	20
$(G_3.h', G_2.h', \text{Bridge-to})$	30

(a) Bridge costs



(b) Bridged system group (1,2,3) with ordered bridge costs labelling bridges

Figure 9.2: Bridge costs and system path tables

Table 9.1 represents a composite system path table for the entire bridged system group shown in Figure 9.2b. Such a table will not exist in one place in practice, each hub's individual system path table amounts to a row from Table 9.1. Whilst a large bridged system group, such as in the case of Internet of Things (see Chapter 11), will likely employ a large number of individual system graphs and, therefore, a large number of bridges, each hub only processes information from it's neighbours allowing this function to easily scale.[7]

		Target hub					
		$G_1.h$	$G_1.h'$	$G_2.h$	$G_2.h'$	$G_3.h$	$G_3.h'$
Source hub	$G_1.h$	$(0, -)$	$(0, G_1.h')$	$(10, G_2.h')$	$(10, G_2.h')$	$(20, G_2.h')$	$(20, G_2.h')$
	$G_1.h'$	$(0, G_1.h)$	$(0, -)$	$(10, G_1.h)$	$(10, G_1.h)$	$(20, G_1.h)$	$(20, G_1.h)$
	$G_2.h$	$(\infty, -)$	$(\infty, -)$	$(0, -)$	$(0, G_2.h')$	$(10, G_3.h)$	$(10, G_3.h)$
	$G_2.h'$	$(\infty, -)$	$(\infty, -)$	$(0, G_2.h)$	$(0, -)$	$(10, G_2.h)$	$(10, G_2.h)$
	$G_3.h$	$(\infty, -)$	$(\infty, -)$	$(20, G_2.h)$	$(20, G_2.h)$	$(0, -)$	$(0, G_3.h')$
	$G_3.h'$	$(\infty, -)$	$(\infty, -)$	$(20, G_3.h)$	$(20, G_3.h)$	$(0, G_3.h)$	$(0, -)$

Table 9.1: Composite system path table for bridged system group (1,2,3)

9.2.2 Inter-Operation Request Evaluation

As presented in previous chapters, the RPPM model evaluates local requests within a stand-alone system graph using two steps: compute principals, where principal-matching rules are evaluated to determine whether security principals match to the request; and compute authorizations, where authorization rules are evaluated to

[7] An appropriate mechanism (e.g., digital signatures) must be in place to enable the receiving hub to ensure the authenticity and freshness of this information. However, I do not mandate the mechanism here.

determine if the matched security principals are authorized to perform the requested action. The introduction of inter-operation in Inter-RPPM does not change this local evaluation process, and it does not extend the authorization scope of a system graph's PDP outside of that system graph. (Authorization is still only decided for local principals, although, as I will show, these decisions can be made on behalf of remote subjects, and these local principals may be activated by remote principals.) However, as well as the existing local requests, within Inter-RPPM, remote requests may be made by a subject $G.s$ to perform a remote action $G'.a$ (where $G \neq G'$) on a remote object $G'.v$. As the authorization scope of system graphs G and G' are not extended, such a remote request relies upon PDPs within both system graphs (and any intermediary graphs also) in order to make the authorization decision for the requested remote action.

To support these remote actions I introduce an *originating remote request (ORR)* and an *incoming remote request (IRR)*.[8] The ORR represents the remote request as it is specified within the originating system graph. The IRR contains additional data which enable subsequent system graphs to contribute to the evaluation. When processing remote requests, every traversed system graphs' PDP employs the compute principals step (whether evaluating an ORR or IRR), but only the target system graph's PDP computes authorizations. Essentially, non-originating system graphs re-compute the set of matched principals based on the set computed by the preceding system graph and identified in the IRR.

Inter-Operation Policy Graph Evaluation

I base Inter-RPPM on the RPPM_{1a} model as I make use of graph-based principal-matching policies. More specifically, a policy graph is used during the compute principals step to order the evaluation of principal-matching rules and to enable principals to be "activated" when specific other principals are matched to a request. The edges of the policy graph determine which rules trigger other rules. Recall that in the context of a principal-matching rule, a target is a path condition or one of two special targets: all or none, where all is always satisfied and none is never satisfied. I update the definition of a principal-matching rule to make use of *extended targets*.

Definition 9.2. *An* extended target *is either a target or a set of principals P_t. Given a system graph $G = (V, E)$, $u, v \in V$, a set of matched principals $[\![\rho]\!]$ and an extended target P_t, then $G, u, v \models P_t$ if and only if $P_t \subseteq [\![\rho]\!]$. Conversely, $G, u, v \not\models P_t$ if and only if $P_t \cap [\![\rho]\!] = \emptyset$.*

[8] Recall that directed bridges link system graphs, and so, conceptually, a system graph sequence refers to the chain of system graphs between the originating system graph and the target system graph.

Rules making use of one or two extended targets, such as (P_t, none, p), activate the rule's principal based on constraining principals in the extended target(s). These constraining principals are either required or forbidden in the current set of matched principals, where that set may come from earlier in the policy graph's evaluation or from some previous evaluation (as will be seen in the case of remote requests).

Example 9.3. *I illustrate a policy graph for system graph G_2, containing examples of these various rules, in Figure 9.3.*

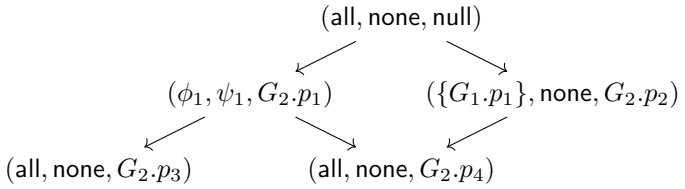


Figure 9.3: G_2's policy graph

In the case of $(\phi_1, \psi_1, G_2.p_1)$, the principal $G_2.p_1$ will be matched if ϕ_1 is satisfied and ψ_1 is not. The rule $(\{G_1.p_1\}, \mathsf{none}, G_2.p_2)$ enables local evaluation in system graph G_2 to directly make use of the matching of principal p_1 in system graph G_1; in this case $G_2.p_2$ will be matched if $G_1.p_1$ was matched in G_1. In this way, I am able to transfer authorization information between autonomous system graphs. (Principal-matching rules that employ extended targets are similar to RT rules of the form $A.R \leftarrow B.R'$, which asserts that an entity assigned to role R' by domain B is also assigned to role R in domain A [125].) The rule $(\mathsf{all}, \mathsf{none}, G_2.p_3)$ will only be evaluated if $G_2.p_1$ was matched, with the principal $G_2.p_3$ being matched automatically when it is. Finally, if principals $G_2.p_1$ and $G_2.p_2$ were matched because their rules where applicable, then rule $(\mathsf{all}, \mathsf{none}, G_2.p_4)$ causes $G_2.p_4$ to also be matched.

More formally, I evaluate a policy graph in Inter-RPPM with respect to a request, in order to compute a set of matched principals, as per Algorithm 9.1.[9]

Originating Remote Request

An originating remote request (ORR) is made within the originating system graph and is processed by the local PDP as per Algorithm 9.2, using a policy graph to compute principals.[10]

[9]Note that the algorithm is passed a set of matched principals to which any it matches are added, and that when evaluating each target it considers the principals in that set if the target defines principals which must be present (or not). Similar to Algorithm 5.1, the local policy graph variant of ComputePrincipals, I do not add the null principal to the set of matched principals so that it does not interfere with authorization decisions, no matter the conflict resolution strategy employed.

[10]For ease of comprehension, I have simplified the call to ComputePrincipals removing the system graph, relationship labels, principals, policy graph and principal matching strategy arguments so as to highlight the start entity, target entity, and set of matched principals arguments.

Algorithm 9.1 ComputePrincipals (inter-operation policy graph variant)

Require: System graph $G = (V, E)$, set of relationship labels $\widetilde{R}$, set of principals P, start entity s, target entity o, policy graph $G_\rho = (V_\rho, E_\rho)$, principal-matching strategy σ and set of matched principals $[\![\rho]\!]$
Ensure: Returns an updated set of matched principals $[\![\rho]\!]$
1: $M_q \leftarrow (V, \widetilde{R}, E, s, \{o\})$
2: **while** Perform breadth-first search of G_ρ starting at root vertex **do**
3: Evaluate current vertex, $(\phi, \psi, p) \in V_\rho$
4: **if** $(\phi = \mathsf{all})$ **or** $(\phi \subseteq P$ **and** $\phi \subseteq [\![\rho]\!])$ **or** $(L(M_\phi) \cap L(M_q) \neq \emptyset)$ **then**
5: **if** $(\psi = \mathsf{none})$ **or** $(\psi \subseteq P$ **and** $\psi \cap [\![\rho]\!] = \emptyset)$ **or** $(L(M_\psi) \cap L(M_q) = \emptyset)$ **then**
6: **if** $p \neq \mathsf{null}$ **then**
7: $[\![\rho]\!] \leftarrow [\![\rho]\!] \cup p$
8: **if** $\sigma = \mathsf{FirstMatch}$ **then**
9: **return** $[\![\rho]\!]$
10: **end if**
11: **end if**
12: **else**
13: Prune all (ϕ', ψ', p') from evaluation, where $(\phi, \psi, p) > (\phi', \psi', p')$
14: **end if**
15: **else**
16: Prune all (ϕ', ψ', p') from evaluation, where $(\phi, \psi, p) > (\phi', \psi', p')$
17: **end if**
18: **end while**
19: **return** $[\![\rho]\!]$

Definition 9.3. *Given two distinct system graphs $G = (V, E)$ and $G' = (V', E')$ in a bridged system group $(\mathcal{G}, \beta)$, an* originating remote request *takes the form $G.q = (G.s, G'.v, G'.a)$,[11] where $G.s \in V$, $G'.v \in V'$ and $G'.a \in G'.A$.*

Algorithm 9.2 ProcessORR

Require: System graph $G = (V, E)$, set of relationship labels $G.\widetilde{R}$, set of principals $G.P$, ORR $G.q = (G.s, G'.v, G'.a)$, policy graph $G.G_\rho = G.(V_\rho, E_\rho)$ and principal-matching strategy $G.\sigma$
Ensure: Malformed ORR rejected **or** set of matched principals $G.[\![\rho]\!]$ sent to next system graph in system graph sequence
1: $LCP_{G,G'} \leftarrow \mathsf{DetermineLeastCostPathToSystemGraph}(G')$
2: **if** G' reachable **then**
3: $G.h \leftarrow \mathsf{IdentifyLocalHubForLCP}(LCP_{G,G'})$
4: $G.[\![\rho]\!] \leftarrow \mathsf{ComputePrincipals}(G.s, G.h, \emptyset)$
5: $G''.h \leftarrow \mathsf{IdentifyNeighbourHubForLCP}(LCP_{G,G'})$
6: Securely send $G.q$ and $G.[\![\rho]\!]$ to $G''.h$ via $G.h$
7: **else**
8: Reject malformed ORR
9: **end if**

An ORR is considered malformed and rejected (denied) if there is no path between the originating and target system graphs (lines 2 and 8). When processing the ORR, the originating system's PDP must take into account that the target object is located in a distinct system graph from the subject. Therefore, the ORR is processed by performing compute principals between the subject and the hub entity which links the originating system graph to the next graph in the system graph sequence (lines 1, 3 and 4).[12] Note that line 4 passes an empty set to the ComputePrincipals

[11] Note that the originating remote request $G.q = (G.s, G'.v, G'.a)$ retains the same underlying *structure* as a local request made within the same system graph $G.q = G.(s, v, a)$.

[12] The function DetermineLeastCostPathToSystemGraph requires the PDP to interrogate its local hubs and to collate their responses, as described earlier and as demonstrated in Example 9.4.

algorithm (defined in Algorithm 9.1) as no principals have yet been matched to this remote request. The result of processing the policy graph is a set of matched principals for traversing the originating system graph. This is then passed, along with the details of the request, across the bridging relationship to the next system in the system graph sequence (lines 5 and 6).[13]

Example 9.4. *Returning to my bridged system group from Example 9.2, let us consider the ORR* $(G_1.s, G_3.v, G_3.a)$*. This request would be evaluated locally in system graph* G_1 *with a target system graph of* G_3*. The* G_1 *PDP can determine from the system path table entries of its local hubs (shown in Table 9.2), that* G_3 *is reachable and that the least cost path is via system graph* G_2*, accessible through the local hub* $G_1.h$ *and its neighbour* $G_2.h'$*.*

		Target hub	
		$G_3.h$	$G_3.h'$
Source hub	$G_1.h$	$(20, G_2.h')$	$(20, G_2.h')$
	$G_1.h'$	$(20, G_1.h)$	$(20, G_1.h)$

Table 9.2: Composite system path table fragment for G_1

The ComputePrincipals *algorithm will, therefore, attempt to match principals between* $G_1.s$ *and* $G_1.h$*. As already noted, the input to Algorithm 9.1 contains an empty set for the existing matched principals when processing an ORR. The set of matched principals* $G_1.[\![\rho]\!]$ *which results will then be securely sent to* $G_2.h'$*, via* $G_1.h$*.*

For illustration, Figure 9.4 shows the progress of the remote request once the ORR has been evaluated in G_1*. I have highlighted a path within* G_1 *to signify the completion of the principal matching process, and the bridge between* $G_1.h$ *and* $G_2.h'$ *to signify the transfer of the resulting matched principals.*

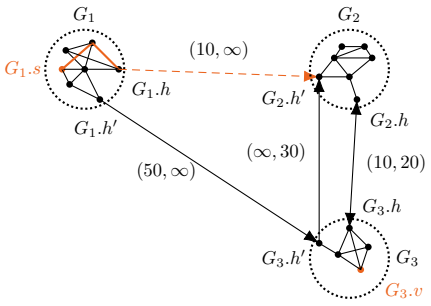


Figure 9.4: Progress of evaluating a remote request having evaluated the ORR $(G_1.s, G_3.v, G_3.a)$ in G_1

[13] An appropriate mechanism (e.g., digital signatures) must be in place to enable the receiving system to ensure the authenticity and freshness of such details.

Incoming Remote Request

An incoming remote request (IRR) is used in any other system graph along the system graph sequence, up to, and including, the target system graph. Within an IRR, the subject component of the ORR is replaced with a tuple identifying the originating subject, the hub entity through which the request was received by the current system and the set of matching principals which resulted from the preceding system's processing of the request. The IRR is processed by the local PDP as per Algorithm 9.3 and using a policy graph to compute principals.[14]

Definition 9.4. *Given a bridged system group* $(\mathcal{G}, \beta)$ *with system graphs and bridges producing the system graph sequence* $(G_1, \ldots, G_{i-1}, G_i, \ldots, G_\ell)$*, an* incoming remote request *in* $G_i = (V_i, E_i)$ *takes the form* $G_i.q = ((G_1.s, G_i.h, G_{i-1}.[\![\rho]\!]), G_\ell.v, G_\ell.a)$*, where* $G_i.h \in V_i$ *is the hub through which the request entered* G_i *and* $G_{i-1}.[\![\rho]\!]$ *is the set of matched principals computed by the preceding system graph.*

Processing of the IRR depends on whether the PDP is in an intermediate system graph or the target system graph (lines 1 and 8).

Algorithm 9.3 ProcessIRR

Require: System graph $G_i = (V_i, E_i)$, set of relationship labels $G_i.\widetilde{R}$, set of principals $G_i.P$, IRR $G_i.q = ((G_1.s, G_i.h, G_{i-1}.[\![\rho]\!]), G_\ell.v, G_\ell.a)$, policy graph $G_i.G_\rho = G_i.(V_\rho, E_\rho)$ and principal-matching strategy $G_i.\sigma$
Ensure: Set of matched principals $G_i.[\![\rho]\!]$ sent to next graph in system graph sequence (intermediate system graph) **or** authorization decision made (target system graph)
1: **if** $G_i \neq G_\ell$ **then**
2: // intermediate system graph
3: $LCP_{G_i,G_\ell} \leftarrow$ DetermineLeastCostPathToSystemGraph(G_ℓ)
4: $G_i.h' \leftarrow$ IdentifyLocalHubForLCP(LCP_{G_i,G_ℓ})
5: $G_i.[\![\rho]\!] \leftarrow$ ComputePrincipals$(G_i.h, G_i.h', G_{i-1}.[\![\rho]\!])$
6: $G_{i+1}.h \leftarrow$ IdentifyNeighbourHubForLCP(LCP_{G_i,G_ℓ})
7: Securely send $G_i.q$ and $G_i.[\![\rho]\!]$ to $G_{i+1}.h$ via $G_i.h'$
8: **else**
9: // target system graph
10: $G_i.[\![\rho]\!] \leftarrow$ ComputePrincipals$(G_i.h, G_\ell.v, G_{i-1}.[\![\rho]\!])$
11: $G_i.[\![\varrho]\!] \leftarrow$ ComputeAuthorizations$(G_i.[\![\rho]\!])$
12: DecideAuthorizationResult$(G_i.[\![\rho]\!], G_i.[\![\varrho]\!])$
13: **end if**

When processing an IRR in an **intermediate system graph**, the system's PDP must take into account that neither the subject nor target object is located in the current system graph. Therefore, the IRR is processed by performing compute principals between two hub entities: the hub entity $G_i.h$ through which the request entered the system and the hub entity which links the intermediate system graph to the next graph in the system graph sequence (lines 4 and 5). The set of matched principals from the preceding system graph is input into the graph policy evaluation

[14] Once again, for ease of comprehension I have simplified the call to ComputePrincipals removing the system graph, relationship labels, principals, policy graph and principal matching strategy arguments so as to highlight the start entity, target entity, and set of matched principals arguments.

(line 5) and may result in additional principals being activated (and added to it), as discussed earlier.

As with processing an ORR, the cumulative set of matched principals that results is then passed, along with the details of the request, across the bridging relationship to the next system in the system graph sequence (lines 6 and 7). This process continues until it reaches the target system.

Example 9.5. *At the end of Example 9.4, the hub entity $G_2.h'$ had received the ORR $(G_1.s, G_3.v, G_3.a)$ and the set of matched principals $G_1.[\![\rho]\!]$ from $G_1.h$. The IRR $((G_1.s, G_2.h', G_1.[\![\rho]\!]), G_3.v, G_3.a)$ is, therefore, generated by $G_2.h'$ and then evaluated in G_2 with a target system graph of G_3. The G_2 PDP can determine from the system path table entries of its local hubs (shown in Table 9.3), that G_3 is reachable and that the least cost path is a direct bridge from the local hub $G_2.h$ to its neighbour $G_3.h$.*

		Target hub	
		$G_3.h$	$G_3.h'$
Source hub	$G_2.h$	$(10, G_3.h)$	$(10, G_3.h)$
	$G_2.h'$	$(10, G_2.h)$	$(10, G_2.h)$

Table 9.3: Composite system path table fragment for G_2

The ComputePrincipals *algorithm will, therefore, attempt to match principals between $G_2.h'$ and $G_2.h$ using $G_1.[\![\rho]\!]$ as part of the input. The set of matched principals $G_2.[\![\rho]\!]$ which results will then be securely sent to $G_3.h$, via $G_2.h$.*

For illustration, Figure 9.5 shows the progress of the remote request once the IRR has been evaluated in G_2. I have highlighted a path within G_2 to signify the completion of the principal matching process, and the bridge between $G_2.h$ and $G_3.h$ to signify the transfer of the resulting matched principals.

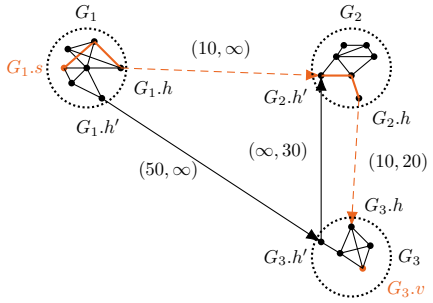


Figure 9.5: Progress of evaluating a remote request having evaluated the IRR $((G_1.s, G_2.h', G_1.[\![\rho]\!]), G_3.v, G_3.a)$ in G_2

When processing an IRR in the **target system graph**, the policy graph evaluation is performed between the hub entity through which the request entered the

system and the object of the request, note $G_i = G_\ell$ (line 10). Once again this makes use of the preceding system graph's set of matched principals. However, once a local set of matched principals is determined the compute authorizations step is performed. This allows a set of authorization decisions to be determined (line 11). A definitive decision is then determined, with the same default decision process and conflict resolution process employed by RPPM_{1a} for local requests (line 12).

9.2.3 Caching in Inter-RPPM

When considering caching, as introduced in Chapter 5, alongside support for the inter-operation of multiple systems, it is necessary to take into account the four variants of request processing that may occur in a system graph within a bridged system group (as shown in Table 9.4).

Request type	**Request**	**Graph**	**Source**	**Target**	**Remote Input**
Local	(s, o, a)	G	s	o	n/a
ORR	$(G.s, G'.v, G'.a)$	G	$G.s$	$G.h$	$\emptyset$
IRR - intermediate	$((G_1.s, G_i.h, G_{i-1}.[\![\rho]\!]), G_\ell.v, G_\ell.a)$	G_i	$G_i.h$	$G_i.h'$	$G_{i-1}.[\![\rho]\!]$
IRR - target	$((G_1.s, G_i.h, G_{i-1}.[\![\rho]\!]), G_\ell.v, G_\ell.a)$	$G_i = G_\ell$	$G_i.h$	$G_\ell.v$	$G_{i-1}.[\![\rho]\!]$

Table 9.4: Request processing variations

Recall that caching edges connect entities in a single system graph, and so are always added between the entities involved in the request evaluation within that system graph. As I have demonstrated, when evaluating remote requests the subject and object are in different system graphs, and so the evaluation performed in any one graph must consider paths involving at least one hub entity (with that hub entity acting as a gateway between system graphs). For example, in the target system graph containing the object of the remote request, the actual evaluation (and, therefore, the resultant caching edge) will be between the hub entity through which the request entered the target system graph and the request's object.

In RPPM_{1a}, as described in Chapter 5, having evaluated a local request (s, o, a) a caching edge of the form $(s, o, [\![\rho]\!])$ is added to E, where $[\![\rho]\!] \subseteq P$ is the set of matched principals determined by the compute principals step of request evaluation. In Inter-RPPM, I instead label caching edges in system graph G_i with a tuple $([\![\rho_{in}]\!], G_i.[\![\rho]\!])$ to indicate the incoming set of matched principals $[\![\rho_{in}]\!]$ as well as the set subsequently determined from the compute principals step in G_i. In the case of local requests and originating remote requests, where no incoming set of matched principals applies, the label will always take the form $(\emptyset, G_i.[\![\rho]\!])$. Note that

the caching edge may only be used to bypass compute principals for a subsequent request if the first element of the caching edge's label is equal to the incoming set of matched principals.[15]

Therefore, having been added between two entities u and v, caching edges enable the compute principals step of request evaluation to be skipped when evaluating subsequent requests between u and v where the incoming set of matched principals is equal to that in the caching edge's label. When evaluating a request without an incoming set of matched principals, the existence of the caching edge $(u, v, (\emptyset, [\![\rho]\!]))$ enables the compute principals step of request evaluation to be skipped, with $[\![\rho]\!]$ used as the result. This may be the case when evaluating the local request (u, v, a) or when the originating remote request $(G.u, G'.v', G'.a')$ is being evaluated, where v is the hub entity which connects the current system graph G to the next system graph in the system graph sequence between G and the target system graph G'.

Remark 9.2. *For convenience and backwards compatibility, a caching edge labelled with just a set of matched principals is equivalent to a caching edge labelled with a tuple where the incoming set is the empty set, i.e.* $(u, v, [\![\rho]\!]) \equiv (u, v, (\emptyset, [\![\rho]\!]))$.

When evaluating an incoming remote request $((G_1.s, G_i.h, G_{i-1}.[\![\rho]\!]), G_\ell.v, G_\ell.a)$ and considering paths between $G_i.h$ and $G_i.v$, a caching edge $(G_i.h, G_i.v, ([\![\rho_{in}]\!], G_i.[\![\rho]\!]))$ only applies if the incoming set of matched principals $G_{i-1}.[\![\rho]\!]$ is equal to $[\![\rho_{in}]\!]$, with $G_i.[\![\rho]\!]$ used if it is. Otherwise, the request must be evaluated in full.

Whilst a partial purging strategy was advocated in Section 5.1.3, such an approach is precluded when Inter-RPPM is used with caching. Specifically the introduction of extended targets increases the potential inter-dependencies to the point that all caching edges must be removed whenever a model change is made. Given that caching edges are a useful addition for inter-operation, an aggressive purging strategy is, therefore, applied when they are employed.

9.3 Model Comparisons

9.3.1 Existing Inter-Operation Models

In contrast to existing support for administration within access control models, support for inter-operation is far more binary. Where models support it, the functionality is described in full as it is, usually, the salient feature of that

[15] Note that it is solely the incoming set of matched principals which must be matched, and not the system graph from whence it comes.

model [34, 35, 125, 149, 169, 170]. In particular, research into the use of RBAC in multi-domain scenarios has led to several approaches to inter-operation based principally upon role mapping. However, many models do not refer to it and, as far as I am aware, no other relationship-based access control model considers inter-operation.

Role-Based Access Control

SERAT Shehab *et al.* define a distributed secure interoperability protocol in which users are assigned roles in remote domains based upon cross-domain access agreements [170]. The order in which roles are acquired within each domain they access is referred to as the user's access path. These paths are checked to ensure that the principles of *autonomy* and *security* are both satisfied. The autonomy principle requires that a locally permitted access must also be permitted under secure inter-operation, whilst the security principle requires that a locally denied access must also be denied under secure inter-operation [90]. Whilst SERAT enables administrators to manage inter-domain request evaluation policies, the model assumes that administrators are trusted entities and so does not define the mechanisms which control their actions.

Multidomain RBAC Shafiq *et al.* introduce a policy composition framework that integrates the RBAC policies of initially distinct domains [169]. They represent the RBAC policies of individual domains using graph models (not dissimilar from, and capturing the information in, role hierarchies), and include links between graphs to show relationships between roles in the distinct domains. To compose a global access control policy they merge the policies of the collaborating domains. They, additionally, implement a conflict resolution technique to deal with conflicts which may arise from the differences in how each domain models or references its access control policies. As with SERAT, Shafiq *et al.* do not define the mechanisms through which administrative actions are controlled.

X-GTRBAC Admin As discussed in Section 8.3.1, Bhatti *et al.* extend various existing works to define X-GTRBAC Admin [34, 35, 113, 169]. The inter-operation component of X-GTRBAC Admin is based on Shafiq *et al.*'s multidomain RBAC model [169]. However, conflict resolution is extended in X-GTRBAC Admin to accommodate the dynamic nature of the inter-domain relationships now that administration is supported.

Attribute-Based Access Control

RT The focus of the RT model is distributed authorization through the delegation of authority between organisations using credentials [125]. Credentials provide capabilities in remote security domains, granted through the delegation of authority over attributes and the delegation of role activations. Specifically, an entity A may delegate authority over R to B through the credential $A.R \leftarrow B.R$, and entity A may map a remote role $B.R_1$ to the local role $A.R$ through the credential $A.R \leftarrow B.R_1$. Such a delegation can be achieved in Inter-RPPM through the use of principal activation and an authorization rule of the form $(\{B.R\}, \mathsf{none}, A.R)$ in system graph A. Whilst administration of the RT model is not clearly documented, Li *et al.* do indicate that credentials are digitally signed and also indicate that the entity which issued a particular credential or request may be identified, for example through the use of a particular key.

XACML The XACML Administration and Delegation Profile specification defines how issuers may delegate authority to others [140]. It is not clear from the specification whether a delegate may be identified within another security domain; however, a template-based approach to (federated) inter-domain delegation using the specification has been introduced [149]. Perez *et al.* use attributes from a "home" domain during the evaluation of access requests in the "visited" domain. In order to simplify the creation of policies by delegates they employ policy templates built using a policy management tool.

9.3.2 Inter-Operation Enhancements

Whilst my work in respect of relationship-based access control inter-operation is novel, the inter-operation of other access control model instances has been considered in previous literature as discussed in Section 9.3.1.

Recall that Shehab *et al.* enable users to be assigned roles in remote domains with the order in which roles are acquired referred to as the user's access path [170]. These paths are checked to ensure the principles of autonomy and security [90] are both satisfied. In Inter-RPPM, remote requests, by construction, are unable to target local resources and the outcome of local request evaluation is unchanged by the additions for remote requests. Therefore, both of these principles are satisfied trivially.

Recall also that Shafiq *et al.* introduce a policy composition framework and implement a conflict resolution technique to deal with conflicts which may arise

from the differences in how each domain models or references its access control policies [169]. Within Inter-RPPM I have intentionally avoided integrating the policies of component model instances, and instead have provided a framework through which those policies may be applied to remote requests.[16] This places a greater requirement upon the manual definition of appropriate policies within each component model instance; however, it ensures a consistent policy language and maintains the autonomy of the security domains.

As already noted, Inter-RPPM's use of extended targets and principal activation is similar to RT's role mapping [125]. However, whilst Inter-RPPM details a process by which requests may traverse multiple security domains, RT only considers pairwise interactions. That is not to say that more complex arrangements could not be supported by RT; however, no indication is given as to how domain collaborations are established and how inter-domain requests are routed.

9.4 Summary

I have introduced two enhancements to RPPM_{1a} to produce the Inter-RPPM model. I have shown how system graphs may be connected via bridges to create a bridged system group, and how these system graphs may participate in evaluating remote requests (whether they originated in that system graph or not). In the next chapter I describe how RPPM's functionality may be consolidated and applied to a popular communication architecture.

[16] That being said I support the use of remote principals, where desired, to provide for more robust policies.

Chapter 10

RPPM-PS

With the increasing interconnection of computers into larger and more complex networks, numerous communication architectures have been proposed, developed and regularly applied. Whilst these architectures are frequently transparent to the users of computer systems, their selection and application for particular use cases is a conscious decision of system designers. Such decisions are particularly influenced by the goals of the services which the system will provide. For example, many enterprise networks are variations on client-server networks (so that company data may be centrally managed), whilst many consumer services are today commonly deployed using a cloud-based architecture (to facilitate global use and efficient content distribution).

As computer systems are integrated into an ever increasing, and widely dispersed, array of machinery and electrical systems, distributed architectures are becoming more relevant. Recent decades, for example, have seen a significant growth in the automation of industrial control systems (ICS) which has raised their profile with regard to critical national infrastructure (CNI) [47, 139] and cyber security [91, §5.4].

A popular approach to information dissemination in large-scale, distributed applications (such as ICS [116, §5]) is that of *publish/subscribe* [20, 68, 73]. Specifically, in publish/subscribe systems *event* notifications generated by *publishers* are distributed to authorized *subscribers*. There is a clear need for access control at both ends of this exchange; only authorized publishers should be able to generate event notifications, with only authorized subscribers able to register (or unregister) their interest in receiving these events.

In this chapter I shall introduce RPPM-PS, a relationship-based publish/subscribe access control system which makes use of the functionality of $\text{RPPM}_{1\text{b}}$, ARPPM, and Inter-RPPM. RPPM-PS is designed for general publish/subscribe systems but may be effectively tailored to more specific application

domains. (In Chapter 11 I will discuss Internet of Things as a topical and important domain; I will also discuss how RPPM-PS may be tailored to support the challenges of domains such as this.)

In this chapter, as a precursor to introducing RPPM-PS, I will first:

- Provide an overview of publish/subscribe systems and their components;
- Introduce the access control approaches so far considered for publish/subscribe systems;
- Consolidate the various RPPM enhancements introduced in this thesis, producing the $RPPM_3$ model; and
- Define the specific handling required to enable auditing to work in conjunction with the inter-operation features of $RPPM_3$.

I then explain how $RPPM_3$ may be applied to publish/subscribe systems, to produce RPPM-PS, by considering:

- The arrangement of entities in the system graph and the specific forms of request;
- The use of *services* and the need for *service discovery*;
- The process for *subscription management*; and
- The ways in which caching, administration and inter-operation are used in RPPM-PS.

10.1 Publish/Subscribe

Publish/subscribe systems are a form of communication architecture in which *subscribers* are able to register their interest in data sources made available by *publishers*. In many cases this is done through intermediary entities, called *event services* or *brokers*, which act as an abstraction and aggregation layer between the other network clients (subscribers and publishers).

Once such interest has been registered, a subscriber will receive an *event* message (in conjunction with all other subscribers) whenever the publisher publishes a *notification* matching the subscriber's registration. In this way, publish/subscribe systems provide a many-to-many communication architecture, which allows the publisher and subscriber to be decoupled in three dimensions [73]:

- Synchronization – the subscribers may be asynchronously notified of an event, with both the subscriber and publisher able to perform other processes whilst the end-to-end event notification takes place.
- Time – the presence of the broker enables the subscriber to be offline when the publisher sends an event message, and the publisher may, in turn, be offline when the subscriber receives that event from the broker.
- Space – the presence of the broker also enables the publishers and subscribers to be unaware of each other's identity.

This decoupling means that publish/subscribe systems are particularly suited to large-scale, distributed information dissemination applications. In order to support various means of specifying events in such applications, three variant schemes have been proposed [73]:

- Topic-based – where brokers present service interfaces for individual, named topics and subscribers register their interest in a topic as a whole.
- Type-based – where subscribers register their interest in types of event, rather than all those events under a named topic.
- Content-based – where publishers categorise individual events, and subscribers register their interest in specific content by specifying filters over event properties.

Past research has highlighted the suitability of publish/subscribe systems to large, distributed applications in domains such as: news delivery; stock market quote distribution; air traffic control; police infrastructure; healthcare systems; environmental monitoring; industrial automation; wireless sensor networks (WSN); and Internet of Things (IoT) [20, 106, 111, 148]. Whatever the application, models of publish/subscribe systems are commonly illustrated and discussed using graphs, where the edges represent connections between subscribers, brokers, and publishers [19, 20, 26, 106, 111, 148, 150]. I believe that graphs of relationships are an obvious substitution for these connection graphs, and that the resulting inter-relationships may be intuitively used to inform authorization decisions about subscription requests.

In fact, I believe, as do others [26, 111, 150], that access control is a key concern of publish/subscribe systems as it is necessary to manage the distribution of data being disseminated within the system. In many cases, the desire to control the data flow comes from the fact that the service publishing that data requires payment for access

to the information [26]. However, in some cases the control relates to privacy and safety concerns (such as in the case of police and healthcare environments [111]). Whatever the driver, a number of publish/subscribe models have been produced with consideration given to access control.

10.1.1 Access Control Approaches

Whilst consideration has been given to security and access control in publish/subscribe systems, some of this work solely focuses on the architectures and protocols employed within an environment to disseminate information to "known authorized" clients whilst preventing unauthorized actions [151, 173]. In these cases authorization enforcement is considered, but the mechanism by which the model determines whether a client is authorized may be glossed over. I do not consider such work further.

In contrast, there are a number of publish/subscribe models which incorporate some consideration of authorization decision making, as well as enforcement.

Capabilities

In their multi-domain peer-to-peer publish/subscribe model, Pesonen *et al.* employ certificates within a simple public-key infrastructure (SPKI) as *signed capabilities* which grant privileges to clients [150]. Issuance of these capabilities to clients is delegated (within a security domain) by the relevant resource owner to the domain's *access control manager*. The resource owner may be one of two parties depending on the requested action: it is the type or topic owner in the case of capabilities granting privileges to extend existing event types or topics, to subscribe to a topic, or to publish events; and it is the *coordinating domain*[1] in the case of capabilities granting privileges to join the network, or to introduce a new event type or topic. As each security domain trusts the coordinating domain and each access control manager is delegated to by that coordinating domain, each access control manager trusts every other access control manager for their delegated privileges. Thus, a broker with a capability issued by a local access control manager may be authorized by a remote access control manager to connect to that remote domain; this connection creates a cross-domain bridge and enables events to be routed between the two domains. Similarly, a client in one domain may be authorized to subscribe to a topic, or publish an event, in another domain.

[1] The single, primary security domain which is responsible for forming the multi-domain publish/subscribe system.

RBAC

Bacon *et al.* previously detailed a multi-domain publish/subscribe model which protected service methods using roles [19]. The domains in Bacon *et al.*'s model may form a hierarchy with (parameterized) roles managed within those domains. The methods, which are authorized at a local broker, include: define; advertise; publish; and subscribe. The roles which protect use of these methods are activated within the authorization policy's OASIS[2] rules if *membership conditions* are satisfied. These conditions may include prerequisite roles, appointment certificates and environmental constraints.

ABAC

More recently, several approaches to using attributes to control access to published data have been presented.

Ion *et al.* present a scheme by which subscriptions and notifications are encrypted using attribute-based encryption (ABE) [111]. The use of key-policy ABE (KP-ABE) and ciphertext policy ABE (CP-ABE) ensure that decryption requires the satisfaction of access policies by attributes (be they attached to the ciphertext, in the case of KP-ABE, or the keys, in the case of CP-ABE[3]). In this way, only clients which have the required attributes are able to decrypt the subscription requests and events.

With the growth in interest of Internet of Things (IoT), See Chapter 11, Carminati *et al.* introduce a privacy framework to enable publishers (or more specifically their users) to control how data (and derived data) can be processed by subscribers [42]. To achieve this, event attributes are classified using a *data category tree* and policies constrain the *purposes* for which specific categories of data may be used. The framework also defines combining strategies by which policies may be automatically generated for data derived from multiple events.

10.1.2 Architectural Components

The existing research into access control in publish/subscribe systems is typified by a number of commonalities in the architectures of the systems considered.

Entity Types Unsurprisingly, the systems considered commonly include entities of three core types: subscribers; publishers; and brokers. In some cases these are sup-

[2]Open Architecture for Secure Interworking Services [99]

[3]Note that the key and ciphertext distinctions of KP-ABE and CP-ABE relate to where the policy is associated. The attributes are associated with the alternate component in each case.

plemented, for example with gateway entities [106] or access control managers [150]; in other cases the broker entity type may be split to allow for several different types of broker (such as those which are acting as PDPs and those which are not [26], and such as those specifically hosting subscribers, hosting publishers or acting as intermediaries [20]).

Protected Requests All publish/subscribe systems consider authorization of requests by a subscriber to register and deregister their interests, and requests by publishers to send notifications. Some researchers also consider requests by publishers to advertise the kinds of event that they publish and requests to create new kinds of event in the system [19, 150].

Inter-Operation As publish/subscribe systems are suited to large-scale, distributed applications, the systems commonly consider inter-domain interactions. The connectivity between such security domains varies, as does the placement and role of broker entities. Whatever the details, these systems enable subscribers in one domain to be notified of events published in another domain [19, 20, 106, 150].

Communication Confidentiality Finally, many systems assume that the communication media which transmit subscription requests and events are vulnerable to eaves-dropping. Whilst this is not an access control issue per se, these systems frequently suggest the use of encryption to protect the confidentiality of the data. I will not consider this point further in this thesis. However, I hold the view that data published must be appropriately transmitted to ensure confidentiality, integrity and availability for authorized subscribers. This may be achieved through physical proximity and direct connectivity, or may be achieved through a more explicit combination of procedural and technical security controls.

10.2 $RPPM_3$

Before I consider the specifics of publish/subscribe within RPPM-PS, I first combine the functionality of $RPPM_{1b}$, ARPPM, and Inter-RPPM. The resulting consolidated RPPM model ($RPPM_3$) enables multiple system graphs supporting administration (as per ARPPM) to inter-operate as part of a bridged system group (as per Inter-RPPM). Further, the $RPPM_3$ model also enables the use of:

- $RPPM_{1a}$'s request evaluation enhancements (policy graph evaluation, target-based request evaluation and caching edges) through Inter-RPPM;

- RPPM_{1b}'s policy configuration enhancements (history-based policies, separation of duty, binding of duty and Chinese Wall); and
- RPPM_{1c}'s targeting enhancement (path expressions) through ARPPM.

In many cases these components coexist trivially by definition (such as in the case of path expressions and policy graph evaluation[4]), or because the components do not interact (such as in the case of caching edges and audit edges). Where specific handling has been required in preceding models to enable components to work together, I have detailed the mechanisms employed in such cases; for example, the interaction of policy graph evaluation and target-based request evaluation was described in Section 5.1.2, and the interaction of caching and inter-operation was described in Section 9.2.3. The interoperability of enhancements in RPPM_3 requires consideration of both administration and auditing in light of inter-operation.

The consideration of administration is trivial but worthy of mention. Given that each of the system graphs in Inter-RPPM's bridged system group represents a distinct security domain and, thereby, a scope of administration, the handling of administrative requests in RPPM_3 is no different from ARPPM – administrative requests are only supported as local requests.

In contrast, auditing requires more specific handling in light of inter-operation and remote requests.

10.2.1 Auditing in RPPM_3

Similar to caching edges in RPPM_{1a}, audit edges introduced in RPPM_{1b} connect entities relevant to the evaluated request. When considered in respect of RPPM_3 and remote requests, there is a need to take into consideration the fact that one of the entities may be located in a remote system graph (just as I did for caching edges in Inter-RPPM, as described in Section 9.2.3). Recall from Section 9.2.2 that authorization scope within Inter-RPPM, and therefore in RPPM_3, is limited to individual system graphs. This continues with the introduction of audit edges, which are also limited to individual system graphs (although they may audit actions by remote subjects).

Adding Decision Audit Edges

In RPPM_{1b}, as described in Chapter 6, having evaluated a local request (s, o, a) a decision audit edge of the type $(s, o, a^{\oplus})$ or $(s, o, a^{\ominus})$ will be added to E depending

[4]Principal-matching rules in the policy graph may employ path expression targets without modifying the behaviour of the policy graph evaluation.

on whether the request was approved or denied, respectively. In RPPM_3, I instead label decision audit edges in system graph G_i with a tuple $(G_1.s, a^{\oplus})$ or $(G_1.s, a^{\ominus})$ to indicate the originating remote subject of the request as well as the decision. In the case of a local request (s, o, a) the label will take the form $(s, a^{\oplus})$ or $(s, a^{\ominus})$ depending on whether the request was approved or denied, respectively.

Remark 10.1. *For convenience and backwards compatibility,* $(s, o, (s, a^{\oplus})) \equiv (s, o, a^{\oplus})$ *and* $(s, o, (s, a^{\ominus})) \equiv (s, o, a^{\ominus})$.

However, in the case of remote requests in RPPM_3, it is only incoming remote requests evaluated in the target system graph which determine a result for the request. Therefore, decision audit edges are neither added during the evaluation of originating remote requests nor added during the evaluation of incoming remote requests in intermediate system graphs. In the case of an IRR $((G_1.s, G_i.h, G_{i-1}.[\![\rho]\!]), G_\ell.v, G_\ell.a)$ evaluated in the target system graph $G_i = G_\ell = (V_\ell, E_\ell)$, when the request evaluation is completed I add the edge $(G_i.h, G_\ell.v, (G_1.s, G_\ell.a^{\oplus}))$ or $(G_i.h, G_\ell.v, (G_1.s, G_\ell.a^{\ominus}))$ to E_ℓ depending on whether the request was approved or denied, respectively. These decision audit edges connect the hub entity through which the request entered the target system graph and the object of the request.

Adding Interest Audit Edges

I employ an equivalent approach to the addition of interest audit edges. Recall that in $\text{RPPM}_{1\text{b}}$ interest audit edges are used to enforce Chinese Wall, with interest audit edges added between subjects and entities in the subset C (e.g., companies) which are members of at most one conflict of interest class. I label interest audit edges in an RPPM_3 system graph G_j with a tuple $(G_1.s, i^{\oplus})$ or $(G_1.s, i^{\ominus})$ to indicate the originating remote subject of the request as well as the interest. In the case of a local request (s, o, a) the label will take the form $(s, i^{\oplus})$ or $(s, i^{\ominus})$ depending on whether the interest is active or blocked, respectively.

Remark 10.2. *Once again, for convenience and compatibility,* $(s, o, (s, i^{\oplus})) \equiv (s, o, i^{\oplus})$ *and* $(s, o, (s, i^{\ominus})) \equiv (s, o, i^{\ominus})$.

As with decision audit edges, in the case of remote requests in RPPM_3 it is only incoming remote requests evaluated in the target system graph which determine a result for the request. Therefore, interest audit edges are not added during the evaluation of originating remote requests or the evaluation of incoming remote requests in intermediate system graphs. In the case of

an IRR $((G_1.s, G_j.h, G_{i-1}.[\![\rho]\!]), G_\ell.v, G_\ell.a)$, evaluated in the target system graph $G_j = G_\ell = (V_\ell, E_\ell)$ and where $G_\ell.v$ is owned by $G_\ell.c$ a member of conflict-of-interest class $G_\ell.i$, when the request is authorized the following edges are added to E_ℓ:

- $(G_j.h, G_\ell.c, (G_1.s, i^{\oplus}))$;
- $(G_j.h, G_\ell.c', (G_1.s, i^{\ominus}))$ for all $G_\ell.c' \neq G_\ell.c$, where $G_\ell.(c, i, m) \in E_\ell$ and $G_\ell.(c', i, m) \in E_\ell$; and
- $(G_j.h, G_\ell.v, (G_1.s, G_\ell.a^{\oplus}))$.

These interest audit edges connect the hub entity through which the request entered the target system graph and the companies which are members of the conflict of interest class related to the object of the request.

Using Audit Edges in Principal Matching

In order to make use of audit edges when satisfying paths in an RPPM_3 system graph I customise the process by which I compute the request NFA $M_q = (V, \widetilde{R}, E, s, \{o\})$. Specifically, when evaluating a request made by the (local or remote) subject $G'.v'$ in RPPM_3, I define the request NFA as:

$$M_q = (V, \widetilde{R}, E', s, \{o\}), \text{ where}$$
$$E' = E \ \cup \ \{(u, v, r) : (u, v, (u', r)) \in E, u' = G'.v'\} \ \setminus \ \{(u, v, (u', r)) \in E\}.$$

This minor revision ensures that the request NFA only contains transitions with singleton labels (as per those in path conditions) and that any audit edge transitions are only included if the subject of the request which triggered the addition of that edge is also the subject of the current request.[5]

10.3 Publish/Subscribe in RPPM-PS

10.3.1 System Graph and Requests

I now develop the RPPM_3 model into the RPPM-PS publish/subscribe system. In order to support publish/subscribe I introduce the concept of *services* and also require an RPPM-PS model instance to include the actions publish, subscribe, and unsubscribe. Then, an RPPM-PS topic-based model using a single (flat) layer of

[5]The efficiency of computing E' will depend on the data structures employed by an implementation, and may be improved by sorting and labelling edges specifically for this process.

brokers has the following features.[6] The set of entities V includes non-intersecting sets of publishers V_{pub}, brokers V_{bro}, subscribers V_{sub} and services V_{srv}. Each broker entity $v \in V_{bro}$ may host one or more *services* to which an entity $u \in V_{sub}$ may subscribe. The notation $v^i \in V_{srv}$ is used to indicate a service of entity v for topic i.

I use the conceptual arrangement of Figure 10.1 to demonstrate the use of the RPPM_3 model.[7] Within a system graph, let us suppose that for every publisher v_p and every service ${v_b}^i$ I have $G, v_p, {v_b}^i \models \pi_1$; Hosts, and for every subscriber v_s and every service ${v_b}^i$ I have $G, v_s, {v_b}^i \models \pi_2$; Hosts. In order to support inter-operation of system graphs I enable brokers to be hub entities within the bridged system group, as will be demonstrated in Section 10.3.6.

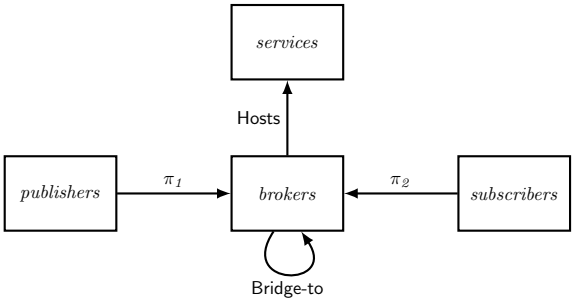


Figure 10.1: RPPM-PS generalisation

Definition 10.1. *A* publish request *takes the form* $q_p = (v_p, {v_b}^i, \textsf{publish})$*, where an entity* $v_p \in V_{pub}$ *requests to* publish *an event for topic* i *to service* ${v_b}^i$ *hosted by broker* $v_b \in V_{bro}$*. If* q_p *is authorized, then the event associated with the* publish *action will be sent to all entities* $U \subseteq V_{sub}$ *subscribed to* ${v_b}^i$*.*

Definition 10.2. *A* subscribe request *takes the form* $q_s = (v_s, {v_b}^i, \textsf{subscribe})$*, where an entity* $v_s \in V_{sub}$ *requests to* subscribe *to service* ${v_b}^i$ *hosted by broker* $v_b \in V_{bro}$*. If* q_s *is authorized, then* v_s *is said to be* subscribed *to* ${v_b}^i$*.*

In order for a subscribe request $q_s = (v_s, {v_b}^i, \textsf{subscribe})$ *to be* well-formed*, the subject* v_s *must not already be subscribed to service* ${v_b}^i$*.*

Definition 10.3. *An* unsubscribe *request takes the form* $q_u = (v_s, {v_b}^i, \textsf{unsubscribe})$*. If* q_u *is authorized then* v_s *is no longer subscribed to* ${v_b}^i$*.*

In order for an unsubscribe request $q_u = (v_s, {v_b}^i, \textsf{unsubscribe})$ *to be* well-formed*, the subject* v_s *must be subscribed to service* ${v_b}^i$*.*

[6] This approach works equally well as a type-based model. I have focused on a topic-based model solely for clarity of explanation. Multiple layers of brokers may also be supported by having brokers forward events to "downstream" brokers hosting the same topic service. The routing of such forwarding occurs using service discovery as discussed in Section 10.3.2.

[7] Figure 10.1 does not show a system graph, it shows a high-level representation of the "shape" of a system graph which may be part of a bridged system group.

Henceforth, I assume all subscription requests (subscribe or unsubscribe) are well-formed. Informally, publishers publish events to brokers, with the broker v_b hosting a service ${v_b}^i$ for topic i. Subscribers may subscribe to service ${v_b}^i$ in order to be informed about topic i. Should a subscriber no longer wish to be informed about topic i, they may send an unsubscribe request.[8]

Example 10.1. *In order to support publishing and subscription management in the arrangement indicated in Figure 10.1, I may employ the principal-matching rules*

$$(\{(\mathit{subject}, t_{pub}) \cdot \pi_1 \,;\, \mathit{Hosts} \cdot (\mathit{object}, t_{srv})\}, \mathsf{none}, p_{pub}),$$
$$(\{(\mathit{subject}, t_{sub}) \cdot \pi_2 \,;\, \mathit{Hosts} \cdot (\mathit{object}, t_{srv})\}, \mathsf{none}, p_{sub}),$$

and authorization rules

$$(p_{pub}, \{t_{srv}\}, \{\mathit{publish}\}, 1),$$
$$(p_{sub}, \{t_{srv}\}, \{\mathit{subscribe}, \mathit{unsubscribe}\}, 1).$$

Using this access control policy a publisher v_p may publish to a service ${v_b}^i$ using the p_{pub} principal and a subscriber v_s may subscribe or unsubscribe from service ${v_b}^i$ using the p_{sub} principal.

Remark 10.3. *Whilst it is instinctive to consider the authorization of access requests by users acting on system resources, recall (from Remark 4.1) that RPPM's mechanisms support requests made by any entity to act on any other entity – no matter what real or virtual system components these may relate to. This is particularly relevant in the case of publish/subscribe systems, where machine-to-machine (M2M) interactions are commonplace. The entities within RPPM's system graph are each instances of entity types present in the underlying system model. Whilst most general computing systems are likely to have one or more human user entity types, some publish/subscribe model instances (such as for WSN and IoT) may be completely devoid of these; instead, solely comprising entity types representing logical and (micro)electromechanical components.*

Within the RPPM-PS model the brokers act as PDPs and PEPs, thus reducing the load on publishers and subscribers.[9] RPPM-PS's request evaluation process enables the evaluation of local and remote requests as per RPPM_3. In this way,

[8] I require unsubscribe requests to be authorized in order to protect a publish/subscribe system from denial of service attacks caused by the unauthorized manipulation of internal data flows degrading or denying data availability.

[9] This approach is particularly desirable for more resource constrained applications such as WSN and IoT (see Chapter 11), but is also relevant to more general publish/subscribe systems which may contain some resource constrained nodes.

inter-domain subscription and publishing may be performed as per Chapter 9 and Section 10.2.1 (see Section 10.3.6).

10.3.2 Service Discovery

Whilst I have identified how publishing and subscription management requests may be made and evaluated, a client must first determine which service endpoint it should target for the related topic. The process of *service discovery* may be more, or less, complicated depending on the modelled system and the manner in which it is to be administered. A manually administered system may have no automated service discovery process, and may instead rely on the human users knowing which brokers host which topic services. In such a system a user will trigger a subscriber's subscription request by interacting with it, and so will specify the target service endpoint as part of that interaction. They may also interact with the publisher in order to specify the target service endpoint it should publish to.

In contrast, within systems using, at least some, automated administration or publishing there will be a need for an automatic service-discovery process. Given the variability in requirements for specific systems, I do not prescribe the format of such a process here. In some situations, a "known", centralized service-discovery broker may be present in the system, hosting an equally "known" topic service which reports the topic endpoints available from each of the system's brokers (a meta-topic if you will). In other situations, a decentralized, peer-to-peer, discovery process may be preferable. The benefits, limitations and application of such approaches have been commonly discussed in other work, such as [129].

Whatever the discovery mechanism employed, it should be possible for subscribers and publishers to identify brokers' service endpoints such that the desired topic events may be exchanged. Multiple brokers may host a service for the same topic, and some mechanism for selecting a particular broker is also required. The mechanism could involve proximity, loading, randomness or some other deciding factor. Finally, it should also be possible for brokers to identify other brokers hosting the same topic service so that they may forward events to ensure full coverage for subscribers.

10.3.3 Subscription Management

In order to track successful subscriptions I employ decision audit edges between clients and brokers. Specifically, a request $(v_s, {v_b}^i, \mathsf{subscribe})$ made by subscriber v_s will be evaluated by v_b; if authorized, it will result in an authorized decision audit

edge $(v_s, {v_b}^i, subscribe^{\oplus})$ between v_s and ${v_b}^i$.[10] If the request $(v_s, {v_b}^i, \mathsf{subscribe})$ is denied then a denied audit edge is added instead.

Subscriptions may be removed by the subscribing entity through the use of an unsubscribe request. A request $(v_s, {v_b}^i, \mathsf{unsubscribe})$ will be evaluated by v_b, as the subscribe request was. However, if authorized it results in the addition of the authorized decision audit edge $(v_s, {v_b}^i, unsubscribe^{\oplus})$ and I require that the authorized decision audit edge $(v_s, {v_b}^i, subscribe^{\oplus})$ be purged.[11]

It may also be desirable for a broker to perform a periodic "validation" of the subscriptions for topic i. Such a check forms a key part of usage control and avoids time-of-check-to-time-of-use (TOCTTOU) errors leading to unauthorized data access (for example, as a result of changes to the access control policy subsequent to a subscription being put in place). Depending on a system's requirements, subscription validation could be performed at specific times or intervals, after the publisher has published a certain number of events on the topic, or by all brokers after administrative changes are made. The criteria could even be specified for individual brokers or topic services, such that some brokers (perhaps associated with more sensitive data) perform validation more frequently.

Whatever the trigger, the validation process is conceptually simple. A broker v_b evaluates the request $(v_s, {v_b}^i, \mathsf{subscribe})$ for all v_s where there is an edge $(v_s, {v_b}^i, subscribe^{\oplus})$ in the system graph. If the request is authorized the subscription remains. However, if the request is denied the entry is no longer valid, the authorized decision audit edge $(v_s, {v_b}^i, unsubscribe^{\oplus})$ is added to the system graph and the edge $(v_s, {v_b}^i, subscribe^{\oplus})$ is purged.

In some publish/subscribe model instances it may be preferable to clearly distinguish the audit edges associated with subscriptions from those for other actions, for example if the model already relies heavily on history-based policies. There are various ways in which the implementation could achieve this; one way is that the subscription audit edges could be tracked by each broker for their hosted services using a *local subscription table*, as a part of the broker's internal state. Whilst this distributed approach modifies the location and presentation of the subscription records, the underlying processing is no different to that just described.

[10] I also require any existing authorized decision audit edge $(v_s, {v_b}^i, unsubscribe^{\oplus})$ be purged so that only the last subscribe or unsubscribe authorized decision audit edge exists.

[11] So that only the last subscribe or unsubscribe authorized decision audit edge exists.

10.3.4 Caching

The subscription validation process introduced in Section 10.3.3 requires re-evaluation of subscription requests to ensure the ongoing authorization of subscribers. "Active" publishers also require regular evaluation of publishing requests. It is, therefore, desirable for caching to be employed in order to reduce the processing during request evaluation by brokers. In the case of RPPM-PS, once a request is evaluated a caching edge is (automatically) added to the system graph. This edge can be used to bypass the compute principals step when evaluating future requests by the same subject on that object.

Example 10.2. *Returning to my example, if a request* $q_1 = (v_p, {v_b}^i, \mathsf{publish})$ *is evaluated with the set of matched principals* $\{p_{pub}\}$ *resulting, then a caching edge* $(v_p, {v_b}^i, \{p_{pub}\})$ *will be added to the system graph. On subsequent requests* $q_2, \ldots, q_n$ *by* v_p *to publish events to topic* i*, this caching edge may be used (until it is purged) to bypass the compute principals step of request evaluation.*

I employ an aggressive purging strategy in order to avoid caching edges being used once they are no longer valid. Specifically, I require that all caching edges be purged whenever an administrative change (other than the addition or deletion of caching edges and subscription-related audit edges) is made to the RPPM-PS model instance.

10.3.5 Administration

Recall that RPPM-PS is based on RPPM_3 such that its system graphs support administration. In this way, changes to the model instance, including the system graph and the policies, are authorized by evaluating administrative requests using RPPM. In order to evaluate and authorize administrative requests, the RPPM-PS model instance must be set up with its relationships and policies built on top of a suitable initial system graph, as described in Section 8.2.2. The initial system graph contains the necessary relationships such that an initial administrative entity, *root*, is able to initiate subsequent requests to construct the model instance and then manage it over the course of its lifetime (likely with the help of other administrators).

It is frequently desirable to distribute the responsibility of administering a system amongst a number of entities. With a single pair of administrative actions, this could result in an entity being overprivileged. Therefore, to provide more fine-grained administration of the RPPM-PS model instance, I replace ARPPM's coarse addEdge and deleteEdge administrative actions with a more refined set of initial

actions

$$A_I = \{\mathsf{publish}, \mathsf{subscribe}, \mathsf{unsubscribe},\\ \mathsf{addRelEdge}, \mathsf{deleteRelEdge},\\ \mathsf{addCachingEdge}, \mathsf{deleteCachingEdge},\\ \mathsf{addAuditEdge}, \mathsf{deleteAuditEdge}\}.$$

By using a more fine-grained set of initial actions, the permissions to administer normal relationship edges, automatically or manually, can be kept separate from those used to automatically administer caching edges and audit edges.

In addition, I introduce an automated administrative agent $root_a$ (of type *auto*) with equivalent relationships to the administrative entity *root*. Specifically, for every edge $(root, v, r)$ in the initial set of edges E_I, I add $(root_a, v, r)$ to E_I; and for every edge $(u, root, r) \in E_I$ I add $(u, root_a, r)$ to E_I. The new $root_a$ automated administrative agent is responsible for initiating and evaluating (as a PDP) automated administrative requests within the system. This includes administrative requests to add (and delete) caching edges and audit edges once requests have been evaluated.

Example 10.3. *Returning to my example once more, should I also desire that brokers be able to add new services (in order to introduce new topics) to the system represented conceptually in Figure 10.1, then I can make use of the administration part of RPPM-PS and include the principal-matching rule*

$$(\{(\mathit{subject}, t_{bro}) \cdot \diamond \cdot (\mathit{object\text{-}start}, t_{bro})\}, \mathsf{none}, p_{bro_\alpha})$$

and the authorization rule

$$(p_{bro_\alpha}, \star, \{\mathit{addRelEdge}, \mathit{deleteRelEdge}\}, 1).$$

These rules enable a broker v_b to have an administrative request to add a new service $(v_b, (v_b, t_{bro}, {v_b}^{i'}, t_{srv}, \mathsf{Hosts}), \mathit{addEdge})$ granted using principal p_{bro_α}. As discussed in Chapter 8, the new service ${v_b}^{i'}$ will be added to the system graph as part of adding the new Hosts edge between v_b and ${v_b}^{i'}$.

10.3.6 Inter-Operation

As RPPM-PS is based on RPPM_3 and includes support for inter-operation, publishers and subscribers may interact with remote brokers using remote requests. In order to enable individual publish/subscribe systems to interact, brokers may act as

hubs and be the endpoints of bridges between systems in the bridged system group. In such an arrangement the specific handling of auditing in light of inter-operation (outlined in Section 10.2.1) is required to enable remote subscriptions to be tracked within a model instance. However, other than that specific handling, the processing of remote requests is no different to that described in Chapter 9.

Example 10.4. *Let us consider the bridged system group shown in Figure 10.2. In this arrangement the broker $G_3.v_b$ hosts service $G_3.{v_b}^i$ for topic i in system graph G_3. Note also that $G_3.v_b$ is a hub for the bridge between system graphs G_2 and G_3.*

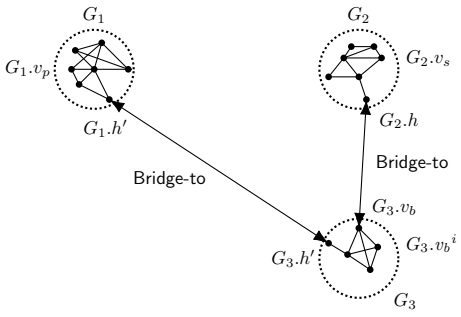


Figure 10.2: Publish/subscribe bridged system group (1,2,3)

A subscriber $G_2.v_s$ may subscribe to the topic i by issuing the originating remote request $q_1 = (G_2.v_s, G_3.{v_b}^i, G_3.\mathsf{subscribe})$ in system graph G_2. This will result in the incoming remote request $q_2 = ((G_2.v_s, G_3.v_b, G_2.[\![\rho]\!]), G_3.{v_b}^i, G_3.\mathsf{subscribe})$ being evaluated in system graph G_3. Assuming this request is authorized, the decision audit edge $(G_3.v_b, G_3.{v_b}^i, (G_2.v_s, G_3.subscribe^{\oplus}))$ will be added to system graph G_3, recording the subscription. The caching edge $(G_3.v_b, G_3.{v_b}^i, (G_2.[\![\rho]\!], G_3.[\![\rho]\!]_{q_2}))$ will also be added to G_3.

If publisher $G_1.v_p$ then sends a notification of an event to this topic by issuing the originating remote request $q_3 = (G_1.v_p, G_3.{v_b}^i, G_3.\mathsf{publish})$ in system graph G_1. This will result in the incoming remote request $q_4 = ((G_1.v_p, G_3.h', G_1.[\![\rho]\!]), G_3.{v_b}^i, G_3.\mathsf{publish})$ being evaluated in system graph G_3. Assuming this request is authorized, the decision audit edge $(G_3.h', G_3.{v_b}^i, (G_1.v_p, G_3.publish^{\oplus}))$ will be added to system graph G_3 and the event will be communicated to all subscribers of $G_3.{v_b}^i$, including $G_2.v_s$. In addition, the caching edge $(G_3.h', G_3.{v_b}^i, (G_1.[\![\rho]\!], G_3.[\![\rho]\!]_{q_4}))$ will be added to G_3.

10.4 Summary

I have introduced the RPPM-PS model as a consolidated RPPM access control model tailored to publish/subscribe. This model makes use of the various features

of RPPM presented in this thesis, and demonstrates that RPPM may be used in conjunction with other access control paradigms in order to meet the needs of various protection systems. Whilst security and access control have received reasonable attention in publish/subscribe systems (as described in Section 10.1.1), I believe that relationship-based access control offers a compelling alternative which intuitively aligns with how the design of such systems is regularly considered and conveyed. As I discussed (in Chapter 3) when motivating the need for RPPM originally: whilst other access control models already exist, none of these models (now including RPPM) is a perfect solution for all circumstances. Each has particular benefits and limitations; it is up to implementers to select the most appropriate model for their application, and up to access control model designers to ensure that the options and information are available for them to make such a selection. In the next chapter I will discuss how RPPM-PS may be tailored to interactions between more constrained devices in an Internet of Things.

Chapter 11

RPPM-IoT

At its core, Internet of Things (IoT) describes a system architecture in which uniquely identifiable physical objects, the "things", are interconnected using an internetwork in such a way that an emergent property of the system is derived from the processing of sensor (input) data;[1] this emergent property is commonly manifested through the triggering of actuators (output) based on the result of primary or secondary processing of that data. In the case of a smart healthcare environment, for example, various devices may provide data about the patient and thus enable medical staff to diagnose, monitor and treat the patient's ailments.

I believe that IoT's dependence on data and device functions means that access control, and the authorization of only legitimate requests, is a key aspect of any safe and secure solution. The growing research and media interests in Internet of Things highlight this topical and important application domain as one where security is particularly relevant [87, 107, 146, 177, 188].

11.1 Internet of Things

When examined carefully, the individual ideas which constitute IoT are not new, even in concert. However, technological and market developments in recent years have resulted in the concept of IoT becoming relevant to many aspects of modern life and enterprise. Specifically, the availability and reliability of fast internet connectivity across the developed world means that many people are able to make use of a data connection, of one form or another, twenty-four hours a day. The adoption of wireless technologies, smartphones and mobile apps has meant that much of this connectivity is utilised whilst people (and devices) are on the move. Developments in data analysis and machine learning have led to a greater capability and appetite

[1] Definition adapted from [137].

for processing large data sets, having been driven in part by the ubiquity of computing which is, subsequently, producing enormous volumes of data. Finally, the low cost and small size of electronic components means that there is little barrier to adding processing and networking capabilities to the many electrical, mechanical and even non-technological "things" that we choose to supplement our existence with. Whatever the drivers, various academic and industrial definitions of IoT exist, each differing in the degree to which they focus on users, applications, connectivity, protocols, availability, interactivity, intelligence and devices [148].

11.1.1 Body of Knowledge

Research into IoT began at the start of the twenty-first century after the term was first used in 1999 [16].[2] Whilst the topic has been around for more than fifteen years, many fundamental aspects of IoT remain open areas for investigation. Early work in the field was principally associated with the use of radio frequency identification (RFID) tags to track physical entities. Whilst this remains a prominent technology within IoT, as time has progressed it has become less dominant in research activities. Security and privacy considerations, in contrast, are gaining more interest, particularly as both academic [119] and industrial [146] researchers have had significant success demonstrating vulnerabilities in a broad range of IoT devices.

There are a number of survey papers which provide a broad summary of the current understanding of IoT. Atzori *et al.* identify IoT as the result of the convergence of three visions: the things-oriented, the internet-oriented and the semantic-oriented [18]. They describe a number of enabling technologies and identify a range of application domains, of which healthcare (as used in my examples) is one. In addition, they identify open issues whose resolution is crucial to the success of IoT: standardisation; addressing and networking; and security and privacy. In respect of security and privacy, they highlight authentication and data integrity as major security issues for IoT (where authentication is commonly employed as a precursor to authorization).

Gubbi *et al.* also highlighted open challenges in their survey paper [93]. Their analysis of IoT was driven more from a web services perspective, with the architecture specifically described through cloud computing, Platform-as-a-Service and Software-as-a-Service constructs. They identify several challenges including secure reprogrammable networks and privacy, which considers the need for nodes to be able

[2]Several authors have claimed that the term Internet of Things was used by the International Telecommunication Union (ITU) in 1997 [15, 137]. However, I believe they may have misinterpreted a statement in the forward of a 2005 ITU report using the term as its title, "'The Internet of Things' is the seventh in the series of 'ITU Internet Reports', originally launched in 1997 under the title 'Challenges to the Network'" [182].

to authenticate and authorize code updates and configuration changes.

Perera *et al.* provide a comprehensive survey (including the analysis of a large number of existing IoT projects) which once again identifies security and privacy (alongside trust) as the key topic requiring further research [148]. They highlight the role users have to play in IoT and the need for security and privacy to be addressed as part of winning the trust of these users.

Lastly, Weber cites four security and privacy requirements related to IoT: resilience to attack; data authentication; access control; and client privacy [186]. He also discusses the legal frameworks for IoT and identifies four axes representing the technical challenges associated with regulation. These challenges are:

- Globality – the need for international laws and legal systems;
- Verticality – the lifetime of an RFID tag should be at least the lifetime of the tagged product, to enable complete life cycle management;
- Ubiquity – the breadth of the interaction with the physical world; and
- Technicity – how the complexity of the technology impacts the development of rules.

These works highlight that IoT is a complex and multi-faceted topic; particular IoT systems are highly variable when compared to each other and to themselves at different moments in time. Given this innate variability and the interaction between an IoT system and its environment, the processing each system performs, and that performed as part of managing it, must be informed by its current state. Recall that the term *context* is defined by Abowd *et al.* as "any information that can be used to characterize the situation of an entity" [2]. I believe context is particularly relevant to IoT, and modify their extended definition only so as to clarify that "an entity is a person, place, or object that is considered relevant to the interaction between a user and an application" or between two applications, including the interacting parties themselves.

Whilst a few researchers, like Perera *et al.* [148], have demonstrated an awareness of the need for context in IoT, the topic of access control has been given limited attention so far, and some initial ideas have shown little appreciation of the constraints likely to arise in IoT application domains. Zhang and Tian attempt to make use of context in access control by extending RBAC to support context constraints for IoT [193]. Unfortunately, the implementation details are vague and the provided case study incomplete. Wu *et al.* also modify RBAC for IoT, this time to support cross-domain requests [190]. However, the extent to which context is supported is

unclear. Obligations and constraints from UCON are included in their modified RBAC model; however, no explanation or example of their use is provided.

Other researchers feel that RBAC is unsuitable for access control in IoT systems. Zhang and Gong suggest how UCON may be applied to IoT environments using a trust management center as an administrator of trust-related attributes for the devices [192]. Whilst this does leverage historical behaviour to inform future requests, the details are once again sparse. Gusmeroli *et al.* employ, overly weighty, digitally signed, XML-formatted capability tokens to communicate a subject's entitlements [95]. Authorization decisions are made following checks on:

- The formal validity of the capability token (and the associated authorization chain);
- The logical validity of the requested operation in respect of the capability token;
- The existence (or not) of local restrictions associated with the operations granted by the capability token; and
- The revocation status of the capabilities in the authorization chain.

However, whilst they identify that IoT impacts the relevance of context, they do not identify the mechanism through which this is considered by their approach when issuing or validating capability tokens.

In contrast, Bernabe *et al.* employ a combination of a lighter weight JSON-formatted capability token along with a fuzzy trustworthiness measure based upon quality of service, security, reputation and social relationships [28]. This approach displays greater awareness of the limitations of IoT devices. However, I believe their simple approach to social relationships, considering groupings (called bubbles) or whether two devices have interests-in-common or friends-in-common, is insufficient for the complex interactions which result in IoT environments. I, therefore, believe that RPPM's greater emphasis on social relationships makes it particularly well suited to supporting IoT and the interactions between IoT devices.

11.1.2 Devices

IoT devices are typically thought of as being relatively small, specialised and resource constrained. Whilst this is true in the case of many wireless sensor network (WSN) devices [106, 137], more recently manufacturers of non-IoT devices are adding processing and networking capabilities to rapidly produce new categories of IoT device. Whether adapted or designed for IoT, the components of the resultant devices can

be visualised using a layered model, as illustrated in Figure 11.1. At the lowest layer is the *hardware* of the device itself, on top of this is the *networking* layer, above this is the *device software* layer, and at the top is the *service* layer through which devices logically interact.

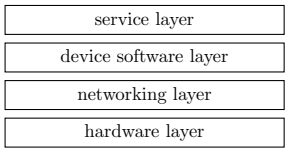


Figure 11.1: Device abstraction model

Remark 11.1. *It is worth noting that the term "constrained device" is often used in the IoT literature without a clearly stated definition (although examples may be cited) [28, 137, 136, 138, 147, 158].*[3] *For the purposes of this thesis I shall employ the definition given for a "constrained node" in RFC 7228 [38].*

> *"A node where some of the characteristics that are otherwise pretty much taken for granted for Internet nodes at the time of writing are not attainable, often due to cost constraints and/or physical constraints on characteristics such as size, weight, and available power and energy. The tight limits on power, memory, and processing resources lead to hard upper bounds on state, code space, and processing cycles, making optimization of energy and network bandwidth usage a dominating consideration in all design requirements. Also, some layer-2 services*[4] *such as full connectivity and broadcast/multicast may be lacking."*

IoT devices individually offer limited capabilities. It is the interconnection of multiple devices into a wider architecture which gives rise to an IoT system's emergent properties. Exactly how devices are interconnected is of little consequence at this level of abstraction. Various wireless communication technologies and protocols can be employed, each requiring a particular implementation and determining specific interoperability constraints. However, the underlying functionality of the device and the wider system is independent of such implementation details. What is more important is that they can interact through the devices' service layers producing, processing and consuming data to support higher functionality.

In a smart healthcare example I can envisage that the combination of a blood pressure cuff, pulse oximetry device[5], nurse's tablet computer, oxygen supply system

[3] Further, some authors refer to "highly constrained devices" and it is unclear if, and how, a distinction is drawn between constrained devices and highly constrained devices [136, 158].

[4] Where layer-2 here refers to layers in the OSI model, and not to layers from Figure 11.1.

[5] Used to non-invasively measure and display blood oxygen saturation.

and hospital's data storage node may enable a patient to be given the necessary supply of oxygen based on their body's requirements, whilst providing a real-time and historical log for the nurse so that they may monitor the patient's condition throughout treatment and recovery.

The interconnection of devices, such as these, within a wider IoT system introduces a new (system-wide) viewpoint which may be modelled in a similar fashion to the devices themselves. A comparable layered model for the system-wide architecture can be seen in [18], adapted in Figure 11.2, with *objects* at the bottom, *object abstraction* above this, and then *service management*, *service composition* and *applications* layered in turn. Alongside these layers Atzori *et al.* identify a vertical (cross-layer) component comprising the *management of trust, privacy and security*; this highlights that such functionality is required throughout the layered stack.

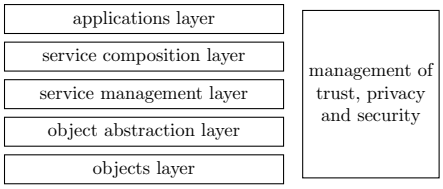


Figure 11.2: IoT system architecture model

11.2 Tailoring RPPM-PS to IoT

In Chapter 10 I introduced RPPM-PS as a relationship-based publish/subscribe access control system. Whilst the model I described is suitable for general publish/subscribe systems, I believe there is a need to tailor the model for application in an Internet of Things. I, therefore, identify particular characteristics and constraints of IoT which are relevant to access control, and identify the refinements which are required to tailor RPPM-PS to this application domain (producing the RPPM-IoT model).

Physical World Interaction The IoT application domain is commonly characterised by the existence of sensors and actuators where these components are associated with the measurement and manipulation of the physical world. A publish/subscribe approach is particularly relevant as there will be situations where sources may produce data at varying rates, intervals and times. This variability may be driven, for example, by the kind of sensor employed or the environment in which it is placed. In such situations actuators will be unable to predict when data would be available,

and a push, rather than pull, architecture is desirable.[6] *I, therefore, treat sensors as publishers and actuators as subscribers.*

Intermediate Processing As the emergent property of IoT systems is commonly manifested through the primary or secondary processing of data, it is desirable to support linear and non-linear arrangements of nodes between the sensor and actuator. Such arrangements will enable intermediate processing to take place, thus providing the actuator with processed, rather than raw, sensor data. These processed data may be, for example, refined, interpreted, supplemented or filtered so that the actuator may appropriately interact (e.g., physically, visually or audibly) with the physical world. Whilst a linear arrangement of nodes allows incremental changes to be made to the sensor data, non-linear arrangements enable a node to consume the same data from distinct peers which may have processed the data in different ways. By subscribing to these alternative data sources a single node may gain redundancy of input, or a more "information-rich" perspective of the original sensor's output. *I, therefore, introduce* processors *to perform this intermediate processing alongside RPPM-PS's publishers, subscribers and brokers.*

Resource Constrained Devices Whilst various publish/subscribe systems may include some resource constrained nodes, there is an expectation that a greater fraction of the nodes in an IoT system will be constrained, much like in a wireless sensor network [106, 137]. Where constrained nodes are present, the constraints that may be encountered are limitations in devices' "memory, computation, communication, latency and energy consumption" [136]. The IoT system must, therefore, include components which may perform access control processing without excessively burdening devices which are constrained. *I, therefore, only allow brokers (which I assume are always unconstrained) and unconstrained processors to host services and to be PDPs and PEPs, thus avoiding burdening publishers, subscribers and constrained processors.*

System Variability Finally, whilst some IoT systems will mostly comprise devices which are stationary and stable (such as in the case of smart meters, smart light bulbs, smart kitchen appliances, smart televisions, environmental monitoring systems and burglar alarm systems in a house), other systems will mostly comprise a dynamic collection of devices due to device movement (such as in the case of a smart

[6] This approach can also be seen in the observer design pattern which is implemented as part of the OBSERVE option of the Constrained Application Protocol (CoAP) [97].

transit environment) or intermittent connectivity (such as in the case of an "electromagnetically noisy" manufacturing environment). There is a need to facilitate this variability so that dynamic systems function efficiently without manual intervention. Depending on the size and stability of an IoT system, subscription-related requests may be made regularly as entities (temporarily) participate in the system and must re-establish all of their subscriptions upon each joining. *I, therefore, require the automated administrative agent* $root_a$ *to manage device membership of the system graph and allow administration to be triggered by various factors including proximity and liveness.*

I now introduce refinements to the RPPM-PS model as I tailor RPPM-IoT to satisfy these constraints. However, before I begin it is worth noting that many of RPPM-PS's features are useful and appropriate as they stand. Particularly, RPPM-IoT's use for IoT systems benefits from RPPM-PS's ability to:

- Perform context-aware request evaluation through the use of labelled relationships between concrete and logical entities in the system graph.
- Administer the access control model instance such that it always reflects the IoT system (even as components, temporarily, join and leave).
- Enable nodes to host data services to which other nodes may subscribe.
- Enable nodes to use service discovery to identify what service endpoints are available from other nodes.
- Perform (periodic) subscription validation to ensure timely revocation when subscriptions are no longer valid.
- Use caching to limit the processing overhead associated with request evaluation (in particular subscription validation and publishing).
- Enable multiple IoT systems to inter-operate.

11.2.1 IoT System Architecture

RPPM-PS's publish/subscribe conceptual arrangement of Figure 10.1 can be used as the basis for a similar arrangement for articulating most IoT architectures in RPPM-IoT (as shown in Figure 11.3, with the RPPM-IoT additions coloured blue).[7]

As indicated above, I believe it is natural to think of IoT sensors as publishers and IoT actuators as subscribers. However, in order that actuators may receive something other than raw sensor input, I introduce processor nodes and enable these to

[7] Figure 11.3 does not show a system graph, it shows a high-level representation of the "shape" of a system graph which may be part of a bridged system group.

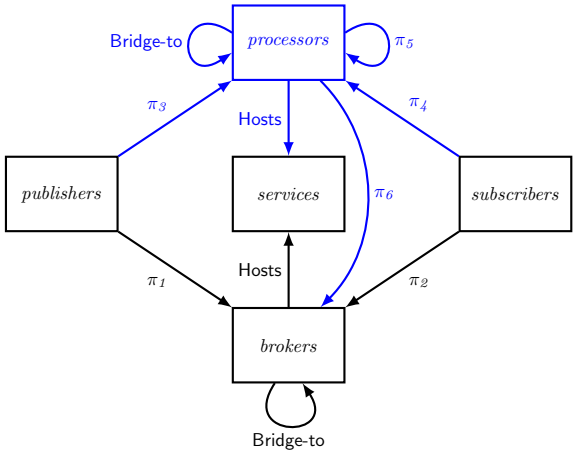


Figure 11.3: RPPM-IoT generalisation

perform some intermediate processing on data published to their hosted services;[8] I envisage that intermediate processing may include the storing, interpreting, modifying, annotating, combining, filtering, sampling or deleting of some amount of the data in the notification. With the addition of processors, I believe that any IoT device may be modelled using some combination of publishers (sensors), processors and subscribers (actuators).

Example 11.1. *For example, Figure 11.4 shows a data flow illustration*[9] *of a pulse oximetry device, where publishers are circles, processor nodes are hexagons, subscribers are squares and arrows indicate data flow.*[10] *Putting the device into its real-world context, the pulsox sensor takes light absorption readings, the processor calculates peripheral oxygen saturation values from these readings and the actuator displays those values to the user. (For the purposes of this thesis I shall assume that the pulsox device in question is not a constrained device.)*

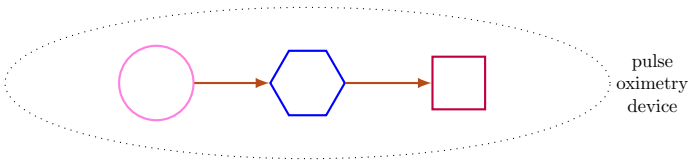


Figure 11.4: Pulsox data flow illustration

This example can be extended, adding four more IoT devices and a broker into

[8] For brevity I have indicated a single path condition (π_5) between processors, and another (π_6) between processors and brokers. In reality there may be different relationships between different sets of processors and brokers in the system graph, and so more than one path condition may apply here. Assuming the principal-matching rules exist to cater for this, I place no limitation on there being a single path condition in reality.

[9] I shall consider system graphs presently.

[10] For ease of discussion, I have not labelled the arrows to indicate what data fields are exchanged. I leave it to the reader to envisage appropriate fields given the example.

Figure 11.5, such that the pulsox device is part of a system with: a ward alerting node, which only comprises a processor; a simple external display, which only comprises an actuator; a ward occupancy broker, represented by a pentagon; and two bed occupancy sensors, which each comprise only a sensor.

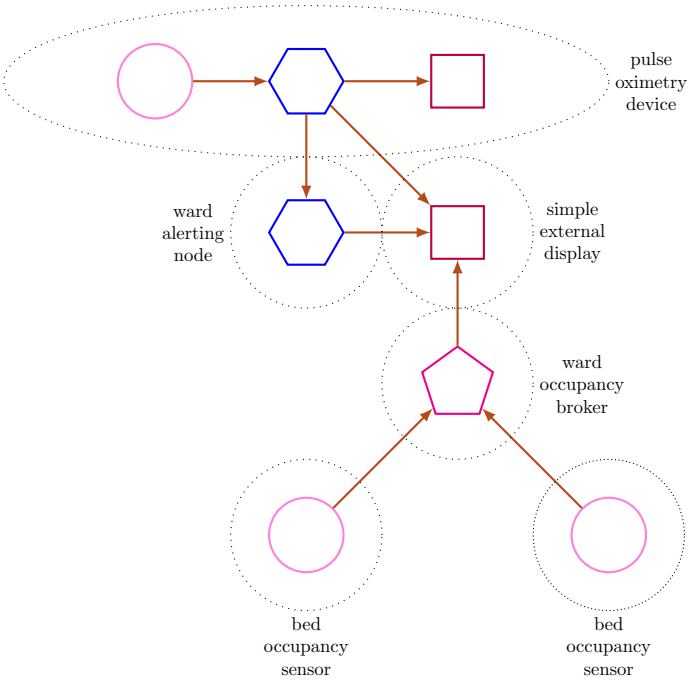


Figure 11.5: IoT healthcare data flow illustration

I will demonstrate the use of this system throughout the remainder of the chapter.

Given the conceptual arrangement in Figure 11.3, the set of entities V within an IoT system graph includes non-intersecting sets of publisher entities V_{pub}, processor entities V_{pro}, subscriber entities V_{sub}, broker entities V_{bro} and services V_{srv}. The notation $v^i \in V_{srv}$ is used to indicate the *ith* service of entity $v \in V_{pro} \cup V_{bro}$.

11.2.2 Brokers

Before I demonstrate how processors function, it is important to clarify the role of brokers in IoT systems. In RPPM-IoT, brokers continue to be used as they were in RPPM-PS; they relay (unmodified) event notifications which are published to their topic services.[11] In so doing, brokers distribute particular sensor data to all of the authorized subscribers of the sensor data topic. As in RPPM-PS, brokers may also act as hubs for bridges connecting multiple IoT system graphs, and so may

[11]Once again, this approach works equally well as a type-based model. I have focused on a topic-based model solely for clarity of explanation. There is also no issue supporting multiple layers of brokers which forward events amongst those hosting the same topic services on the way to subscribers.

distribute the sensor data to remote subscribers, where present, and may receive notifications from a remote publisher.

Example 11.2. *Figure 11.6 shows a system graph fragment for part of the healthcare IoT system introduced in Example 11.1. For ease of exposition the figure has been annotated with grey arrows indicating the intended direction of publish requests (dashed lines) and subscribe requests (dotted lines). Specifically, two IoT bed sensors publish data indicating whether the associated beds are occupied and the external display subscribes to this topic, and may thereby display the beds' current occupancy statuses. Given that none of these nodes is powerful from a processing perspective, the ward occupancy broker hosts the topic service (represented by a triangle) for these sensors and the actuator. (It would also be the PDP and PEP for the indicated requests.)*

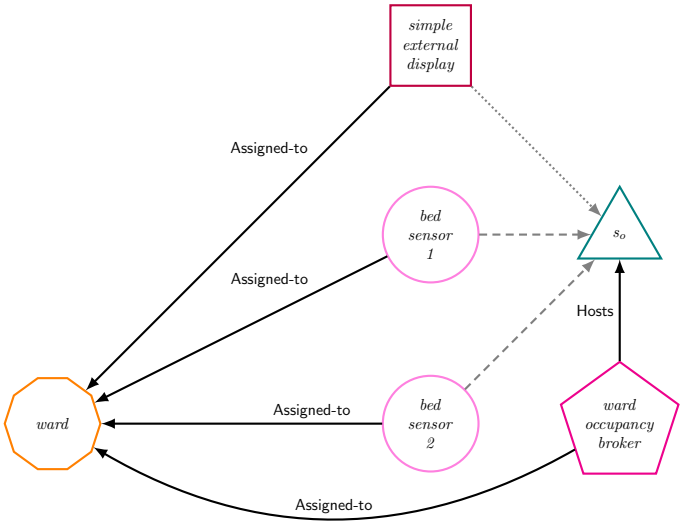


Figure 11.6: IoT healthcare system graph fragment 1 (annotated)

In this (simple) example I may choose to employ the principal-matching rules

$$(\{(\mathsf{subject}, t_{pub}) \cdot \mathsf{Assigned\text{-}to}\,;\, \overline{\mathsf{Assigned\text{-}to}}\,;\, \mathsf{Hosts} \cdot (\mathsf{object}, t_{srv})\}, \mathsf{none}, p_{pub}),$$
$$(\{(\mathsf{subject}, t_{sub}) \cdot \mathsf{Assigned\text{-}to}\,;\, \overline{\mathsf{Assigned\text{-}to}}\,;\, \mathsf{Hosts} \cdot (\mathsf{object}, t_{srv})\}, \mathsf{none}, p_{sub}),$$

and authorization rules

$$(p_{pub}, \{t_{srv}\}, \{\mathsf{publish}\}, 1),$$
$$(p_{sub}, \{t_{srv}\}, \{\mathsf{subscribe}, \mathsf{unsubscribe}\}, 1).$$

11.2.3 Processors

In contrast to brokers, processors may take a more active role in the IoT system by performing some intermediate processing on topic data.[12] No matter the specific type of intermediate processing to be performed, I require that an active processor:

- Subscribes to service topics which provide its input – this involves subscribing to one or more services which may each be hosted either by the processor in question or by some other processor or broker.
- Publishes its outputs to relevant service topics – this involves publishing to one or more services which, once again, may each be hosted either by the processor in question or by some other processor or broker.

This approach offers a broad range of capability; I highlight the extremes to illustrate this fact. In the simplest form, a constrained processor which does not host any services itself will act much like a combination of a subscriber and publisher; it performs intermediate processing on the events between receiving its registered notifications and publishing its output.

In the most complex form, an unconstrained processor may host several services itself; it subscribes to those services, and also to several services hosted by other entities spread across the system graphs of a bridged system group. From these subscriptions it receives numerous events which it uses as inputs to its intermediate processing. This processing may generate multiple outputs which are published to multiple services, several of which it hosts itself and others which are hosted by other entities spread across the system graphs of the bridged system group.

Example 11.3. *Returning to my healthcare example, consider the annotated system graph fragment in Figure 11.7. This shows the pulsox device comprising a sensor (publisher), core (processor) and UI (subscriber); services are, once again, represented by triangles. (As before, the solid arrows indicate the relationships, the dashed arrows indicate the direction of publish requests, and the dotted arrows indicate the direction of subscription requests.)*

As I described in Example 11.1, at an abstract level the pulsox sensor takes light absorption readings, the processor calculates peripheral oxygen saturation values from these readings and the actuator displays those values to the user. This is achieved by the pulsox sensor publishing the raw light absorption readings as events to the pulsox core's unprocessed input service s_u. The pulsox core is subscribed to this

[12] Processors are not required to perform intermediate processing, and may simply act as relays (in a similar fashion to brokers).

topic and processes the event readings, publishing the resulting peripheral oxygen saturation values as events to its processed output service s_p. The pulsox UI is subscribed to this topic and receives these event notifications, displaying them to the pulsox device's user.

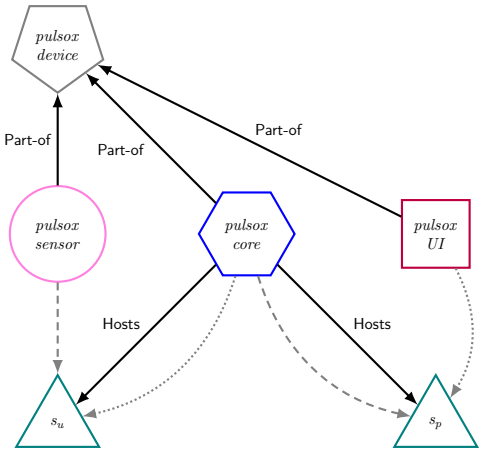


Figure 11.7: IoT healthcare system graph fragment 2 (annotated)

In order for the relevant request to be authorized in this example, I may choose to employ the principal-matching rules

$$(\{(\mathsf{subject}, t_{pub}) \cdot \mathsf{Part\text{-}of}\,;\, \overline{\mathsf{Part\text{-}of}}\,;\, \mathsf{Hosts} \cdot (\mathsf{object}, t_{srv})\}, \mathsf{none}, p'_{pub}),$$
$$(\{(\mathsf{subject}, t_{pro}) \cdot \mathsf{Hosts} \cdot (\mathsf{object}, t_{srv})\}, \mathsf{none}, p'_{pub}),$$
$$(\{(\mathsf{subject}, t_{sub}) \cdot \mathsf{Part\text{-}of}\,;\, \overline{\mathsf{Part\text{-}of}}\,;\, \mathsf{Hosts} \cdot (\mathsf{object}, t_{srv})\}, \mathsf{none}, p'_{sub}),$$
$$(\{(\mathsf{subject}, t_{pro}) \cdot \mathsf{Hosts} \cdot (\mathsf{object}, t_{srv})\}, \mathsf{none}, p'_{sub}),$$

and authorization rules

$$(p'_{pub}, \{t_{srv}\}, \{\mathsf{publish}\}, 1),$$
$$(p'_{sub}, \{t_{srv}\}, \{\mathsf{subscribe}, \mathsf{unsubscribe}\}, 1).$$

Remark 11.2. *To clarify, whilst there were no other subscribers present in Example 11.3, when events were published to the pulsox core's services, these events would be forwarded to all subscribers, not just those which are part of the pulsox device.*

It should be clear that there are similarities between brokers and processors, but that processors may take on the actions of publishers and subscribers in order to gain direct access to events which they are to perform intermediate processing on. A complete comparison of brokers and processors is provided in Table 11.1.

Whilst Example 11.3 illustrated how a linear arrangement of nodes could make

	Brokers	**Processors**
May Host Services	Yes	If unconstrained
Constrained	Never	Frequently
Mode	Relay only	Relay or intermediate processing
PDP/PEP	Yes	If unconstrained
May be a Hub for a Bridge	Yes	If unconstrained
May Issue Authorization Requests	No	When performing intermediate processing

Table 11.1: Comparison of brokers and processors in RPPM-IoT

use of intermediate processing to refine the raw sensor data for an actuator, in reality more complex arrangements are likely useful. Through non-linear arrangements an IoT system may provide a more robust handling of events and may support more realistic operational environments. To this end, I return to the healthcare IoT system illustrated in Figure 11.5 of Example 11.1. In this system I can envisage that the simple external display is located at the nurse's station outside of the ward. Each bed sensor identifies whether the ward's corresponding bed is occupied by a patient[13] and the events from these sensors are used to provide status indicators on the nurse's station display. Further, the pulse oximetry device may be used in the ward as part of diagnostics performed by trainee nurses. Events containing the peripheral oxygen saturation values may, therefore, be received by the ward alerting node which is programmed with a threshold which triggers visual alerts on the nurse's station display; in this way supervising nurses may determine (even from outside the ward) if there is a need to provide assistance.

Example 11.4. *Figure 11.8 shows the annotated system graph of the healthcare IoT system illustrated in Figure 11.5. The simple external display subscribes to the bed occupancy topic provided by service s_o hosted by the ward occupancy broker. It also subscribes to the alerting topic provided by service s_a hosted by the ward alerting node and to the peripheral oxygen saturation topic provided by service s_p hosted by the pulsox core. I have already considered the function of the bed sensors (in Example 11.2) and the pulsox device (in Example 11.3). The ward alerting node subscribes to the peripheral oxygen saturation topic provided by service s_p hosted by the pulsox core and publishes alert events if the oxygen saturation values are below the programmed threshold. These alert events are published via the ward alerting node's service s_a. (Note that, as a further example of the flexibility of processor nodes, the ward alerting node gets its input data from a service hosted by another local entity,*

[13]This may simply involve detecting a particular level of pressure, or it may involve a more robust indication using an RFID tag built into the patient's ID band (and thus able to identify the particular individual) or the detection of a heartbeat using a sub-mattress sensor pad.

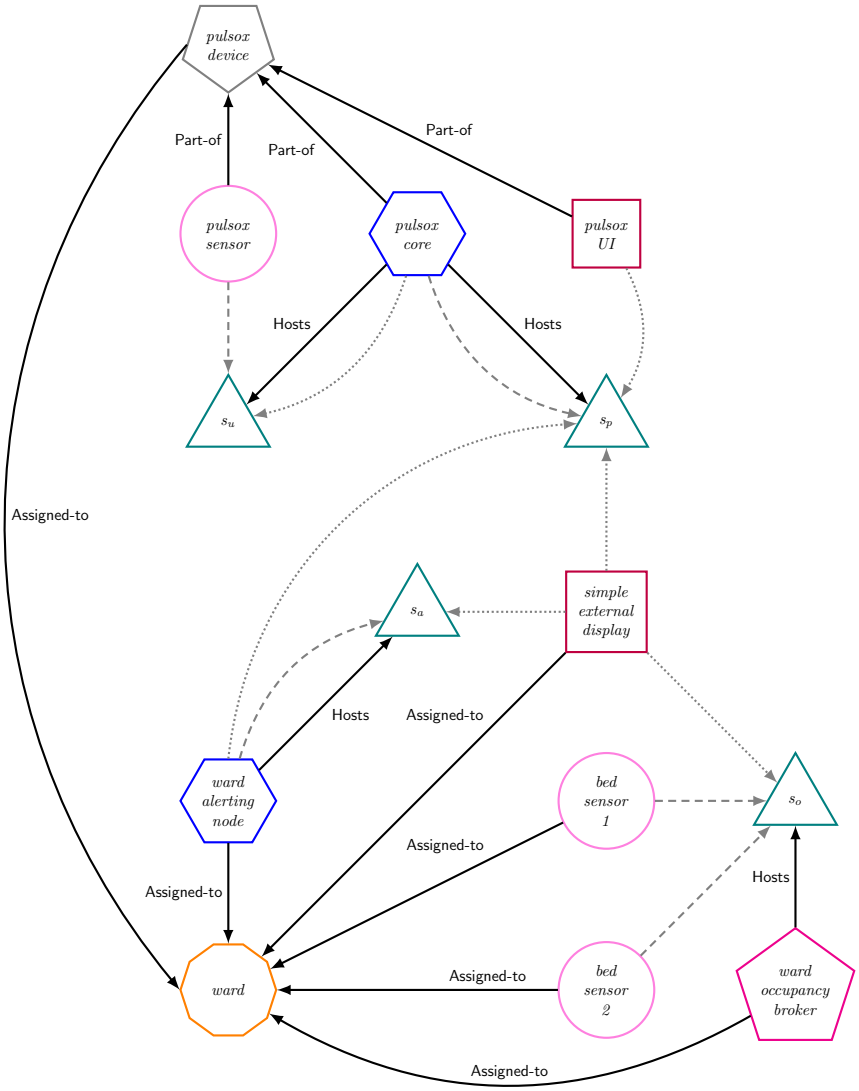


Figure 11.8: IoT healthcare system graph (annotated)

but it publishes its output to a local service. In both cases, the events are forwarded from those services to all registered subscribers.)

11.2.4 Administration

As has been identified, the need for context is particularly relevant to IoT systems as their component devices are expected to operate in and interact with their (changing) environment. In some cases this involves devices moving from one IoT system to another, for example as geographical location or system requirements change. RPPM's design enables it to support such a dynamic system architecture through the addition and deletion of relationships, whether performed by a human

administrator or automatically.

Depending on the IoT system being modelled it may be necessary for some, or all, of the addRelEdge and deleteRelEdge administrative requests to also be issued by $root_a$ (or another entity of type *auto* with appropriate relationships).[14] I can envisage, for example, that in such an application the proximity of a new device to an existing IoT system may be used to trigger the addition of that device to the system's graph, through an addRelEdge request issued and evaluated by $root_a$. Once part of the system graph, subsequent authorization requests may be made within the system by that device. In other IoT systems, such as my healthcare example, users may interact with nodes and, thereby, trigger administrative requests. In these cases I, once again, rely on $root_a$ to be the PDP; however, the requested administrative action will have the user in question as the subject of the request.

As a device moves away from the system, it may be "removed" by the automated deletion of its relationships after some threshold distance and duration.[15] In some cases an entity may lose connectivity and so not trigger any proximity-related processing as it "leaves" the system. In systems where it is appropriate some form of liveness test mechanism may be used and a lack of response to consecutive "heartbeat" requests may be used as the trigger for automated deletion of the entity.

Obviously, should an entity leave an IoT system graph, any relationships it has with other system entities (including caching edges) would be removed from the system graph along with it. Any subscriptions associated with the entity would be removed as part of the regular validation process, which may be triggered immediately to ensure the clean-up is performed promptly.

11.3 Summary

In this chapter I have introduced Internet of Things and presented RPPM-IoT, a variant of the RPPM-PS model tailored to providing publish/subscribe access control between IoT devices in light of the characteristics and constraints affecting such architectures. In particular, the RPPM-IoT model makes use of processors to provide a flexible means of performing intermediate processing between IoT's sensors and actuators. In order to avoid burdening constrained devices in an IoT architecture, I limit unconstrained brokers and processors to performing the roles of policy decision point and policy enforcement point.

[14] This is in addition to the administrative requests to add (and delete) caching edges and audit edges once requests have been evaluated, as identified in Section 10.3.5.

[15] Note that, as per Chapter 8, the deletion of an entity's last relationship edge results in the removal of the entity itself from a system graph.

Chapter 12

Conclusions

In this thesis I have presented the RPPM model, a novel relationship-based access control model designed for general computing applications. I have used a systematic approach, commonly used in the field of access control [3, 49, 58, 144, 164, 165], and presented a series of models:

- RPPM_0 – the basic functional model able to evaluate authorization requests based on relationships between concrete and logical entities.
- RPPM_{1a} – a model containing *request evaluation enhancements* which enable the support of conjunction and graph-based policies, as well as caching.
- RPPM_{1b} – a model containing *policy configuration enhancements* which enable the support of history-based policies and specific policy configurations.
- RPPM_{1c} – a model containing a *targeting enhancement* which enables the evaluation of requests based upon paths between arbitrary entities within the modelled system.
- ARPPM – a model containing *administrative enhancements* which enable the configuration and ongoing administration of model instances.
- Inter-RPPM – a model containing *inter-operation enhancements* which enable multiple systems to be connected and remote requests to be evaluated.
- RPPM_3 – a model which consolidates the features of all of the previous models.
- RPPM-PS – a relationship-based publish/subscribe access control system.
- RPPM-IoT – a relationship-based publish/subscribe access control system tailored to an Internet of Things.

In this chapter I review the contributions of my thesis to the body of knowledge. I first revisit my motivations and confirm that each of these has been met. I then identify limitations of RPPM and highlight opportunities for future work.

12.1 Motivations

In Chapter 3 I detailed existing access control model issues and, thereby, my motivations for designing the RPPM model. In addition, in Chapters 8, 9 and 10 I discussed motivations for specific enhancements to broaden the management and applicability of RPPM in general computing scenarios. I now return to these motivations as part of reviewing my novel contribution to logical access control through the design of RPPM, which (as originally introduced in Section 1.3):

1. Is able to model a wide range of systems, including online social networks which originally motivated relationship-based access control, due to its support for any entity and relationship types required.

2. Can be configured to guarantee that a conclusive authorization decision (approve or deny) can be computed for any request.

3. Can control actions requested by any type of entity on any other type, such that subjects may be users, autonomous entities, automated agents or even inanimate objects if so desired.

4. Abstracts permission assignment away from subjects to principals, reducing the administrative burden and enabling the system graph to be managed in isolation to the policies.

5. Employs flexible policy rules which support the definition of policies covering individual entities, multiple specific entities, all entities of one (or more) types, or every entity in the system.

6. Can define set-based or graph-based policies comprising multiple rules, where each can employ a required and forbidden path of relationships built with regular expression-like operators to support concatenation, disjunction, conjunction, Kleene plus, optional and Kleene star.

7. Is able to cache the result of principal-matching directly within the system graph such that future requests between that subject and object (no matter what action is requested) may be processed far faster.

8. Is able to support useful policy configurations such as history-based access control, separation of duty, binding of duty and Chinese Wall.

9. Can match principals based on paths of relationships between arbitrary entities, thus enabling subgraph patterns to be used to evaluate requests.

10. Can be used to completely administer instances of itself without extra-model authorization.

11. Can evaluate both local and remote requests, such that subjects may interact with objects in the local security domain or a remote security domain to which their system is connected.

12. Can implement a range of access control models not based on relationships, specifically: multi-level security; RBAC; UNIX; and multi-level security's *-property.

13. Can consolidate all of these features to produce a robust model for access control which can be further tailored to popular communication architectures.

List Management In Section 3.2.1 I identified the need for "an intuitive, easily managed access control model" able to replace access control lists. I believe that basing RPPM on relationships between entities offers that intuition. As I indicated in Section 2.4.1, humans commonly employ relationships to comprehend and communicate our understanding of abstract ideas. As a species we view the world based on how we relate to those around us, a fact that has been exploited (with positive and negative repercussions) in the proliferation of social networking services in the last decade (or so). I believe that the specification of security policy in respect of those relationships is intuitive (to entities involved in those systems). What's more, I believe that (as demonstrated by social networks, company organograms and HR systems) we are already capturing and managing these relationships in numerous cases. This is a view held by others too, Barkley *et al.* also believe "relationship information may already be kept as part of the information content associated with the business process" [21]. Leveraging this information for access control enables an easily managed model where the day-to-day administration focuses on the addition and deletion of direct relationships, but where changes to these naturally impact subsequent authorization decisions as defined by the policies.

Role Explosion I next identified the need for "an intuitive, context-aware access control model" able to avoid the issues suffered by role-based access control in re-

lation to role explosion. The fact that RPPM's system model is able to include arbitrary concrete and logical entities enables it to provide the necessary context. What's more, these entity types and their particular inter-relationships may be tailored to the modelled system and, thereby, may be chosen so as to be intuitive and appropriate to that system's particular contextual requirements. The two specific issues raised (wide-acting roles and conflicting roles) have been addressed in examples throughout this thesis. It should be clear from these examples that an RPPM model instance can employ logical entities and specific connecting relationships to clearly distinguish "situations" where the access control policy should grant authorization requests from those in which the policy should not.

Attribute Applicability Whilst it is clear that attribute-based access control also offers great potential, this does not mean that it can achieve everything that relationship-based access control models can. The recent comparison work by Ahmed *et al.* has illustrated that generic models of both types have shown some equivalences, but that other aspects and model variants are incomparable [3]. Ahmed *et al.* separately considered multiple axes of comparison (dynamics and structure), highlighting the complexity of determining equivalences for generic models; determining equivalences for real-world ones is likely to be no less complex given their specific nuances. In their taxonomy, the consolidated RPPM_3 model may be classified as a "node dynamic" ReBAC_{BE} model.[1] Some of the ABAC generic models are believed to be more expressive than such a model, but they require an infinite attribute domain and structured entity attributes. It is unclear at this time the impact such requirements may have on these models. Whilst it is clear that an ABAC model may express policies from relationship-based access control by capturing relations within entity attributes, such an approach is likely to be cumbersome. Enumerating the presence (or not) of all paths of relationships in ABAC using "composite attributes" quickly demands a cap on the depth of such paths and so make this an undesirable option for general computing systems such as those that RPPM_3 can support [3]. The alternative, "attribute composition or chaining" seems more plausible [3], but will still likely present significant challenges from a storage space perspective in large, highly connected systems.

Relationship Context Whilst a number of relationship-based access control models existed prior to RPPM, each had limited applicability due to constraints in their

[1] ReBAC_{BE} models support node types, edge types and edge attributes. RPPM_3 supports each of these, with edge attributes present in the form of the tuple labels associated with caching and auditing edges when working with inter-operation.

support for:

- Types of entity – with several models only supporting users in social networks [41, 44, 82, 85], or only supporting users and their resources [43].
- Types of relationship – with several models only supporting ownership of non-user entities [41, 44, 82, 85].
- Relationship use – with one model not supporting cycles in the relationship graph [51, 52].
- Policy language expressiveness – with limitations on whether negative rules are supported [41, 44, 51, 52, 82, 85] and the lengths of paths of relationships [44, 51, 52].

In contrast, RPPM does not suffer these limitations, as shown by the (updated) comparisons in Table 12.1. RPPM is, therefore, a far more expressive and robust model for relationship-based access control than the other existing models. From its outset RPPM has been designed to support access control in generic computing systems rather than just social networks, where relationship-based access control originated. Further, RPPM is generic enough to implement various non-relationship-based access control models as demonstrated in Sections 4.6.2, 5.2.2 and 6.2.2.

Administration In Chapter 8 I motivated the design of administrative enhancements to RPPM on the need to control changes to model instances, with the underlying goal of avoiding type I (false negative) and type II (false positive) errors during request evaluation. I subsequently identified administration requirements for any administrative RPPM model; I based these requirements on best practice and desirable security and usability properties. The ARPPM model I presented meets each of the identified requirements, thus enabling model instances to be completely self-managed by one or more administrators without extra-model authorization. Whilst several existing relationship-based access control models are able to satisfy three of the five requirements, they both fall short of providing complete administration and so also rely on some extra-model authorization for control of the remaining aspects.

Inter-Operation In Chapter 9 I motivated the design of inter-operation enhancements to RPPM on two key drivers for inter-operation:

- Autonomy – where individual systems have distinct authorities which need to maintain their control over their individual security domains. However, in-

	Social Network Model [44]	**Semantic Web [43]**	**ReBAC [41, 82, 85]**	**U2R [51, 52]**	**RPPM$_3$**
Graph Entities	Users	Users and resources	Users	Users, objects, policies and sessions	$\star$
Graph Relations	$\star$	$\star$	$\star$	$\star$	$\star$
Graph Limitations	No resources	None	No resources	No cycles	None
Resource Relations	Ownership	$\star$	Ownership	$\star$	$\star$
Path Evaluation Between (Types)	Users and Users	Users and Resources	Users and Users	Users and $\star$	$\star$ and $\star$
Path Evaluation Between (Entities)	Requestor and resource owner	Subject and object	Requester and resource owner	A pair from requester, target and controlling user	One or more pairs of arbitrary entities
Multiple Distinct Relations in Paths	No	Yes	Yes	Yes	Yes
Policy Language	Conditions over relation, max length and min strength	Atoms over type of resource, direct relation, and same or different entity	Propositions over direct relation, variables and nominals	Regular expressions over relations	Regular expressions over relations
Paths in Rules	Positive conjunction of conditions	Positive conjunction of atoms	Positive and negative, conjunction and disjunction of propositions	Positive and negative, conjunction and disjunction of expressions	Positive and negative, conjunction and disjunction of expressions
Rule Types	Positive only	Positive and negative	Positive only	Positive only	Positive and negative
Rule Limitations	Single relation within exact path	Exact path via pre-determined objects	Defined per resource	Path length limited by hopcount	None
Evaluation Basis	CWM reasoner [29]	SweetRules reasoner [92]	Depth-first search model checker	Depth-first search with limited depth	Regular language emptiness evaluation

Key: The wild card character $\star$ is used in this table to indicate "anything" or "all".

Table 12.1: Relationship-based access control model comparisons with RPPM$_3$

creasing connectivity results in increasing demand for authorization over remote actions on remote objects.

- Scalability – where large systems are likely to be impacted by increasing time complexity of request evaluation, correlated with the number of nodes in the system graph. The subdivision of such a large system both reduces the load of evaluation in any one subgraph and (in the average case) reduces the percentage

of the total entity space which must be considered when matching paths to principals.

The bridged system group and bridges of Inter-RPPM enable distinct autonomous systems to be brought together or large systems to be subdivided. The remote requests (originating and incoming) then enable multiple distinct policies, from traversed security domains, to contribute to the result of the ultimate authorization decision.

Publish/subscribe In Chapter 10 I motivated the consolidation of RPPM's features into $RPPM_3$ on demonstrating the manner in which the various enhancements may coexist.[2] I, similarly, motivated $RPPM_3$'s application to the publish/subscribe communication architecture (in the form of RPPM-PS) as part of demonstrating how RPPM may be used to satisfy the requirements of wider data sharing approaches. I chose publish/subscribe as it is a popular approach to information dissemination in large-scale, distributed applications [20, 68, 73]. Whilst other access control models have previously been considered for the publish/subscribe approach (as discussed in Section 10.1.1), until RPPM relationship-based access control was unable to be broadly applied to generic computing systems and architectures such as this. The combination of $RPPM_3$'s various capabilities enables it to easily support a publish/subscribe approach to information sharing, allowing subscribers to register their interest in (potentially newly added) topics with (potentially remote) brokers which receive events from (potentially remote) publishers.

Internet of Things Finally, in Chapter 11 I motivated the need to tailor RPPM-PS for application in an Internet of Things on the specific characteristics and constraints of this topical and important field. There is growing research and media interests in Internet of Things, with the security of IoT devices and IoT networks regularly called into question [87, 107, 146, 177, 188]. In fact numerous researchers have identified some aspect of security as a key area for consideration [93, 148, 186]. The introduction of a small number of refinements to RPPM-PS enables the RPPM-IoT model to be tailored to an Internet of Things, in particular giving consideration to the need for intermediate processing between sensors and actuators, and not burdening constrained devices which are typical in IoT systems.

[2] In reality many of the features had already been shown to coexist through my systematic approach to model development.

12.2 Future Work

The introduction of RPPM has broadened the applicability of relationship-based access control in the last four years and has, thereby, contributed to it being an active area of research in the last decade. Like Ahmed *et al.*, I believe that relationship-based access control has "considerable applications in industry" and that it is "anticipated to continue being important for the foreseeable future" (alongside ABAC) [3]. Moreover, I believe that RPPM is currently the most broadly applicable relationship-based access control model, with the most expressive policy language and the widest range of coexisting features. That said, there are potential limitations which highlight areas requiring further investigation as well as many other areas offering opportunities for novel future work. I discuss a range of limitations and opportunities here with the hope of stimulating further discussion in the research community.

12.2.1 Potential Limitations Requiring Further Investigation

Complexity Whilst Rizvi *et al.* have implemented the ReBAC model in an open source medical records system [157], work is required to investigate the requirements of large systems (from various domains) on the application of relationship-based access control. Specifically, it is unclear how many relationships may actually be necessary to usefully model various systems, including, for example, a modern computer system, a multi-national company, or a hospital. (The computer on which I am writing this thesis contains 367,717 files spread amongst 67,487 folders, which leads to a graph of 435,203 edges in the trivial case of a tree utilising solely a Contained-in relationship to model the filesystem hierarchy.) It is also unclear whether the algorithms employed within RPPM will be efficient at the required scales, and whether the associated storage requirements will be sustainable. Modelling of "representative" systems and an analysis of the storage and processing demands that result would be helpful to alleviate concerns which some implementers may have. Such results may identify a need for further optimisations, alternative approaches, or may simply highlight the necessity of utilising those optimisations already introduced (such as caching and inter-operation). It is worth noting that this is not a novel, intractable problem; the Facebook social network was reported to have 1.39 billion active users as of December 2014 with more than 400 billion edges [53]. Whilst processing on graphs of this size requires significant computing power which is excessive for the operation of a protection system, the scale of Facebook's social network is very many orders of magnitude greater than that required for access control (particularly when inter-operation is taken into consideration).

"Before the Fact" Audit As introduced in Section 2.1.3, the access matrix offers simple "before the fact" auditing of policy from the perspectives of both the subject and object. Answers to the questions "what can a specific subject do?" and "which subjects can do things to a specific object?" can be easily determined by assessing the access attributes in a particular row or column. Performing such an audit of policy in relationship-based access control is more complicated as, at its crudest, it requires searching for paths of relationships between all entities and a specific entity (with the direction in or out of the specific entity identifying it as the object or subject). Bennett *et al.* have performed some initial research into the automated analysis of relationship-based policies, having translated their OSN-focused model into unified modelling language (UML) [27]. As they indicate, a "lot of work remains to be done" as part of investigating potential approaches to "before the fact" auditing, particularly with the goal of aiding administrators in understanding the repercussions that specific administrative changes will have so that they may avoid misconfigurations. In the case of RPPM, my caching edges may be a useful aid to such analysis; at the very least the results of any automated analysis may be recorded in caching edges to optimise subsequent request evaluation. However, at least some benefit may be gained by abstracting the subject-focused audit questions to "what can a specific principal do?" and "which principals can do things to a specific object?".

12.2.2 Novel Opportunities

Visualisation Tools Whether as part of "before the fact" auditing or of aiding administration more generally, I believe there is a need for graph and policy visualisation tools tailored to the administration of relationship-based access control. Graph visualisation is a topic which has received previous attention, both generally and with respect to access control [100, 175]. However, work is required to consider the information useful to administrators during the specification and auditing of relationship-based policies, and the potential approaches to presenting this in conjunction with the critical relationship information contained in the system graph. It may be that a suitable approach involves overlaying the graph with information relevant to a selected policy or policy component; highlighting matching paths or entities at the ends of matching paths may be particularly helpful, if achievable. Certainly highlighting the different types of edges (caching, auditing, relationship), types of entities and types of relationships will likely be useful to administrators.

Caching Strategy Evaluation Whilst the introduction of caching edges in the RPPM system graph has a clear benefit to the evaluation of future requests between the same subject and object (as discussed in Section 5.1.3), practical investigation is required to validate and tune the approach. Particularly, there is a need to determine the likely performance improvement which may result for "representative" systems through both direct assessment and an analysis of the proportion of requests which may benefit during "normal" operation (i.e., which are between the same subject and object).[3] Practical investigation of purging and pre-emptive caching strategies is also necessary to understand and propose optimal mechanisms for specific application domains or the general case (within certain constraints). The storage and time complexity of request evaluation will, obviously, be important considerations for the tuning of any such strategies.

Stateful Entities Whilst RPPM supports history-based policies through the inclusion of audit edges (as discussed in Section 6.1), I believe there is significant opportunity to extend the model through the inclusion of state information in entities (most likely through the introduction of entity attributes). Cheng *et al.* previously introduced node attributes into their OSN-focused relationship-based access control model which employs user-to-user relationships [50]. However, I believe that such attributes may be particularly useful in tailoring relationship-based access control for workflow authorization. Whilst Khan and Fong consider a workflow specification as a graph, with tasks as nodes and constraints as edges [114], I believe that tasks are comparable to RPPM's actions and that the constraints are comparable to a pair of principal-matching rule targets. I, instead, envisage that entities may have an associated state attribute, and that the entity conditions within extended path conditions may themselves be extended to identify an endpoint for a path condition by state (in addition to the existing variables and entity labels of Section 7.1.1). In this way, an action may be authorized when participating entities are in particular states, and part of the result of that authorization may be a change in one or more entities' state.

[3] Obviously which "representative" systems are evaluated and what constitutes "normal" operation will depend on the specific application domain of interest.

Bibliography

[1] Martín Abadi and Cédric Fournet. Access control based on execution history. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2003, San Diego, California, USA*. The Internet Society, 2003.

[2] Gregory D. Abowd, Anind K. Dey, Peter J. Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In Hans-Werner Gellersen, editor, *Handheld and Ubiquitous Computing, First International Symposium, HUC'99, Karlsruhe, Germany, September 27-29, 1999, Proceedings*, volume 1707 of *Lecture Notes in Computer Science*, pages 304–307. Springer, 1999.

[3] Tahmina Ahmed, Ravi S. Sandhu, and Jaehong Park. Classifying and comparing attribute-based and relationship-based access control. In Gail-Joon Ahn, Alexander Pretschner, and Gabriel Ghinita, editors, *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, CODASPY 2017, Scottsdale, AZ, USA, March 22-24, 2017*, pages 59–70. ACM, 2017.

[4] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1986.

[5] Bader Ali, Wilfred Villegas, and Muthucumaru Maheswaran. A trust based approach for protecting user data in social networks. In Kelly A. Lyons and Christian Couturier, editors, *CASCON*, pages 288–293. IBM, 2007.

[6] James P. Anderson. Computer security technology planning study. Volume II. Technical Report ESD-TR-73-51 Volume II, USAF Electronic Systems Division, October 1972.

[7] James P. Anderson. Information security in a multi-user computer environment. *Advances in Computers*, 12:1–36, 1972.

[8] Per-Erik Andersson. Directory enabled networks, DEN. Master of science thesis, KTH Royal Institute of Technology, February 2000.

[9] Android Open Source Project. Security-Enhanced Linux in Android. Online. https://source.android.com/security/selinux/index.html Accessed: February 26, 2017.

[10] Android Open Source Project. System and kernel security. Online. https://source.android.com/security/overview/kernel-security.html Accessed: February 26, 2017.

[11] Apple Inc. Authentication, authorization, and permissions guide - Understanding permissions. Online, January 2013. https://developer.apple.com/library/content/documentation/Security/Conceptual/AuthenticationAndAuthorizationGuide/Permissions/Permissions.html Accessed: February 26, 2017.

[12] Apple Inc. Kernel programming guide - Mach overview. Online, August 2013. https://developer.apple.com/library/content/documentation/Darwin/Conceptual/KernelProgramming/Mach/Mach.html Accessed: February 26, 2017.

[13] Apple Inc. File system programming guide - File system details. Online, September 2016. https://developer.apple.com/library/content/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemDetails/FileSystemDetails.html Accessed: February 26, 2017.

[14] Alessandro Artale, Bruno Crispo, Fausto Giunchiglia, Fatih Turkmen, and Rui Zhang. Reasoning about relation based access control. In Yang Xiang, Pierangela Samarati, Jiankun Hu, Wanlei Zhou, and Ahmad-Reza Sadeghi, editors, *Fourth International Conference on Network and System Security, NSS 2010, Melbourne, Victoria, Australia, September 1-3, 2010*, pages 231–238. IEEE Computer Society, 2010.

[15] Mohsen Hallaj Asghar. RFID and EPC as key technology on Internet of Things (IoT). *International Journal of Computer Science and Technology*, 6(1), 2015.

[16] Kevin Ashton. That 'Internet of Things' thing. Online, June 2009. http://www.rfidjournal.com/articles/view?4986 Accessed: March 31, 2017.

[17] Atlassian. Manage users and groups. Online. `https://confluence.atlassian.com/cloud/manage-users-and-groups-744721623.html` Accessed: February 27, 2017.

[18] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The Internet of Things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[19] Jean Bacon, David M. Eyers, Ken Moody, and Lauri I. W. Pesonen. Securing publish/subscribe for multi-domain systems. In Gustavo Alonso, editor, *Middleware 2005, ACM/IFIP/USENIX, 6th International Middleware Conference, Grenoble, France, November 28 - December 2, 2005, Proceedings*, volume 3790 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2005.

[20] Jean Bacon, David M. Eyers, Jatinder Singh, and Peter R. Pietzuch. Access control in publish/subscribe systems. In Roberto Baldoni, editor, *Proceedings of the Second International Conference on Distributed Event-Based Systems, DEBS 2008, Rome, Italy, July 1-4, 2008*, volume 332 of *ACM International Conference Proceeding Series*, pages 23–34. ACM, 2008.

[21] John F. Barkley, Konstantin Beznosov, and Jinny Uppal. Supporting relationships in access control using role based access control. In Charles E. Youman and Sylvia L. Osborn, editors, *Proceedings of the Fourth ACM Workshop on Role-Based Access Control, RBAC 1999, Fairfax, VA, USA, October 28-29, 1999*, pages 55–65. ACM, 1999.

[22] Moritz Y. Becker, Cédric Fournet, and Andrew D. Gordon. SecPAL: Design and semantics of a decentralized authorization language. *Journal of Computer Security*, 18(4):619–665, 2010.

[23] D. Elliott Bell and Leonard J. La Padula. Secure computer systems: A mathematical model. Technical Report MTR-2547 Volume II, The MITRE Corporation, Bedford, Massachusetts, November 1973.

[24] D. Elliott Bell and Leonard J. La Padula. Secure computer systems: Mathematical foundations. Technical Report MTR-2547, The MITRE Corporation, Bedford, Massachusetts, November 1973.

[25] D. Elliott Bell and Leonard J. La Padula. Secure computer system: Unified exposition and Multics interpretation. Technical Report ESD-TR-75-306, USAF Electronic Systems Division, March 1976.

[26] András Belokosztolszki, David M. Eyers, Peter R. Pietzuch, Jean Bacon, and Ken Moody. Role-based access control for publish/subscribe middleware architectures. In Hans-Arno Jacobsen, editor, *Proceedings of the 2nd International Workshop on Distributed Event-Based Systems, DEBS 2003, Sunday, June 8th, 2003, San Diego, California, USA (in conjunction with SIGMOD/PODS)*. ACM, 2003.

[27] Phillipa Bennett, Indrakshi Ray, and Robert B. France. Analysis of a relationship based access control model. In Jake Yue Chen, Mohammed J. Zaki, Tamer Kahveci, Saeed Salem, and Mehmet Koyutürk, editors, *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering, Yokohama, Japan, July 13-15, 2015*, pages 1–8. ACM, 2015.

[28] Jorge Bernal Bernabé, José Luis Hernández Ramos, and Antonio F. Gómez-Skarmeta. TACIoT: Multidimensional trust-aware access control system for the Internet of Things. *Soft Computing*, 20(5):1763–1779, 2016.

[29] Tim Berners-Lee. Cwm. Online, October 2009. `https://www.w3.org/2000/10/swap/doc/cwm.html` Accessed: March 9, 2017.

[30] Elisa Bertino, Piero A. Bonatti, and Elena Ferrari. TRBAC: A temporal role-based access control model. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):191–233, 2001.

[31] Elisa Bertino, Elena Ferrari, and Vijayalakshmi Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Transactions on Information and System Security (TISSEC)*, 2(1):65–104, 1999.

[32] Elisa Bertino, Ravi Sandhu, and Alexander Pretschner, editors. *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy, CODASPY 2016, New Orleans, LA, USA, March 9-11, 2016*. ACM, 2016.

[33] Rafae Bhatti, Elisa Bertino, and Arif Ghafoor. A trust-based context-aware access control model for web-services. In *Proceedings of the IEEE International Conference on Web Services (ICWS'04), June 6-9, 2004, San Diego, California, USA*, pages 184–191. IEEE Computer Society, 2004.

[34] Rafae Bhatti, Arif Ghafoor, Elisa Bertino, and James Joshi. X-GTRBAC: An XML-based policy specification framework and architecture for enterprise-wide access control. *ACM Transactions on Information and System Security (TISSEC)*, 8(2):187–227, 2005.

[35] Rafae Bhatti, Basit Shafiq, Elisa Bertino, Arif Ghafoor, and James Joshi. X-GTRBAC Admin: A decentralized administration model for enterprise-wide access control. *ACM Transactions on Information and System Security (TISSEC)*, 8(4):388–423, 2005.

[36] Kenneth J. Biba. Integrity considerations for secure computer systems. Technical Report ESD-TR-76-372, USAF Electronic Systems Division, April 1977.

[37] Kevin Borders, Xin Zhao, and Atul Prakash. CPOL: High-performance policy evaluation. In Vijay Atluri, Catherine Meadows, and Ari Juels, editors, *ACM Conference on Computer and Communications Security*, pages 147–157. ACM, 2005.

[38] Carsten Bormann, Mehmet Ersue, and Ari Keranen. *Terminology for Constrained-Node Networks*. Internet Engineering Task Force (IETF), May 2014. Request for Comments: 7228.

[39] Reinhardt A. Botha and Jan H. P. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Systems Journal*, 40(3):666–682, 2001.

[40] David F. C. Brewer and Michael J. Nash. The Chinese Wall security policy. In *IEEE Symposium on Security and Privacy*, pages 206–214. IEEE Computer Society, 1989.

[41] Glenn Bruns, Philip W. L. Fong, Ida Siahaan, and Michael Huth. Relationship-based access control: Its expression and enforcement through hybrid logic. In Elisa Bertino and Ravi S. Sandhu, editors, *Second ACM Conference on Data and Application Security and Privacy, CODASPY 2012, San Antonio, TX, USA, February 7-9, 2012*, pages 117–124. ACM, 2012.

[42] Barbara Carminati, Pietro Colombo, Elena Ferrari, and Gokhan Sagirlar. Enhancing user control on personal data usage in Internet of Things ecosystems. In Jia Zhang, John A. Miller, and Xiaofei Xu, editors, *IEEE International Conference on Services Computing, SCC 2016, San Francisco, CA, USA, June 27 - July 2, 2016*, pages 291–298. IEEE Computer Society, 2016.

[43] Barbara Carminati, Elena Ferrari, Raymond Heatherly, Murat Kantarcioglu, and Bhavani M. Thuraisingham. A semantic web based framework for social network access control. In Barbara Carminati and James Joshi, editors, *SACMAT 2009, 14th ACM Symposium on Access Control Models and Technologies, Stresa, Italy, June 3-5, 2009, Proceedings*, pages 177–186. ACM, 2009.

[44] Barbara Carminati, Elena Ferrari, and Andrea Perego. Rule-based access control for social networks. In Robert Meersman, Zahir Tari, and Pilar Herrero, editors, *On the Move to Meaningful Internet Systems 2006: IFIP WG 2.12 and WG 12.4 International Workshop on Web Semantics (SWWS), Montpellier, France, October 29 - November 3, 2006. Proceedings, Part II*, volume 4278 of *Lecture Notes in Computer Science*, pages 1734–1744. Springer, 2006.

[45] Barbara Carminati, Elena Ferrari, and Andrea Perego. Enforcing access control in web-based social networks. *ACM Transactions on Information and System Security (TISSEC)*, 13(1), 2009.

[46] CentOS. SELinux. Online, February 2017. `https://wiki.centos.org/HowTos/SELinux` Accessed: February 26, 2017.

[47] Centre for the Protection of National Infrastructure (CPNI). Cyber security. Online. `https://www.cpni.gov.uk/cyber-security` Accessed: April 20, 2017.

[48] Song-hwa Chae and Wonil Kim. Semantic representation of RTBAC: Relationship-based access control model. In Kevin Chen-Chuan Chang, Wei Wang, Lei Chen, Clarence A. Ellis, Ching-Hsien Hsu, Ah Chung Tsoi, and Haixun Wang, editors, *Advances in Web and Network Technologies, and Information Management, APWeb/WAIM 2007 International Workshops: DBMAN 2007, WebETrends 2007, PAIS 2007 and ASWAN 2007, Huang Shan, China, June 16-18, 2007, Proceedings*, volume 4537 of *Lecture Notes in Computer Science*, pages 554–563. Springer, 2007.

[49] Yuan Cheng, Khalid Zaman Bijon, and Ravi S. Sandhu. Extended ReBAC administrative models with cascading revocation and provenance support. In Wang et al. [185], pages 161–170.

[50] Yuan Cheng, Jaehong Park, and Ravi Sandhu. Attribute-aware relationship-based access control for online social networks. In Vijay Atluri and Günther Pernul, editors, *Data and Applications Security and Privacy XXVIII - 28th Annual IFIP WG 11.3 Working Conference, DBSec 2014, Vienna, Austria, July 14-16, 2014. Proceedings*, volume 8566 of *Lecture Notes in Computer Science*, pages 292–306. Springer, 2014.

[51] Yuan Cheng, Jaehong Park, and Ravi S. Sandhu. Relationship-based access control for online social networks: Beyond user-to-user relationships. In *2012 International Conference on Privacy, Security, Risk and Trust, PASSAT 2012,*

and 2012 International Confernece on Social Computing, SocialCom 2012, Amsterdam, Netherlands, September 3-5, 2012, pages 646–655. IEEE, 2012.

[52] Yuan Cheng, Jaehong Park, and Ravi S. Sandhu. A user-to-user relationship-based access control model for online social networks. In Nora Cuppens-Boulahia, Frédéric Cuppens, and Joaquín García-Alfaro, editors, *DBSec*, volume 7371 of *Lecture Notes in Computer Science*, pages 8–24. Springer, 2012.

[53] Avery Ching, Sergey Edunov, Maja Kabiljo, Dionysios Logothetis, and Sambavi Muthukrishnan. One trillion edges: Graph processing at Facebook-scale. *Proceedings of the Very Large Database VLDB Endowment*, 8(12):1804–1815, 2015.

[54] David D. Clark and David R. Wilson. A comparison of commercial and military computer security policies. In *IEEE Symposium on Security and Privacy*, pages 184–195. IEEE Computer Society, 1987.

[55] Louis Cozolino. *The Neuroscience of Human Relationships: Attachment and the Developing Social Brain*. WW Norton & Company, 2014. Norton Series on Interpersonal Neurobiology.

[56] Jason Crampton. A reference monitor for workflow systems with constrained task execution. In Ferrari and Ahn [81], pages 38–47.

[57] Jason Crampton. Why we should take a second look at access control in Unix. In *13th Nordic Workshop on Secure IT Systems, NordSec 2008, Kongens Lyngby Oct 9-10, 2008. Proceedings*. Technical University of Denmark, DTU Informatics, 2008.

[58] Jason Crampton and George Loizou. Administrative scope: A foundation for role-based administrative models. *ACM Transactions on Information and System Security (TISSEC)*, 6(2):201–231, 2003.

[59] Jason Crampton and Charles Morisset. PTaCL: A language for attribute-based access control in open systems. In Pierpaolo Degano and Joshua D. Guttman, editors, *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings*, volume 7215 of *Lecture Notes in Computer Science*, pages 390–409. Springer, 2012.

[60] Jason Crampton and James Sellwood. Caching and auditing in the RPPM model. In Sjouke Mauw and Christian Damsgaard Jensen, editors, *Security and Trust Management - 10th International Workshop, STM 2014, Wroclaw, Poland, September 10-11, 2014. Proceedings*, volume 8743 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 2014.

[61] Jason Crampton and James Sellwood. Caching and auditing in the RPPM model. *CoRR*, abs/1407.7841, 2014. Longer version including proofs.

[62] Jason Crampton and James Sellwood. Path conditions and principal matching: A new approach to access control. In Sylvia L. Osborn, Mahesh V. Tripunitara, and Ian Molloy, editors, *19th ACM Symposium on Access Control Models and Technologies, SACMAT '14, London, ON, Canada - June 25 - 27, 2014*, pages 187–198. ACM, 2014.

[63] Jason Crampton and James Sellwood. Relationships, paths and principal matching: A new approach to access control. *CoRR*, abs/1505.07945, 2015.

[64] Jason Crampton and James Sellwood. ARPPM: Administration in the RPPM model. In Bertino et al. [32], pages 219–230.

[65] Jason Crampton and James Sellwood. Inter-ReBAC: Inter-operation of relationship-based access control model instances. In Silvio Ranise and Vipin Swarup, editors, *Data and Applications Security and Privacy XXX - 30th Annual IFIP WG 11.3 Conference, DBSec 2016, Trento, Italy, July 18-20, 2016. Proceedings*, volume 9766 of *Lecture Notes in Computer Science*, pages 96–105. Springer, 2016.

[66] Richard E. Creps. A methodology for defining application-specific security requirements for C3 systems. In *Conference Record for the 1989 IEEE Military Communications Conference (MILCOM)*, October 1989.

[67] Maria Luisa Damiani, Elisa Bertino, Barbara Catania, and Paolo Perlasca. GEO-RBAC: A spatially aware RBAC. *ACM Transactions on Information and System Security (TISSEC)*, 10(1), 2007.

[68] Alan J. Demers, Johannes Gehrke, Mingsheng Hong, Mirek Riedewald, and Walker M. White. Towards expressive publish/subscribe systems. In Yannis E. Ioannidis, Marc H. Scholl, Joachim W. Schmidt, Florian Matthes, Michael Hatzopoulos, Klemens Böhm, Alfons Kemper, Torsten Grust, and Christian Böhm, editors, *Advances in Database Technology - EDBT 2006, 10th International Conference on Extending Database Technology, Munich, Germany,*

March 26-31, 2006, Proceedings, volume 3896 of *Lecture Notes in Computer Science*, pages 627–644. Springer, 2006.

[69] Department of Defense. *Department of Defense Trusted Computer System Evaluation Criteria*, December 1985. DoD 5200.28-STD.

[70] Arthur Conan Doyle. *The Adventures of Sherlock Holmes*. George Newnes, Limited, 1901.

[71] Dropbox. Team folders: an overview. Online. `https://www.dropbox.com/help/986?path=dropbox_business` Accessed: February 27, 2017.

[72] Guy Edjlali, Anurag Acharya, and Vipin Chaudhary. History-based access control for mobile code. In Jan Vitek and Christian Damsgaard Jensen, editors, *Secure Internet Programming*, volume 1603 of *Lecture Notes in Computer Science*, pages 413–431. Springer, 1999.

[73] Patrick Th. Eugster, Pascal Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35(2):114–131, 2003.

[74] Fedora. Working with SELinux. Online. `https://docs.fedoraproject.org/en-US/Fedora/11/html/Security-Enhanced_Linux/chap-Security-Enhanced_Linux-Working_with_SELinux.html` Accessed: February 26, 2017.

[75] Adrienne Porter Felt, Erika Chin, Steve Hanna, Dawn Song, and David Wagner. Android permissions demystified. In Yan Chen, George Danezis, and Vitaly Shmatikov, editors, *Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, Illinois, USA, October 17-21, 2011*, pages 627–638. ACM, 2011.

[76] David F. Ferraiolo, John F. Barkley, and D. Richard Kuhn. A role-based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Securiry (TISSEC)*, 2(1):34–64, 1999.

[77] David F. Ferraiolo, Janet Cugini, and D. Richard Kuhn. Role-based access control (RBAC): Features and motivations. In *11th Annual Computer Security Applications Conference (ACSAC 1995), 11-15 December 1995, Los Alamitos, CA, USA*, pages 241–248. IEEE Computer Society, 1995.

[78] David F. Ferraiolo, Ramaswamy Chandramouli Vincent C. Hu, and D. Richard Kuhn. *A Comparison of Attribute Based Access Control (ABAC) Standards for Data Service Applications*. National Institute of Standards and Technology (NIST), October 2016. NIST Special Publication 800-178.

[79] David F. Ferraiolo and D. Richard Kuhn. Role-based access controls. In *Proceedings of the Fifteenth National Computer Security Conference*. National Institute of Standards and Technology (NIST), 1992.

[80] David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[81] Elena Ferrari and Gail-Joon Ahn, editors. *SACMAT 2005, 10th ACM Symposium on Access Control Models and Technologies, Stockholm, Sweden, June 1-3, 2005, Proceedings*. ACM, 2005.

[82] Philip W. L. Fong. Relationship-based access control: protection model and policy language. In Ravi S. Sandhu and Elisa Bertino, editors, *First ACM Conference on Data and Application Security and Privacy, CODASPY 2011, San Antonio, TX, USA, February 21-23, 2011, Proceedings*, pages 191–202. ACM, 2011.

[83] Philip W. L. Fong, Mohd M. Anwar, and Zhen Zhao. A privacy preservation model for Facebook-style social network systems. In Michael Backes and Peng Ning, editors, *ESORICS*, volume 5789 of *Lecture Notes in Computer Science*, pages 303–320. Springer, 2009.

[84] Philip W. L. Fong, Pooya Mehregan, and Ram Krishnan. Relational abstraction in community-based secure collaboration. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 585–598. ACM, 2013.

[85] Philip W. L. Fong and Ida Siahaan. Relationship-based access control policies and their policy languages. In Ruth Breu, Jason Crampton, and Jorge Lobo, editors, *16th ACM Symposium on Access Control Models and Technologies, SACMAT 2011, Innsbruck, Austria, June 15-17, 2011, Proceedings*, pages 51–60. ACM, 2011.

[86] FreeBSD. Mandatory access control. Online. https://www.freebsd.org/doc/handbook/mac.html Accessed: February 26, 2017.

[87] Gartner. Internet of Things. Online. http://www.gartner.com/technology/research/internet-of-things/?cm_sp=itg-_-hub-_-iot Accessed: April 6, 2017.

[88] Luigi Giuri and Pietro Iglio. Role templates for content-based access control. In Charles E. Youman, Edward J. Coyne, and Trent Jaeger, editors, *Proceedings of the Second Workshop on Role-Based Access Control, RBAC 1997, Fairfax, VA, USA, November 6-7, 1997*, pages 153–159. ACM, 1997.

[89] Virgil D. Gligor, Serban I. Gavrila, and David F. Ferraiolo. On the formal definition of separation-of-duty policies and their composition. In *IEEE Symposium on Security and Privacy*, pages 172–183. IEEE Computer Society, 1998.

[90] Li Gong and Xiaolei Qian. Computational issues in secure interoperation. *IEEE Transactions on Software Engineering (TSE)*, 22(1):43–52, 1996.

[91] HM Government. National cyber security strategy 2016–2021. Technical report, November 2016.

[92] Benjamin Grosof and Chitro Neogy. SweetRules: Tools for semantic web rules and ontologies, including translation, inferenceing, analysis, and authoring. Online, October 2005. http://sweetrules.projects.semwebcentral.org/ Accessed: March 9, 2017.

[93] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.

[94] Yuri Gurevich and Itay Neeman. DKAL: Distributed-knowledge authorization language. In *Proceedings of the 21st IEEE Computer Security Foundations Symposium, CSF 2008, Pittsburgh, Pennsylvania, 23-25 June 2008*, pages 149–162. IEEE Computer Society, 2008.

[95] Sergio Gusmeroli, Salvatore Piccione, and Domenico Rotondi. A capability-based security approach to manage access control in the Internet of Things. *Mathematical and Computer Modelling*, 58(5-6):1189–1205, 2013.

[96] Michael A. Harrison, Walter L. Ruzzo, and Jeffrey D. Ullman. Protection in operating systems. *Communications of the ACM*, 19(8):461–471, 1976.

[97] Klaus Hartke. *Observing Resources in the Constrained Application Protocol (CoAP)*. Internet Engineering Task Force (IETF), September 2015. Request for Comments: 7641.

[98] Red Hat. Working with SELinux. Online. `https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/chap-Security-Enhanced_Linux-Working_with_SELinux.html` Accessed: February 26, 2017.

[99] Richard Hayton. OASIS: An open architecture for secure interworking services. Technical Report UCAM-CL-TR-399, University of Cambridge, Computer Laboratory, June 1996.

[100] Ivan Herman, Guy Melançon, and M. Scott Marshall. Graph visualization and navigation in information visualization: A survey. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):24–43, 2000.

[101] Burkhard Hilchenbach. Observations on the real-world implementation of role-based access control. In *20th National Information Systems Security Conference*, pages 341–352. National Institute of Standards and Technology (NIST), 1997.

[102] James A. Hoagland, Raju Pandey, and Karl N. Levitt. Security policy specification using a graphical approach. *CoRR*, cs.CR/9809124, 1998.

[103] Hongxin Hu and Gail-Joon Ahn. Multiparty authorization framework for data sharing in online social networks. In Yingjiu Li, editor, *Data and Applications Security and Privacy XXV - 25th Annual IFIP WG 11.3 Conference, DBSec 2011, Richmond, VA, USA, July 11-13, 2011. Proceedings*, volume 6818 of *Lecture Notes in Computer Science*, pages 29–43. Springer, 2011.

[104] Hongxin Hu, Gail-Joon Ahn, and Jan Jorgensen. Multiparty access control for online social networks: Model and mechanisms. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 25(7):1614–1627, 2013.

[105] Vincent C. Hu, David F. Ferraiolo, D. Richard Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. *Guide to Attribute Based Access Control (ABAC) Definition and Considerations*. National Institute of Standards and Technology (NIST), January 2014. NIST Special Publication 800-162.

[106] Urs Hunkeler, Hong Linh Truong, and Andy Stanford-Clark. MQTT-S - A publish/subscribe protocol for wireless sensor networks. In Sunghyun Choi, Jim Kurose, and Krithi Ramamritham, editors, *Proceedings of the Third International Conference on COMmunication System softWAre and MiddlewaRE (COMSWARE 2008), January 5-10, 2008, Bangalore, India*, pages 791–798. IEEE, 2008.

[107] IEEE. IEEE Internet of Things Journal. Online. `http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=6488907` Accessed: April 6, 2017.

[108] InterNational Committee for Information Technology Standards (INCITS). *American National Standard for Information Technology - Role Based Access Control*. American National Standards Institute, February 2004. ANSI INCITS 359-2004.

[109] InterNational Committee for Information Technology Standards (INCITS). *Information Technology - Next Generation Access Control - Functional Architecture (NGAC-FA)*. American National Standards Institute, March 2013. ANSI INCITS 499-2013.

[110] InterNational Committee for Information Technology Standards (INCITS). *Information Technology - Next Generation Access Control - Functional Architecture (NGAC-FA)*. American National Standards Institute, August 2016. Working Draft - INCITS 499-201x.

[111] Mihaela Ion, Giovanni Russello, and Bruno Crispo. Design and implementation of a confidentiality and access control solution for publish/subscribe systems. *Computer Networks*, 56(7):2014–2037, 2012.

[112] William E. Johnston, Srilekha Mudumbai, and Mary R. Thompson. Authorization and attribute certificates for widely distributed access control. In *7th Workshop on Enabling Technologies (WETICE '98), Infrastructure for Collaborative Enterprises, June 17-19, 1998, Palo Alto, CAUSA, Proceedings*, pages 340–345. IEEE Computer Society, 1998.

[113] James Joshi, Elisa Bertino, Usman Latif, and Arif Ghafoor. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(1):4–23, 2005.

[114] Arif Akram Khan and Philip W. L. Fong. Satisfiability and feasibility in a relationship-based workflow authorization model. In Sara Foresti, Moti Yung,

and Fabio Martinelli, editors, *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, volume 7459 of *Lecture Notes in Computer Science*, pages 109–126. Springer, 2012.

[115] Pranab Kini and Konstantin Beznosov. Speculative authorization. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 24(4):814–824, 2013.

[116] Eric D. Knapp and Joel Thomas Langill. *Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems*. Syngress, 2014.

[117] Mathias Kohler, Achim D. Brucker, and Andreas Schaad. ProActive caching: Generating caching heuristics for business process environments. In *Proceedings of the 12th IEEE International Conference on Computational Science and Engineering, CSE 2009, Vancouver, BC, Canada, August 29-31, 2009*, pages 297–304. IEEE Computer Society, 2009.

[118] Mathias Kohler and Robert Fies. ProActive caching - A framework for performance optimized access control evaluations. In *POLICY 2009, IEEE International Symposium on Policies for Distributed Systems and Networks, London, UK, 20-22 July 2009*, pages 92–94. IEEE Computer Society, 2009.

[119] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage. Experimental security analysis of a modern automobile. In *31st IEEE Symposium on Security and Privacy, S&P 2010, 16-19 May 2010, Berleley/Oakland, California, USA*, pages 447–462. IEEE Computer Society, 2010.

[120] Sebastian Ryszard Kruk, Slawomir Grzonkowski, Adam Gzella, Tomasz Woroniecki, and Hee-Chul Choi. D-FOAF: Distributed identity management with access rights delegation. In Riichiro Mizoguchi, Zhongzhi Shi, and Fausto Giunchiglia, editors, *The Semantic Web - ASWC 2006, First Asian Semantic Web Conference, Beijing, China, September 3-7, 2006, Proceedings*, volume 4185 of *Lecture Notes in Computer Science*, pages 140–154. Springer, 2006.

[121] Karl Krukow, Mogens Nielsen, and Vladimiro Sassone. A logical framework for history-based access control and reputation systems. *Journal of Computer Security*, 16(1):63–101, 2008.

[122] S.-Y. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223, 1964.

[123] Butler W. Lampson. Protection. In *Proceedings of the Fifth Princeton Conference on Information Sciences and Systems*. Princeton, 1971.

[124] Ninghui Li and Ziqing Mao. Administration in role-based access control. In Feng Bao and Steven Miller, editors, *Proceedings of the 2007 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2007, Singapore, March 20-22, 2007*, pages 127–138. ACM, 2007.

[125] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust-management framework. In *2002 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 12-15, 2002*, pages 114–130. IEEE Computer Society, 2002.

[126] Catherine J. McCollum, J. R. Messing, and LouAnna Notargiacomo. Beyond the pale of MAC and DAC - Defining new forms of access control. In *Proceedings of the 1990 IEEE Symposium on Security and Privacy, Oakland, California, USA, May 7-9, 1990*, pages 190–200. IEEE Computer Society, 1990.

[127] Robert McNaughton and Hisao Yamada. Regular expressions and state graphs for automata. *IRE Transactions on Electronic Computers*, 9(1):39–47, 1960.

[128] Pooya Mehregan and Philip W. L. Fong. Policy negotiation for co-owned resources in relationship-based access control. In Wang et al. [185], pages 125–136.

[129] Elena Meshkova, Janne Riihijärvi, Marina Petrova, and Petri Mähönen. A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks. *Computer Networks*, 52(11):2097–2128, 2008.

[130] Microsoft Corporation. Managing authorization and access control. Online, November 2005. `https://technet.microsoft.com/en-gb/library/bb457115.aspx` Accessed: February 28, 2017.

[131] Microsoft Corporation. Hate to see you go, but it's time to move on to greener pastures. A farewell to Authorization Manager aka AzMan. Online, August 2014. `https://blogs.technet.microsoft.com/askds/2014/08/21/hate-to-see-you-go-but-its-time-to-move-on-to-greener-pastures-a-farewell-to-authorization-manger-aka-azman/` Accessed: February 26, 2017.

[132] Microsoft Corporation. Overview of Authorization Manager. Online, July 2014. https://technet.microsoft.com/en-us/library/cc732203.aspx Accessed: February 26, 2017.

[133] Microsoft Corporation. Access control overview. Online, June 2016. https://technet.microsoft.com/en-us/itpro/windows/keep-secure/access-control Accessed: February 26, 2017.

[134] Microsoft Corporation. Dynamic access control overview. Online, September 2016. https://technet.microsoft.com/en-us/itpro/windows/keep-secure/dynamic-access-control Accessed: February 26, 2017.

[135] Microsoft Corporation. Security principals. Online, January 2016. https://technet.microsoft.com/en-us/itpro/windows/keep-secure/security-principals Accessed: February 26, 2017.

[136] Julien Mineraud, Oleksiy Mazhelis, Xiang Su, and Sasu Tarkoma. A gap analysis of Internet-of-Things platforms. *Computer Communications*, 89-90:5–16, 2016.

[137] Roberto Minerva, Abyi Biru, and Domenico Rotondi. Towards a definition of the Internet of Things (IoT). Technical report, IEEE Internet of Things Technical Community, May 2015.

[138] Andrei Mituca, Amir H. Moin, and Christian Prehofer. Access control for apps running on constrained devices in the Internet of Things. In Gabriel Ghinita, Razvan Rughinis, and Ahmad-Reza Sadeghi, editors, *2014 International Workshop on Secure Internet of Things, SIoT 2014, Wroclaw, Poland, September 10, 2014*, pages 1–9. IEEE Computer Society, 2014.

[139] National Cyber Security Centre (NCSC). Security for industrial control systems. Online. https://www.ncsc.gov.uk/guidance/security-industrial-control-systems Accessed: April 20, 2017.

[140] OASIS XACML Technical Committee. *XACML v3.0 Administration and Delegation Profile Version 1.0*. OASIS, August 2010. OASIS Committee Specification 01.

[141] OASIS XACML Technical Committee. *eXtensible Access Control Markup Language (XACML) Version 3.0*. OASIS, January 2013.

[142] OASIS XACML Technical Committee. *XACML v3.0 Administration and Delegation Profile Version 1.0*. OASIS, November 2014. Edited by Erik Rissanen

and Hal Lockhart. OASIS Committee Specification Draft 04 / Public Review Draft 02.

[143] Jaehong Park and Ravi S. Sandhu. Towards usage control models: Beyond traditional access control. In Ravi Sandhu and Elisa Bertino, editors, *7th ACM Symposium on Access Control Models and Technologies, SACMAT 2002, Naval Postgraduate School, Monterey, California, USA, June 3-4, 2002*, pages 57–64. ACM, 2002.

[144] Jaehong Park and Ravi S. Sandhu. The UCON_{ABC} usage control model. *ACM Transactions on Information and System Security (TISSEC)*, 7(1):128–174, 2004.

[145] Joon S. Park and Ravi S. Sandhu. Secure cookies on the web. *IEEE Internet Computing*, 4(4):36–44, 2000.

[146] PenTestPartners. Blog - Internet of Things. Online. `https://www.pentestpartners.com/internet-of-things/` Accessed: March 31, 2017.

[147] Pablo Punal Pereira, Jens Eliasson, and Jerker Delsing. An authentication and access control framework for CoAP-based Internet of Things. In *Proceedings of the 40th Annual Confernce of the IEEE Industrial Electronics Society, IECON 2014, Pittsburgh, Pennsylvania, Oct 29 - Nov 1 2014*, pages 5293–5299. IEEE Industrial Electronics Society, 2014.

[148] Charith Perera, Arkady B. Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. Context aware computing for the Internet of Things: A survey. *IEEE Communications Surveys and Tutorials*, 16(1):414–454, 2014.

[149] Manuel Gil Perez, Gabriel López, Antonio F Gómez Skarmeta, and Aljosa Pasic. Advanced policies for the administrative delegation in federated environments. In *Dependability (DEPEND), 2010 Third International Conference on*, pages 76–82. IEEE, 2010.

[150] Lauri I. W. Pesonen, David M. Eyers, and Jean Bacon. A capability-based access control architecture for multi-domain publish/subscribe systems. In *2006 International Symposium on Applications and the Internet (SAINT 2006), 23-27 January 2006, Phoenix, Arizona, USA*, pages 222–228. IEEE Computer Society, 2006.

[151] Lauri I. W. Pesonen, David M. Eyers, and Jean Bacon. Encryption-enforced access control in dynamic multi-domain publish/subscribe networks. In Hans-Arno Jacobsen, Gero Mühl, and Michael A. Jaeger, editors, *Proceedings of the*

2007 Inaugural International Conference on Distributed Event-Based Systems, DEBS 2007, Toronto, Ontario, Canada, June 20-22, 2007, volume 233 of *ACM International Conference Proceeding Series*, pages 104–115. ACM, 2007.

[152] Quest. Active roles. Online. `https://www.quest.com/products/active-roles/` Accessed: February 28, 2017.

[153] Intuit QuickBooks. User management - Add, delete, or change a user's access. Online. `https://community.intuit.com/articles/1145546-user-management-add-delete-or-change-a-user-s-access` Accessed: February 27, 2017.

[154] Michael O. Rabin and Dana S. Scott. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3(2):114–125, 1959.

[155] Dennis M. Ritchie and Ken Thompson. The UNIX time-sharing system. *The Bell System Technical Journal*, 57(6):1905–1929, 1978. Revised version of an article that appeared in Communications of the ACM, 17, No. 7 (July 1974), pp. 365-375.

[156] Syed Zain R. Rizvi and Philip W. L. Fong. Interoperability of relationship- and role-based access control. In Bertino et al. [32], pages 231–242.

[157] Syed Zain R. Rizvi, Philip W. L. Fong, Jason Crampton, and James Sellwood. Relationship-based access control for an open-source medical records system. In Edgar R. Weippl, Florian Kerschbaum, and Adam J. Lee, editors, *Proceedings of the 20th ACM Symposium on Access Control Models and Technologies, Vienna, Austria, June 1-3, 2015*, pages 113–124. ACM, 2015.

[158] Rodrigo Roman, Pablo Najera, and Javier Lopez. Securing the Internet of Things. *IEEE Computer*, 44(9):51–58, 2011.

[159] Salesforce. Comparing security models. Online. `https://help.salesforce.com/articleView?id=users_sharing_vs_obj_level_perms.htm&type=0&language=en_US&release=206.9` Accessed: February 28, 2017.

[160] Farzad Salim, Jason Reid, and Ed Dawson. An administrative model for UCON_{ABC}. In Colin Boyd and Willy Susilo, editors, *8th Australasian Information Security Conference 2010, AISC 2010, Brisbane, Australia, January 2010*, volume 105 of *CRPIT*, pages 32–38. Australian Computer Society, 2010.

[161] Jerome H. Saltzer. Protection and the control of information sharing in Multics. *Communications of the ACM*, 17(7):388–402, 1974.

[162] Jerome H. Saltzer and Michael D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63(9):1278–1308, 1975.

[163] Ravi S. Sandhu, Venkata Bhamidipati, and Qamar Munawer. The ARBAC97 model for role-based administration of roles. *ACM Transactions on Information and System Security (TISSEC)*, 2(1):105–135, 1999.

[164] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996.

[165] Ravi S. Sandhu and Jaehong Park. Usage control: A vision for next generation access control. In Vladimir Gorodetsky, Leonard J. Popyack, and Victor A. Skormin, editors, *MMM-ACNS*, volume 2776 of *Lecture Notes in Computer Science*, pages 17–31. Springer, 2003.

[166] Ravi S Sandhu and Pierangela Samarati. Access control: Principles and practice. *IEEE Communications Magazine*, 32(9):40–48, 1994.

[167] Kent E. Seamons, Marianne Winslett, and Ting Yu. Limiting the disclosure of access control policies during automated trust negotiation. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2001, San Diego, California, USA*. The Internet Society, 2001.

[168] James Sellwood and Jason Crampton. Sleeping Android: The danger of dormant permissions. In William Enck, Adrienne Porter Felt, and N. Asokan, editors, *SPSM'13, Proceedings of the 2013 ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, Co-located with CCS 2013, November 8, 2013, Berlin, Germany*, pages 55–66. ACM, 2013.

[169] Basit Shafiq, James Joshi, Elisa Bertino, and Arif Ghafoor. Secure interoperation in a multidomain environment employing RBAC policies. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17(11):1557–1577, 2005.

[170] Mohamed Shehab, Elisa Bertino, and Arif Ghafoor. SERAT: Secure role mapping technique for decentralized secure interoperability. In Ferrari and Ahn [81], pages 159–167.

[171] Richard T. Simon and Mary Ellen Zurko. Separation of duty in role-based environments. In *CSFW*, pages 183–194. IEEE Computer Society, 1997.

[172] Aaron Sorkin. The West Wing. Warner Bros. Television, April 2000.

[173] Mudhakar Srivatsa, Ling Liu, and Arun Iyengar. EventGuard: A system architecture for securing publish-subscribe networks. *ACM Transactions on Computer Systems (TOCS)*, 29(4):10:1–10:40, 2011.

[174] Scott D. Stoller. An administrative model for relationship-based access control. In Pierangela Samarati, editor, *Data and Applications Security and Privacy XXIX - 29th Annual IFIP WG 11.3 Working Conference, DBSec 2015, Fairfax, VA, USA, July 13-15, 2015, Proceedings*, volume 9149 of *Lecture Notes in Computer Science*, pages 53–68. Springer, 2015.

[175] Roberto Tamassia, Bernardo Palazzi, and Charalampos Papamanthou. Graph drawing for security visualization. In Ioannis G. Tollis and Maurizio Patrignani, editors, *Graph Drawing, 16th International Symposium, GD 2008, Heraklion, Crete, Greece, September 21-24, 2008. Revised Papers*, volume 5417 of *Lecture Notes in Computer Science*, pages 2–13. Springer, 2008.

[176] Kaijun Tan, Jason Crampton, and Carl A. Gunter. The consistency of task-based authorization constraints in workflow systems. In *17th IEEE Computer Security Foundations Workshop, (CSFW-17 2004), 28-30 June 2004, Pacific Grove, CA, USA*, page 155. IEEE Computer Society, 2004.

[177] The Guardian. Internet of things. Online. `https://www.theguardian.com/technology/internet-of-things` Accessed: April 6, 2017.

[178] The TrustedBSD Project. TrustedBSD mandatory access control (MAC) framework. Online. `http://www.trustedbsd.org/mac.html` Accessed: February 26, 2017.

[179] Tizen. Tizen IVI 3.0 M3 June 2014 has been released. Online, July 2014. `https://www.tizen.org/blogs/tsg/2014/tizen-ivi-3.0-m3-june-2014-has-been-released` Accessed: February 26, 2017.

[180] Michael Carl Tschantz and Shriram Krishnamurthi. Towards reasonability properties for access-control policy languages. In David F. Ferraiolo and Indrakshi Ray, editors, *11th ACM Symposium on Access Control Models and Technologies, SACMAT 2006, Lake Tahoe, California, USA, June 7-9, 2006, Proceedings*, pages 160–169. ACM, 2006.

[181] Ubuntu. AppArmor. Online, June 2012. `https://help.ubuntu.com/community/AppArmor` Accessed: February 26, 2017.

[182] International Telecommunication Union. The Internet of Things. Technical report, November 2005.

[183] J. Vollbrecht, P. Calhoun, S. Farrell, L. Gommans, G. Gross, B. de Bruijn, C. de Laat, M. Holdrege, and D. Spence. *AAA Architecture Framework*. Internet Engineering Task Force (IETF), January 2000. Draft Version 00.

[184] Jacques Wainer, Paulo Barthelmess, and Akhil Kumar. W-RBAC - A workflow security model incorporating controlled overriding of constraints. *International Journal of Cooperative Information Systems*, 12(4):455–485, 2003.

[185] X. Sean Wang, Lujo Bauer, and Florian Kerschbaum, editors. *Proceedings of the 21st ACM Symposium on Access Control Models and Technologies, SACMAT 2016, Shanghai, China, June 5-8, 2016*. ACM, 2016.

[186] Rolf H Weber. Internet of Things - New security and privacy challenges. *Computer Law & Security Review*, 26(1):23–30, 2010.

[187] Qiang Wei, Jason Crampton, Konstantin Beznosov, and Matei Ripeanu. Authorization recycling in hierarchical RBAC systems. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):3, 2011.

[188] WIRED. IOT. Online. `https://www.wired.com/tag/iot/` Accessed: April 6, 2017.

[189] Derick Wood. *Theory of Computation*. Harper & Row, Publishers, Inc., New York, NY, USA, 1987.

[190] Jun Wu, Mianxiong Dong, Kaoru Ota, Jianhua Li, and Bei Pei. A fine-grained cross-domain access control mechanism for social Internet of Things. In *2014 IEEE 11th Intl Conf on Ubiquitous Intelligence and Computing and 2014 IEEE 11th Intl Conf on Autonomic and Trusted Computing and 2014 IEEE 14th Intl Conf on Scalable Computing and Communications and Its Associated Workshops, Bali, Indonesia, December 9-12, 2014*, pages 666–671. IEEE Computer Society, 2014.

[191] Raj Yavatkar, Dimitrios Pendarakis, and Roch Guerin. *A Framework for Policy-based Admission Control*. Internet Engineering Task Force (IETF), November 1997. Draft Version 00.

[192] Guoping Zhang and Wentao Gong. The research of access control based on UCON in the Internet of Things. *Journal of Software (JSW)*, 6(4):724–731, 2011.

[193] Guoping Zhang and Jiazheng Tian. An extended role based access control model for the Internet of Things. In *2010 International Conference on Information, Networking and Automation, ICINA 2010, Kunming, China, October 17-19, 2010, Proceedings*, volume 1, pages 319–323. IEEE, 2010.

[194] Rui Zhang, Alessandro Artale, Fausto Giunchiglia, and Bruno Crispo. Using description logics in relation based access control. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Description Logics*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.