Business Process Access Control (BPAC): workflow-based authorisation for complex systems

by

Derrick Newton

A thesis submitted to
The University of Birmingham
for the degree of
DOCTOR OF PHILOSOPHY

School of Computer Science
The University of Birmingham
April 2011

Abstract

Segregation of duties and least privilege are two business principles that protect an organisation's valuable data from information leak. In this thesis we demonstrate how these business principles can be addressed through workflow-based access control. We present Business Process Access Control (BPAC), a workflow-based access control modelling environment that properly enacts the key business principles through constraints and we implement BPAC in the applied pi calculus. We ensure that constraints are correctly applied within our BPAC implementation by introducing the concept of stores. We propose a selection of security properties in respect of the business principles and we develop tests for these properties. The collusion metric is introduced as a simple indicator as to the resistance of a workflow-based access control policy to fraudulent collusion. We identify an anonymity property for workflows as the inability of an outside observer to correctly match agents to workflow tasks and we propose that anonymity provides protection against collusion. We introduce a lightweight version of labelled bisimilarity: the abstraction test and we apply this test to workflow security properties. We develop a test for anonymity using labelled bisimilarity and we demonstrate its application through simple examples.

To my darling wife Sarah

and in memory of

Graham Newton 1942–2006

Baz 2010

Acknowledgements

Firstly, I would like to extend my deepest gratitude to my supervisor, Eike Ritter. Eike has guided, tested and critically appraised the project throughout its evolution and I am certain that it would never have reached fruition if it was not for his hard work and constant support. I have learnt a huge amount from Eike throughout the course of this project and I will miss our meetings now that it is over.

Next, I would like to thank my thesis group members: Mark Ryan and Martín Escardó. They have made a real and significant contribution to this project and their comments and analysis are gratefully appreciated.

Many thanks are extended to Steve Vickers, particularly in respect of his support and assistance when external events overshadowed the project, Manfred Kerber for our delightful conversations and Russell Beale for the pep talks and encouragement.

I acknowledge the work performed by my external examiner Eerke Boiten in identifying the various flaws in this thesis and providing suggestions as to their correction.

I thank my colleagues in the School of Computer Science, particularly the members of the Computer Security Reading Group. A PhD can a be pretty lonely and depressing pursuit at times and the friendship of the School has been a considerable asset.

I thank my wife, Sarah, for her support, encouragement and patience. I will clear up all of my papers so that you can have the house back now.

Finally, I thank my mother, Paula, my mother-in-law, Joan, my sister, Tina, Min and all of my friends, particularly Mark, Richard and Pam, for keeping me sane throughout the duration of this project.

Contents

List of Figures

List of Tables

Nomenclature

ACM Access Control Matrix

ANSI American National Standards Institute

BPAC Business Process Access Control

BPEL4WS Business Process Execution Language for Web Services

BPM Business Process Management

BPML Business Process Modeling Language

BPMN Business Process Modeling Notation

CCS Calculus of Communicating Systems

CSP Communicating Serial Processes

DAC Discretionary Access Control

DMER Dynamic Mutually Exclusive Roles

ISO International Organization for Standardization

MAC Mandatory Access Control

NIST National Institute of Standards and Technology

OASIS Organization for the Advancement of Structured Information Standards

PC Programme Committee

RBAC Role-Based Access Control

SMER Statically Mutually Exclusive Roles

SoD Segregation/Separation of Duties

SPKI Simple Public Key Infrastructure

UML Unified Modeling Language

W-RBAC Workflow-based Role-Based Access Control

WSBPEL Web Services Business Process Execution Language

XML eXstensible Markup Language

XPDL XML Process Definition Language

Chapter 1

Introduction

1.1 Thesis problem

Access control in computer systems of organisations represents a key implementation of regulatory constraints. Access control is required to replicate complex regulatory requirements within a heterogeneous mix of hardware and software. This is achieved by ensuring that users are properly assigned the resources they require. Additionally, access to resources is prevented for those agents who lack the required levels of authorisation.

Access control is the mapping of users or agents to resources. Role-Based Access Control systems (RBAC) have been researched and developed extensively and a formal standard has been proposed [82] and adopted by the US National Institute of Standards and Technology (NIST) [69]. Often, RBACs incorporate a policy language that is based upon symbolic logic, such as the Datalog-based RT family of languages [64, 63, 60] or Cassandra [13]. Languages have been extended to incorporate such organisational constructs as hierarchies and delegation [63, 70]. Other implementations concentrate on the verification of policies [12, 104]. RBACs have also been applied to workflows as a more appropriate reflection of real-world organisational structures [10, 18, 26].

No single user should be able to exercise control over an organisation's data and resources to such an extent that it could cause a leak of information to a third party which would be materially detrimental to the organisation. This concept can be extended to consideration of the collective leak of information to third parties either through fraudulent collusion or accidental distribution among users. Additionally, users should only be able to access sufficient data and resources to properly enable them to fulfil their duties and responsibilities within an organisation. To facilitate this protection, certain business principles

are applied as structurally fundamental constraints to the access control paradigm: specifically, the requirements of static and dynamic segregation of duties (SoD) and the least privilege or need-to-know principle. Static SoD involves the generation of a fixed set of user/role/task constraints for an access control policy ruleset whereas dynamic SoD represents a runtime user/role/task constraint base that is dependent upon historical access mappings.

With particular reference to RBACs, we observe in the review of literature in section 2.4.4 below that static SoD is sometimes misinterpreted as a least privilege problem [13], or it is enacted using statically mutually exclusive roles (SMER) within access control systems. Whilst SMERs can yield static SoD as specified in the RBAC standard [69, 83], constructing SMERs is a complex problem and verification of SMERs is intractable [61]. Conceptually, static SoD is a comparatively straightforward implementation of the business rule but it is also inflexible, particularly within multi-role environments and it is not well suited to the management of human interaction with the system.

Dynamic SoD is much better suited to the complexities of real-world access control within organisations but implementation in the literature is confused. In some papers [13, 69, 82, 104], dynamic mutually exclusive roles (DMER) are used as a means to enacting dynamic SoD but we point out in section 2.4.4 that DMERs do not of themselves yield dynamic SoD [61]. Dynamic SoD is properly implemented in systems of access control designed around tasks and workflows [19, 26] but these systems do not incorporate a formal modelling structure for the analysis of security properties for the access control system as a whole.

Our thesis presents an access control modelling environment called Business Process Access Control (BPAC), a workflow-based paradigm that properly and fully implements static and dynamic segregation of duties, coupled with a strict interpretation of the least privilege rule. Unlike the existing workflow-based solutions that enable SoD constraints, our solution incorporates a formal modelling structure based upon process calculus. This modelling structure facilitates testing of access control policies for reachability properties, such as satisfiability and testing of the policy dynamics for information leak through agent control of a process chain. Additionally, the modelling structure extends the testing to incorporate an analysis and measure for n-party collusion, an area that has received considerably less attention within the literature on RBACs and their derivatives. Furthermore, we utilise the considerable power of process algebra, specifically observational equivalence, to produce a test for anonymity of agents interacting with workflows.

1.2 Our solution

We tackle the problem of access control modelling, in which business principles can be properly implemented and analysed, by developing a modelling environment for access control that fully implements dynamic SoD through workflow-based access control and state histories. The proposed system provides an environment for developing access control policies and testing that the resulting security level is sufficient to satisfy the particular requirements of an organisation. Furthermore, we extend the security model to consider thresholds of n-party collusion as a measure of the security provided by the access control system: the collusion metric. Additionally, we investigate the issue of agent anonymity within workflow-based access control and we outline a test for agent anonymity based upon observational equivalence.

A workflow-based system of access control represents an evolution of role-based access control that:

- enables the application of static and dynamic segregation of duties through the combination of task dependencies and state histories,
- provides an abstraction layer between agents and agent resources through the use of tasks. Consequently, the precise details of resource allocation are irrelevant from the perspective of creating and analysing access control policies,
- provides a means whereby access control policy creation can be linked directly to process design within business process modelling and development,
- can be used to model traditional RBAC.

1.3 Summary of the key thesis contributions

1.3.1 Workflow-based access control systems

Whilst workflow-based access control systems have been proposed by others, e.g. Botha et al [25, 26, 27], we concentrate on the implementation of business rules within systems of access control. Also, we propose a selection of security properties relating to the correct or otherwise implementation of business rules. This provides motivation for the development of security tests within our modelling environment.

1.3.2 A modelling environment for systems of workflow-based access control: BPAC

We present a modelling environment for workflow-based systems of access control called Business Process Access Control (BPAC), which we implement using the applied pi calculus. The applied pi calculus can be considered to be somewhat akin to a primitive, low-level programming language with a well established formal theoretical structure. In particular, the applied pi calculus supports powerful analytical tools, such as observational equivalence, that facilitate the design of tools for testing and analysis of workflow-based access control models. This is an essential component of our modelling concept.

As part of our development of the BPAC implementation we design processes in the applied pi calculus that enable us to handle stored data. The resulting store processes can be written to, have items deleted from them and can be tested against via a simple pattern matching process. The stores are an essential part of our BPAC modelling environment that enable SoD to be implemented fully. In effect, the stores provide a means for recording access control state within the modelling environment. This is essential to ensuring that conditional task access policies can be modelled properly without recourse to complex task access permission matrices and the like.

There are tools, ProVerif [1, 21, 22] for example, that enable modelling and automated testing of models prepared in the applied pi calculus and it is envisaged that, ultimately, an automated tool for the analysis of BPAC models will be developed. However, these tools are currently somewhat limited in their scope: observational equivalence proof techniques are not yet complete within ProVerif, there is an inability to handle state histories and there is the problem of computational explosion. Consequently, we concentrate upon manual coding and testing of models within this thesis so that the basic modelling environment and analysis tools are established.

1.3.3 Security analysis of workflow-based access control models

As part of our modelling environment we propose and develop various security tests that can be applied to models of workflow-based access control within the BPAC environment. These include tests of reachability, satisfiability, collusion analysis and anonymity.

1.3.3.1 Tests of satisfiability

We use satisfiability tests on workflow-based access control models to establish whether or not a workflow, together with a set of agents, role assignments and an access control policy, can be completed. An access control policy is poorly defined in respect of a workflow if that workflow cannot be completed.

1.3.3.2 The collusion metric

We introduce a measure, which we use to analyse the consequences of the application of segregation of duties to access control, called the collusion metric. The collusion metric represents the minimum number of agents that are required to establish complete control over workflow resources, given a set of agents, role assignments and an access control policy. It is a basic security measure for the workflow. The smaller the collusion metric then the more exposed is a workflow to agent control and the fewer the number of agents needed to release complete information and control to an outside third party.

1.3.3.3 Anonymity testing via observational equivalence

We propose an anonymity test for workflow-based access control based upon observational equivalence. We consider anonymity for a particular agent to be preserved if it can be demonstrated that no information about the agent and its association with a task or tasks within a workflow can be observed by an outside third party. Anonymity is not preserved for a particular agent if it can be demonstrated that the agent is critical to the completion of a workflow or the agent can be identified with a particular task. Failure can occur either because of the specific mechanics of agent/task interaction or via information leak to an outside third party.

1.4 Outline of thesis

Chapter 2 provides a background summary and related work. We discuss the modelling of workflow-based access control, which we call Business Process Access Control (BPAC), together with examples in chapter 3. We present a simple workflow example in chapter 4, which we revisit throughout the remainder of the thesis. In chapter 5 we introduce the applied pi calculus, adapted for use as our access control modelling environment. In chapter 6 we define the building blocks of our BPAC implementation using the applied pi calculus. We return to the simple workflow example of chapter 4 in chapter 7 and we demonstrate the encoding of the simple model in the applied pi calculus implementation of BPAC. We discuss security analysis of models within BPAC and its application in the applied pi calculus in chapter 8 and we return once again to our simple workflow example in chapter 9 to discuss examples of security analysis in the context of the workflow example. Finally, we present our conclusions and possible future work in chapter 10.

Chapter 2

Background and related work

2.1 Introduction

In this chapter, we present a discussion of literature regarding systems of access control and business process management (BPM). In the section assigned to BPM we concentrate on a review of suitable modelling environments for BPM with particular emphasis on Petri nets and process calculi. We draw together our arguments from both fields to motivate our thesis concept of Business Process Access Control (BPAC) and we present arguments to justify our selection of the applied pi calculus as the basis for our access control modelling environment.

2.2 The security requirement

Companies, charities, professional bodies and public organisations throughout the world are subjected to increased regulatory controls. In the European Union protection for the storage of personal data by organisations is enacted through various national interpretations of the European Data Protection Directive [45], such as the Data Protection Act 1998 [51] in the United Kingdom. Financial constraints are imposed by tax laws and accounting regulations or by the regulatory frameworks of professional bodies where applicable. These financial constraints are becoming increasingly prescriptive in respect of the day-to-day recording and maintenance of business transactions, e.g. Sarbanes-Oxley section 404 [87] in the USA. Additionally, organisations are keen to display their adherence to quality assurance standards, such as ISO 9000, which require detailed documentation and application of systems and organisation structures.

The world press frequently highlights and dramatises breaches of IT security, especially when the breach represents a threat to individual privacy through the failure of large organisations. Examples of security failure include the UK Department of Health publishing applicant doctors' personal information [40, 71] and compromise of personnel records of over half a million US military employees by a Pentagon contractor [73, 81]. Insiders of organisations cause many security breaches, either deliberately or through negligence or accident. An audit review in 2007 by the US Treasury of 102 employees of the Internal Revenue Service (IRS) discovered that 60% of the employee sample released login information and changed their passwords for auditors posing as computer support helpdesk representatives [75]. When this exercise was performed some three years earlier the failure was 35% and the failure rate was 71% some three years prior to that. The recommendations of the report focused, not unreasonably, upon employee training and discipline. However, given the enduring nature of employee security failure, an additional contribution could be made by reducing the availability of information and resources to employees to that which is strictly necessary for the performance of their duties, i.e. the proper implementation of access control.

Reducing employee access to strictly necessary IT resources implies the system-wide enforcement of two key organisational principles, full segregation (alternatively separation) of duties and least privilege or the need-to-know basis. In the real world, such a draconian approach to access control may be impractical because of staff numbers, counter-productive because human beings are inclined to take shortcuts when faced with too many obstacles or contrary to productivity or creativity within an organisation. However, the application of these principles to real access control problems must surely be considered if the cost to the organisation of loss of information outweighs the cost of implementation. Modelling of access control policies provides organisations with a cost-effective means of policy appraisal that can provide valuable information for decision making in respect of their subsequent implementation.

2.3 Access control

Access control is the assignment of resources to users or agents within organisations. Various mechanisms for access control have been proposed and applied to real-world situations and these are discussed in greater detail in section 2.5 below. Most current implementations of access control are based upon role-based access control (RBAC). Within RBAC the assignment of resources to agents is mediated by the mapping of roles to agents and access to resources is constrained to specific roles. Formally, a simple RBAC can be defined as follows if we assume that units of resources are each assigned a single role for access control purposes:

Definition 1

Let S be a set of resources, A be a set of agents and R be a set of roles. We represent agent/role assignment by the finite binary relation $AR \subseteq A \times R$ and $SR : S \longrightarrow R$ is a resource/role assignment function. If for some $s \in S$ there exists an a such that $a \in A$ and $a\ AR\ SR(s)$ then agent a can access resource s.

2.3.1 Implementation of access control

Part of the implementation of access control is the application of business rules that specify how agents interact with resources and the relationship between agent/resource interactions for a given set of agents and resources. We discuss some of these business rules below.

2.4 Business rules

2.4.1 Segregation of duties

The Auditing Practices Board International Standard of Auditing (UK and Ireland) 315 defines segregation of duties (SoD) as follows:

> *"Assigning different people the responsibilities of authorizing transactions, recording transactions, and maintaining custody of assets is intended to reduce the opportunities to allow any person to be in a position to both penetrate and conceal errors or fraud in the normal course of the person's duties."* [8]

This well-established principle has long been promoted by the major accounting and auditing institutes worldwide as a means to counter the possibility of an individual employee of an organisation gaining control over an entire process chain. For example, a purchase clerk controlling the ordering, authorisation and payment for a new computer for herself would clearly be depriving the organisation. A simple segregation of duties can be achieved in this example by assigning a second employee to the authorisation task. Now it is observed prima facie that the purchase clerk (or the newly installed employee) cannot control the entire process chain and a level of protection against fraud is attained. However, it is still possible for the two employees to collude to control the process chain so although protection is improved there is still a real risk of fraudulent activity. Increasing the number of employees to three further improves protection against fraud, as three-party collusion is now required to gain control over the process chain.

Whilst it is not unreasonable to assume that the greater the level of segregation of duties the greater the protection against fraudulent activity there are practical limitations to the implementation of the policy. These limitations include the number of available employees, the number of tasks per process chain and the organisation's attitude to risk.

SoD can be invoked in either a static or a dynamic context. Static SoD refers to the predefinition of access policies for users through a matrix, say, of users mapped to permissions. Dynamic SoD, on the other hand, assigns permissions to users based upon both the static permission mappings and the history of previous user/permission mappings for a particular process chain. An alternative definition by Nash and Holland is that a user can execute a transaction if they are authorised to perform that transaction on a data item and they have not previously executed a transaction on the data item [68]. Within a real world scenario, a purchase ledger transaction, say, it is likely that precise a priori permission mappings as required by static SoD are not practical as users might be assigned multiple roles: a supervisor might also be a purchases clerk, for example and a user can potentially access a transaction based upon any of her role assignments. Consequently, the user might be assigned a transaction based upon her purchases clerk role and then assigned a subsequent transaction within the process chain based upon her supervisor role in breach of an SoD requirement. Dynamic SoD uses its user/permission history to enforce SoD in this case by ensuring that if the user has accessed the first transaction as a purchases clerk then she cannot access the subsequent transaction as a supervisor. This is an important requirement in practical systems.

2.4.2 Least privilege

The US Department of Defense defines least privilege in its publication "Trusted Computer System Evaluation Criteria", commonly referred to as the Orange Book, as follows:

> *"Least Privilege — This principle requires that each subject in a system be granted the most restrictive set of privileges (or lowest clearance) needed for the performance of authorized tasks. The application of this principle limits the damage that can result from accident, error, or unauthorized use."* [38]

This principle can manifest itself in various ways. An example of least privilege occurs when roles are assigned to employees within an organisation and an employee is able to activate more than one role. For example, a purchases manager might be able to activate the purchases clerk role. If the manager is accessing some manager-specific resources in pursuit of their managerial duties, that employee should not be able to invoke their clerk role as well. Implementation of least privilege in this context can be seen

to contribute to segregation of duties as discussed above. After all, a role-based segregation of duties could be compromised by the presence of an employee activating multiple rules so strict least privilege is a requirement of the segregation of duties policy.

Resources should be the minimum required for the performance of a given task and the resources should only be made available for the duration of the task. Application of this principle reduces the opportunity for employees to leak or damage sensitive information.

2.4.3 Delegation

Delegation is the business process whereby an agent with a resource access permission temporarily transfers permission to another agent. For example, within an RBAC a delegated permission can be achieved by temporarily assigning a role r to another agent such that r satisfies the resource/role assignment $(s, r) \in SR$ for resource s. Delegation provides a mechanism to circumvent practical problems, such as the absence of given agents, that might otherwise prevent the operation of some particular task.

2.4.4 The application of business rules within systems of access control

Given the business context for access control set out in section 2.2 above then a requirement for a system for access control is that it should properly enact business rules and policies. These include role hierarchies and inheritance, static and dynamic segregation of duties, delegation, role appointment, role enforcement and strict need-to-know. It is apparent from the literature that the various proposed systems tackle some of these rules with varying degrees of success. For example, the extensions to RT, RT^{T} and RT^{D} provide language components for segregation of duties and role delegation respectively [60] and Sandhu et al's extensions to the RBAC96 proposed standard, RBAC_1 and RBAC_2 add role hierarchies and constraints respectively [83].

Segregation of duties proves to be particularly problematic in the literature. Simon and Zurko [84] provide a detailed review on segregation of duties and its implementation in the literature to date. They stress that the simplicity of static SoD is coupled with inflexibility and a lack of applicability to real-world human situations. Simon and Zurko's discussion on dynamic segregation identifies the most flexible variant of SoD as that which is based upon the use of individual histories assigned to users. These SoD ideas are incorporated in the team's authorisation tool called Adage, a self-contained user-centred RBAC.

The definition in section 2.4.1 highlights the fact that segregation of duties requires a number of different agents to perform a connected set of tasks to ensure that no one person has control over an entire process chain. However, in the literature SoD is sometimes dealt with in access control models, both

within the static and the dynamic contexts, as a single party problem implemented via the activation and deactivation of roles by agents. For instance, in Cassandra a policy can be defined so that an agent cannot activate more than one role at the same time [13]. The policy treatment recalls a similar approach by Ferraiolo et al some years earlier [46]. This is not strictly segregation of duties but an interpretation of the principle of least privilege through mutual exclusivity.

In other papers, such as those by Sandhu, segregation of duties is presented as a set of constraints upon the agent/role mapping [83] and this is the basis for static SoD as implemented in the NIST standard [69]. Ninghui Li et al refer to this implementation as Statically Mutually Exclusive Roles (SMER) [61]. Whilst SMERs can properly be used to enforce static segregation of duties, setting up an SMER matrix for a large organisation represents a complex task. Indeed, Li et al demonstrate that verifying that an SMER enforces SoD is intractable (coNP complete) [61]. Operationally, the problem with SMERs is their separation from the complexities of business processes within organisations and the lack of granularity and flexibility of a user/role based constraint system. If a simple SMER constraint is defined over three roles as follows per Li [61]:

$$c_1 \quad = \quad smer\left\langle \left\{ Warehouse, Marketing, Finance \right\}, 2 \right\rangle$$

i.e. no user can be a member of any two roles in the constraint, then although this constraint properly enforces static SoD the rule is absolute over all business processes to which the roles are assigned. Granularity can be improved by defining roles more specific to business processes but this approach increases the complexity of the SMER and tends toward negating the benefits of RBACs via, for example, an increased ratio of role numbers to users/agents.

If the literature is somewhat confused over static SoD, it is completely muddled over dynamic SoD. Dynamic Mutually Exclusive Roles (DMER) are sometimes implemented in the literature. As defined by Li et al DMER constrains a user such that she cannot simultaneously activate mutually exclusive roles [61]. Various publications [13, 46, 69, 82, 104] interpret the application of DMER as an enactment of dynamic SoD. DMER can be seen to be an interpretation of least privilege but as Li et al state it is not SoD and cannot of itself enable static or dynamic SoD [61]. If there are multiple sessions within a model of access control then an agent can activate a role and deactivate a role in one session then activate another role in a second session so as to circumvent any dynamic SoD constraint enacted by DMER. For example, we might require for SoD that a payroll preparation task is performed by a different person to a payroll payment approval task. Using DMER we can ensure that a specified agent cannot simultaneously

access both tasks but DMER does not prevent the agent from relinquishing her payroll preparation role and subsequently enacting her payment role, circumventing the SoD constraint. Dynamic SoD is properly handled by Botha [26] and Bertino et al [19] with their workflow and task-based systems but there are limitations to their approaches with respect to the formal modelling and analysis of the access control system as reported in section 2.5.6.

A key requirement of a business system of access control is to prevent single agent fraud through ensuring that agents cannot perform a number of critical tasks within a process chain. This necessitates different role assignments to tasks within the process chain together with the possibility of task access dependencies on the completion of other tasks within the chain i.e. dynamic SoD. A further requirement is that the access control system has a memory of access for the duration of the process chain or workflow. In addition, it is desirable that the possibilities of multiple agent fraudulent collusion are reduced by ensuring that certain communicating groups of agents cannot perform a number of critical tasks within the process chain and by enforcing the strict need-to-know basis for agent/role/task assignments. In summary, the access control system should incorporate an implementation of global constraints across business processes. An important consequence of the application of these requirements is the reduced potential for information leak from the system to unauthorised parties by ensuring that agents are unable to access information outside of their role/task responsibilities.

We have identified above that a number of access control systems either do not attempt to implement DDoS or that they use DMER to enact DDoS. These access control systems are limited in their ability to prevent agents from gaining control over a process chain either singly or in a small colluding group and deliberate or accidental information leak to untrusted third parties can arise as a consequence.

Assuming that an access control system can be established that satisfies the requirements outlined above, then there should be a mechanism for testing that the system is satisfactorily protected against single and n-party fraud and information leak. This requires the modelling of the access control system within a suitable analysis environment. We discuss possible environments later in this chapter.

2.5 A brief history of access control

2.5.1 Mandatory access control (MAC)

An early and significant academic development in access control was the formal mathematical presentation of security in computer systems by David Bell and Leonard LaPadula [15, 17, 16], the Bell-LaPadula model, which signifies the emergence of the subject as an academic discipline. The consequence was

the development of Mandatory Access Control (MAC) as the first major authorisation security model. MAC is an authorisation method devised for the US military based upon the US classification system and the assignment of access rights according to clearance. The Bell-LaPadula MAC model is sometimes summarised in publications by the mantra: "no read-up, no write-down" [6, 23]. Whilst various interpretations of MAC have been devised in an attempt to expand the model to non-military applications, such as that due to David Clark and David Wilson [29], the limitations of the model are apparent when it is applied to business environments, for example. The MAC model is somewhat inflexible and unsuited to situations where practical constraints, such as staff sickness and holidays, require a softening of the strict security requirements. For instance, flexibility may be required to facilitate delegation of responsibilities and the selective elevation of access rights and privileges.

2.5.2 Discretionary access control (DAC)

Butler Lampson suggested an alternative approach to the protection of computer-based resources in 1971 in a seminal article called ”Protection” [59]. In this model resource users (domains in the literature) are mapped to resources (files, programs, domains) via access rights or attributes and a mapping is maintained within an access control matrix (ACM). Typically, the Discretionary Access control (DAC) model provides flexibility of assignment of access rights to the owner of resources, hence the title. The DAC model subsequently evolved into Access Control Lists and the attributes-based system of access control, which is familiar to users of modern computer operating systems. Although DAC provides greater flexibility than MAC, it does so through dilution of the security requirement. DAC incurs scalability and management problems as the numbers of users and resources increase, particularly in respect of the ACM implementation of the model. Additionally, users do not necessarily understand their assigned rights and responsibilities and system security can be seriously undermined by the inappropriate use of root or administrator access capabilities.

2.5.3 Role-based access control (RBAC)

Whilst MAC is the generally accepted authorisation model within the military and DAC evolved into the access control system applied to the major operating systems, the academic world was shifting its attention elsewhere within the field of authorisation. Research was directed towards the formal analysis of access control systems [2] and to the development of scalable models of access control that are more appropriate to complex heterogeneous computer systems, such as Role-Based Access Control (RBAC) [47, 70, 83]. David Ferraiolo and Richard Kuhn outlined their basic RBAC model as a more appropriate system of

control in civilian government or commercial organisations than either the multilayer security of MAC or the user-centred security model of DAC [47]. Matunda Nyanchama and Sylvia Osborn extended RBAC research through modelling and analysis of roles and hierarchies and the interactions between roles within RBAC models. They developed a hierarchical role graph model for role-based access control based upon organisational hierarchies [70]. Ravi Sandhu et al proposed a family of RBAC models to provide a reference point for further RBAC development [83]. Extensions to the base RBAC model, such as role hierarchies and constraints, were outlined and discussed.

Evolution of this early work ultimately resulted in the development of a proposed standard for RBAC by Ferraiolo, Kuhn, Sandhu and others under the aegis of the National Institute of Standards and Technology (NIST) [48] and its subsequent adoption as a full standard 359–2004 by the American National Standards Institute (ANSI) [69].

The underlying principle of the RBAC model as detailed in this early work is the abstraction of resources from users via a set of roles. Consequently, the set of users is mapped many-to-many to the set of roles: a given user can occupy a number of roles and a number of users can occupy a given role. The set of roles is mapped many-to-many to the set of resources [83].

2.5.4 Policy languages based upon Datalog

Further to the early work on RBAC, numbers of research groups now targeted the development of practical models of RBAC, policy languages, the encoding of business principles and formal modelling and analysis. In particular, policy language development provided a fertile ground for researchers in formal methods and logic. Frequently, policy languages were devised from predicate calculus, particularly Horn clause subsets of predicate calculus, such as Datalog. The benefit of this approach is that, in its purest form, Datalog is decidable [13, 37] and it retains the formal proof structure, coupled with the inherited syntax and semantics, of predicate calculus. The cost of using Datalog is a restriction on its expressiveness: pure Datalog is not Turing-complete. Examples of Datalog-based languages include RT_0 and extensions [64, 63, 60] by Ninghui Li and others, DeTreville's Binder [37] and Becker and Sewell's Cassandra [14, 13], each of which is discussed below.

The RT family of policy languages represent a trust management RBAC environment. This is a superset of the SDSI 2.0 certificate handling language for authorisation developed by Ellison et al as part of their SPKI certificate theory [41]. Through language extensions devised via careful additions to the basic Datalog-style model, the expressiveness of RT is enhanced to accommodate attribute assignment, segregation of duties and delegation [63]. Negation (or prohibition in policy terms) is not facilitated.

Binder was devised as a direct interpretation of Datalog for trust management and access control to ensure polynomial time query resolution coupled with communication constructs [37]. A consequence of the tight binding to Datalog is that there are limitations to the expressiveness of the Binder language, particularly in respect of attributes, constraints and negation.

Cassandra is based upon an extension to Datalog, called Datalog with constraints [13]. Datalog with constraints is more expressive than Datalog but is only decidable if restrictions are carefully applied to the constraint domain [62]. Cassandra provides a powerful, decidable, trust-based policy language within its own, self-contained implementation, which can encode business rules such as hierarchies, delegation and limited segregation of duties. However, Cassandra cannot handle externally triggered conditions, segregation of duties based upon different agents or full negation. These limitations arise as a consequence of restricting the constraint domain to ensure decidability.

2.5.5 Policy languages not based upon Datalog

Alternatives to the Datalog approach to policy languages include Halpern and Weissman's Lithium [50], Ponder by Damianou et al [34, 33] (subsequently Ponder2) and RW by Nan Zhang, Mark Ryan, Dimitar Guelev and others [49, 104].

Halpern and Weissman used an alternative subset of predicate calculus for their Lithium policy language [50] and were able to incorporate negation. However, such flexibility was gained with a tractability cost that necessitated careful development of the language meta-logic to ensure that the system always terminates.

Instead of using a formal logic foundation to its language, the Ponder team developed a policy language from object-oriented programming principles and concentrated on producing an expressive language that is applicable to various security problems in addition to access control [34]. Whilst the language is wide-ranging and powerful, nonetheless it lacks the constructs to counter policy conflicts or the formal logical semantics that enable reasoning over the problem domain. Consequently, researchers such as Bandara, Lupu and Rosso investigated translating Ponder into a formal calculus (specifically event calculus) so that policy analysis can be performed to resolve conflicts and inconsistencies and provide methods for optimising policy rulesets [12].

RW is based upon propositional calculus with predicates such that variables are extended over a finite space [49]. As propositional calculus is decidable then RW is decidable and indeed is well suited to analysis through truth functional modelling, e.g. Binary Decision Diagrams. Whilst the RW language incorporates a simple syntax, it is capable of encoding policy rules such as permissions assignment.

However, the language is limited in its expressiveness in respect of organisational requirements such as role hierarchies and denial of permissions.

2.5.6 Access models extending RBAC

Before discussing models of access control that extend RBAC it is worth mentioning that, about the time that research teams were starting to look into RBAC, Thomas and Sandhu proposed an alternative access control approach based upon tasks [86]. This approach implies a paradigm shift away from the user toward the organisation and processes and can be construed as a precursor to workflow-based control. The model is not based upon sequences of tasks, however, but upon task and sub-task groups and there is no concept of roles.

Various research teams have responded to limitations within the basic RBAC model. In particular, most implementations of RBAC do not provide mechanisms to incorporate external influences, temporal constraints or state histories. Attempts to extend the RBAC model include the work of Elisa Bertino and others. They incorporated a formal authorisation constraint model and policy language within workflow management systems [20, 19] for the definition of full static and dynamic segregation of duties. Subsequently, Bertino et al added temporal constraints on roles with their definition of a fully specified policy language called Temporal Role-Based Access Control (TRBAC) [18]. It should be noted that the constraint language was the prime focus of the research by Bertino et al and that they did not set out to model the entire access control process, nor did they endeavour to model the full variety of workflow types beyond a straightforward sequential workflow. Later work [55] generalises the constraints language incorporating temporal constraints and cardinalities. Whilst the Generalized Temporal Role-Based Access Control (GTRBAC) represents a powerful and comprehensive language for the definition of constraints, there is no formalised modelling environment with which to analyse the security properties of the access control system as a whole.

Atluri and Huang devised a model of workflow-synchronised access control using coloured and timed Petri nets [10]. Their model, the Workflow Authorization Model (WAM), limits access to the timeframe for the enacting of tasks and incorporates time-based constraints on the performance of tasks. However, the model does not include a role-based approach and problems such as hierarchical inheritance, role-based segregation of duties and delegation are not tackled as a consequence. A further problem is that Petri nets can rapidly become very complex when used for analysing complex organisations and they do not lend themselves naturally to modelling open, mobile systems [43] and the vagaries of human behaviour.

An alternative approach was provided by Botha and Eloff who prioritised the business principle of least privilege, dynamic separation of duties, conflicting roles and users and event sequencing within the

application of access control to workflow management [25, 26, 27]. However, the access control model designed by Botha is a self-contained practical implementation that neither specifies a logically derived policy language nor provides for a formal means of analysis and testing [26].

2.5.7 W-RBAC

A review of the work by Wainer et al [98, 99, 100] reveals a comprehensive model of workflow-based role-based access control (W-RBAC). The authors properly identify key business principles, such as static and dynamic segregation of duties, that should be incorporated into a system for access control and the published work is populated with informative examples of the application of these business principles. A detailed policy and constraint language is specified, based upon logical propositions and a prolog-based implementation that is designed to be coupled to a workflow system. The model was subsequently extended to incorporate role hierarchies and delegation plus a policy metalanguage for the specification of workflow controls.

The access control model of Wainer et al addresses a number of areas that are key to my thesis. However, this work, like the work of Botha et al [25, 26, 27], Atluri and Huang [10] and Bertino et al [20, 19] does not seek to address important issues arising from access control policy design and implementation, namely reachability testing of policy rulesets and evaluation of policy rulesets against threat models, anonymity and collusion analysis.

2.5.8 Reference model for workflow systems

Crampton developed an access control model for the specification of constraints such as separation of duty that is independent of the access control implementation [31]. The model was subsequently updated to incorporate additional access control features such as delegation [32]. The key test requirement within the model is the satisfiability of any given constrained workflow. The model does not incorporate roles within the model execution itself, i.e. roles are used to preassign users to tasks and the user/task assignment is utilised within the model. Also, the model defines constraints on each pair of users in respect of task pairs. Whilst it is possible to define constraints over a more complex space using pairs, such an approach adds complexity when compared to our history-based approach.

The reference model is somewhat limited in its capabilities regarding the identification and analysis of security issues arising from the application of access control policy.

2.5.9 X-Policy

Qunoo and Ryan [78] extended the work of Zhang [104] in respect of the RW access control policy modelling framework and developed X-Policy. X-Policy is able to model dynamic systems of access control through read/write actions with preconditions called execution permissions that specify the access control policy. X-Policy utilises predicate-based atomic formulae so that a read action returns the truth values of any number of predefined atomic formulae and a write action alters the truth values of any number of predefined atomic formulae. Execution permissions are constructed from atomic formulae together with the basic logic operators: negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$) and the existential ($\exists$) and universal ($\forall$) quantifiers. Consequently, the write action provides the modelling environment with a dynamic state modifier (as execution permissions use the truth values of atomic formulae) that enables modelling of dynamic segregation of duties and least privilege. Whilst it is possible to model workflow-based access control systems using X-Policy, the declarative nature of the modelling language requires a multitude of predicates to model interactions with each task and a complex interaction of write actions and execution permissions to model task execution dependencies. Also, it is not apparent that anything other than the simplest of workflow structures such as series and parallel tasks can be modelled using X-Policy.

2.6 Business process management (BPM)

2.6.1 Introduction

The Workflow Management Coalition defines a business process as:

> *"A set of one or more linked procedures or activities which collectively realise a business objective or policy goal, normally within the context of an organisational structure defining functional roles and relationships."* [101]

Business processes provide the operational framework for organisations, e.g. companies, public bodies, charities and non-profit organisations, so that they can properly perform their day-to-day activities. A business process that is automated constitutes a workflow [101] and a computer-based system for control of workflows is called a workflow management system [101]. Numerous systems have been devised and consultant man-hours and finances invested in workflow management systems.

Various standards bodies have developed frameworks for the definition of business processes and workflows in XML-based languages such as BPML, BPEL4WS and XPDL [97, 96]. To date, the situation has been

a confused mixture of standards and standard setters and a battle for supremacy has ensued based upon adoption and support of standards by the major corporate players in the industry. Currently (March 2011), the major players IBM, Microsoft, SAP, Siebel, BEA are implementing their jointly authored BPM language BPEL4WS version 1.1 within their BPM products [7]. The Organization for the Advancement of Structured Information Standards (OASIS) has developed a full standard for the next iteration called WSBPEL version 2 [5]. It is apparent that BPM is seen to be an important topic within the corporate systems industry.

The industry motivation has been to produce a web services style XML language for the development and management of BPM models. From an academic viewpoint, the focus has been more upon frameworks for modelling, analysing and testing workflow systems that are abstracted from the practical implementation. Wil van der Aalst and colleagues formalised workflow modelling through the application and extension of Petri nets [94, 88] and proposed a set of patterns in workflows as defined in the higher-order Petri nets [93, 79].

An alternative approach to workflow modelling and analysis was presented by Cook et al [30, 56] who devised a programming language called Orc for orchestrating tasks. Using Orc, Cook et al modelled the set of workflow patterns proposed by van der Aalst as discussed above. Meanwhile, Puhlmann and Weske [77] demonstrated that it is possible to use the pi calculus as a modelling environment for workflows including the encoding of van der Aalst's workflow patterns.

There has been considerable debate concerning alternative modelling environments for workflows. This is exemplified by a published article in 2004 by Smith and Finbar [85] in which the authors argue enthusiastically that the pi calculus provides an ideal environment for the construction of a Business Process Management System (BPMS) of which workflow management is a subset. Van der Aalst countered this argument with a couple of publications [92, 90] in which the assertions concerning the pi calculus (particularly the use of the calculus as a basis for BPML — the Business Process Modeling Language) are rebutted and examples of workflows are proposed that are easily modelled in Petri nets but are difficult in the pi calculus.

The various approaches to workflow modelling are still battling for supremacy and choosing between the alternatives is likely to be application specific or a matter of personal preference.

A discussion of the various modelling environments is set out below.

2.6.2 Petri nets

Carl Adam Petri devised Petri nets in the 1960s [74] as a graphical yet mathematically rigorous method for analysing states and state transitions in distributed systems [11]. The basic components of a Petri net are places or states (S), transitions (T) and edges (F). A net (N) is defined as a tuple $N = (S, T, F)$ and the dynamics of the net are represented by the placement of tokens in the state spaces. A simple collection of rules is defined for Petri nets: a.) all edges are directed, b.) state-to-state and transaction-to-transaction connections are not permitted and c.) states can hold zero or more tokens. If a transition contains tokens on all of its inputs, the transition "fires" such that tokens are decremented by one unit on each input and tokens are incremented by one unit on each output [89]. Given this simple framework and an extensive theoretical foundation then Petri nets provide a powerful tool for the modelling of complex distributed system state dynamics over static nets.

Although Petri nets are a powerful analytical tool nonetheless they suffer from a number of limitations when applied to realistic systems. They can become large, complex and unwieldy, they cannot handle time-constrained transitions, they cannot test whether or not a state space is empty and they do not satisfactorily represent the movement through nets of complex components [91]. Consequently, various additions to basic Petri net theory have been proposed to extend its descriptive power. Jensen et al developed the coloured Petri net model [54, 58], extending the initial work of Zervos [103] and others. Coloured Petri nets provide for the tokens of basic Petri nets to be assigned values and the rules governing transitions are expanded accordingly. Tokens can now be assigned to identifiable objects within a net such as the documents that flow through a purchase ledger order system. Van der Aalst [91] extended the work of van Hee and others [95] by invoking time delay management in Petri nets through the assignment of timestamps to all tokens in a net. The timestamp in this model is an indicator of a token's availability and a transition is ready to fire when the maximum timestamp (the "enabling time" in the literature) for the transition inputs is attained. Huber, Jensen and Shapiro proposed a further extension to coloured Petri nets [53], namely hierarchies. The hierarchical coloured Petri net enables reuse of workflow patterns and the ability to construct high-level summary workflows with places representing lower level workflows.

The extended, higher order Petri nets have been developed to provide a suitable environment for the modelling of workflows. They address the workflow-modelling problem by providing an intuitive visual analysis tool that is formally specified with a sound theoretical framework.

However, there are still deficiencies:

2.6.2.1 Deficiencies in Petri nets as a tool for modelling workflows (and access control)

Petri nets do not provide a satisfactory environment for the modelling of mobile or dynamic processes or workflows. Eshuis and Dehnert pointed out that Petri nets model closed active systems whereas workflow management systems tend to be open and reactive [43]. Consequently, whilst it may be possible to model a workflow and test the system with token dynamics, the real-world enactment of the workflow may involve externally triggered conditions on transitions between states, such that the real-world and model dynamics differ considerably. In particular, practical workflows (and indeed access control systems) involve the interaction with human agents with all of the decision-making uncertainty and irrationality that this entails.

Oren and Haller [72] suggested that modelling activities, environmental events and data within Petri nets can be achieved in various ways and each method causes problems with workflows. Summarising their arguments: modelling activities as transitions leads to a lack of definition between descriptive and prescriptive behaviours i.e. between workflow activities and workflow management, whilst modelling activities as places is impractical as activities change workflow parameters whereas places are merely state descriptors. Also, modelling of external trigger events as transitions blurs the difference between events and activities, whilst modelling events as places and tokens yields functional problems with communication of these events.

Global constraints and cancellation events are difficult to model as they extend over the whole or some fraction of the whole of a workflow and not just over local transitions or states [72]. In effect, modelling workflows under global constraints requires a set of different workflows each satisfying all of the permutations under the global constraint. Alternatively, it requires a complex rat's nest of edges to apply the constraints to all transitions and places as appropriate.

2.6.3 Process calculi

Robin Milner developed the pi calculus, a calculus for concurrent processes, as an extension of his earlier work with the Calculus of Communicating Systems (CCS) [66, 67]. Milner's CCS was itself influenced by the earlier work of Hoare with his Communicating Serial Processes (CSP) calculus [52]. The pi calculus was built upon the work of others, notably the labelling semantics of the Extended CCS of Engberg and Nielson [42]. Milner introduced the capability within the calculus to facilitate communication of the names of channels, whilst simplifying the naming conventions and retaining the underlying theoretical formalism. Consequently, the pi calculus provides a modelling environment for dynamic processes.

The pi calculus is Turing complete as demonstrated by Milner in his famous 1989 paper "Functions as

Processes" [65] through the encoding in the pi calculus of the Turing complete λ calculus. In addition, the calculus is syntactically compact so although it is possible to model workflows or security, say, within the calculus such encodings may be neither efficient nor human comprehensible. Consequently, various extensions to the pi calculus have been proposed in response to specific problem spaces so as to provide a more appropriate syntactic framework, whilst retaining the underlying theory and analysis tools. Some extensions to the pi calculus have been devised to provide an environment for modelling security protocols such as the applied pi calculus [2, 3] or access control [28] and other calculi have been specified to tackle wider ranging problems such as biochemical molecular processes [76]. Workflows and workflow patterns have been tackled by Pulhmann and Weske [77].

We consider pi calculus and its derivatives to be suitable for modelling workflows. Indeed a pi calculus environment tackles the areas where Petri nets are observed to be weak given that the calculus was devised for distributed and mobile environments. However, the calculus environment lacks the user-friendly visual appeal of Petri nets and depending upon the calculus formalism used, then certain workflow patterns and constructs can be very complex and unwieldy to model within the calculus.

The applied pi calculus of Abadi and Fournet [3] extends the pi calculus to incorporate functions and names. These additional features, when combined with an equational theory, enable modelling of complex processes in a more succinct style whilst retaining a powerful underlying theory. To our knowledge the applied pi calculus has not been used to model BPM. However, the applied pi calculus is a promising candidate, not only for modelling BPM but also, and crucially to our thesis, the modelling of workflow-based access control.

2.6.4 Specialised languages and formalisms

An alternative approach to workflow modelling is briefly discussed in section 2.6.1 above, namely the formulation of a programming language such as Orc [56]. Whilst the language was developed primarily as a programming style environment, work is under way to provide a formal theoretical framework and semantics [56].

Given that Orc was designed with workflow modelling in mind, then it is reasonable to assume that there is a good fit between the two and this is demonstrated by the encoding of van der Aalst's workflow patterns by Cook et al [30]. However, the language lacks the extensive theoretical foundations of Petri nets and process calculi and this could be problematic when attempting to test and verify the dynamics of workflow formalisms.

2.6.5 Other approaches — UML

Other methods for modelling workflows have been used, most notably approaches using the Unified Modeling Language (UML) and associated diagrammatic tools. UML is a loosely coupled modelling framework with a large syntax that allows for individual interpretation and application of constructs to model development [36]. Efforts have been made to define, for example, the workflow patterns in UML2.0 in comparison with the Business Process Modeling Notation (BPMN) (a diagrammatic environment for the visualisation of business procedures) [102], or as a test of expressiveness in respect of workflow patterns [39]. Whilst these investigations demonstrate that such diagrammatic tools can be used to visually describe workflows and in the case of BPMN, in a business-user, non-technical way, these methods provide no mechanism for a formal analysis of workflows for reachability or redundancy, say.

Work by Derrick, Boiten et al was directed at formalisation within the UML architecture, specifically in respect of conformance, consistency, unification and refinement [36, 24]. This approach provides a theoretical framework for UML model analysis, particularly in respect of the matching of UML models to programs (conformance), consistency checking between diagrams within a UML model and consistency within successive UML model specifications (refinement). The framework provides a suitable environment for analytical project development. However, we consider that the applied pi calculus is currently better suited to analysing the security properties of workflows and access control for our purpose, through the use of observational equivalence and an underlying equational theory.

2.7 The interaction between BPM and access control

We discuss access control in section 2.3 and an extension of access control to a workflow-based implementation is outlined in section 2.5.6 et seq. We can model BPM as workflows and if we couple the BPM workflow models with RBAC together with a mechanism for specifying and implementing constraints, we can implement a business process access control modelling environment as discussed in the next section.

2.8 Business process access control (BPAC)

Business processes generally comprise a collection of discrete processes called tasks that are connected by the control flow of documents and authorisations (or their virtual equivalents within a computer system). A task may represent an activity (or group of activities) that has to be performed by some agent. The agent can be either a human, e.g. an employee within an organisation, or a computer process. Within the

context of our access control model we consider the task to be associated with a set of resources that are necessary for the satisfactory completion of that task by an agent. Consequently, when an agent gains access to a particular task then they also gain access to the set of resources associated with that task. Workflows and tasks represent an abstraction layer between agents and resources that:

- enables the simplified modelling of access control without the complexity of multiple agent/resource allocations — tasks can be treated as atomic processes,
- allows resources to be updated and altered in respect of specific tasks without affecting the access control model,
- enables proper implementation and modelling of business rules such as segregation of duties,
- couples access control management to BPM so that it can be considered to be an additional component of BPM modelling and implementation and
- enables modelling of access control through the use of process algebra.

Given this BPM-based modelling environment, access control can then be implemented by traditional means, such as role-based access control using static agent/role assignments coupled with dynamic constraints handling, applied to agent/task or agent/workflow interactions.

Having abstracted access control away from resources we can devise a modelling environment for access control that enables us to analyse and test possible access control policies within a BPM context. We call this Business Process Access Control (BPAC) and we model it using the applied pi calculus to provide us with a powerful analytical tool for interactive processes. Our justification for using the applied pi calculus as the basis for our model is presented in the next section.

2.9 Applied pi calculus as the basis for BPAC

The previous sections provide motivation for the adoption of a process-centric workflow-based approach to access control specification and modelling. Now we turn our attention to the selection of a suitable modelling environment for BPAC.

The key requirements for a suitable modelling environment are as follows:

- the capability to handle modelling of complex workflows, including complex interrelations between tasks,

- a formal underlying theory that provides suitable tools for security testing of workflow-based access control models,
- a comprehensive syntax that can be used to model interactions between entities and can replicate predicates that represent properties of entities or associations between entities,
- the potential for automation of modelling and analysis via existing or future software tools,
- the scope for expanding models around the basic access control model to incorporate additional model properties such as inter-agent communication,
- a well established security model,
- abstraction from specific programming environments enabling translation into a variety of different computer-based implementations.

Based upon the contents of the previous sections, particularly the discussion concerning BPM in section 2.6, we consider the applied pi calculus of Abadi and Fournet [3] to be suitable for BPAC modelling. The applied pi calculus satisfies all of the requirements listed above and because it is an extension of the pi calculus then it retains the capability to handle Puhlmann and Weske constructs for modelling workflows [77]. We favour the use of the applied pi calculus when coupled with workflow constructs as the basis for our BPAC modelling environment over the more traditional Horn clause Datalog approach, Prolog implementations or Petri nets. None of the alternatives are as capable at providing the power and flexibility of the applied pi calculus, particularly in respect of its theoretically underpinned analytical potential.

2.10 Summary

In this chapter we present a description of access control and business rules. We follow this with an outline of the history of research in access control, then a discussion of BPM and possible modelling environments for same. Drawing these ideas together, we present the motivation for our access control modelling environment called Business Process Access Control (BPAC) and we outline our justification for the selection of the applied pi calculus as the modelling environment for BPAC. We discuss BPAC in greater detail in the next chapter.

Chapter 3

Business process access control (BPAC)

3.1 Introduction

In this chapter, we define formally our understanding of workflows and business process access control (BPAC). We introduce the security problems in respect of BPAC that we investigate in subsequent chapters via our modelling environment. Throughout, we present a trivial workflow example to demonstrate the operation of BPAC and we discuss the security problems within the context of this trivial model. Following this discussion we present two access control examples and we review some examples of coding using protocol examples from the literature.

3.2 Workflows

3.2.1 A formal presentation of workflows

For our formal definition of a workflow we utilise the definitions of Eshuis and Kumar [44] with minor adaptations for our requirements. Specifically, we change the nomenclature so that it is consistent with our model definitions and we add constraints that ensure that conjunctive (resp. disjunctive) splits are matched to conjunctive (resp. disjunctive) joins. We define a workflow as a directed graph comprising tasks as graph nodes connected by edges. We identify seven different types of task: a task immediately followed by two or more parallel tasks is a conjunctive (AND) split task and a task that follows two or more parallel tasks is a conjunctive (AND) join. A task that involves a choice in respect of the subsequent paths is represented by a disjunctive (XOR) split. If a task action is triggered by the action of one of a number of preceding parallel tasks then the task is represented as a disjunctive (XOR) join. A task

that is preceded by a single task and followed by a single task is a sequential task. The task at the commencement of the workflow is the start task, t_s and the terminating task is the end task, t_e.

Definition 2

A workflow is a directed graph $W = (T, F)$ *where* T *is a set of tasks as nodes and* $F \subseteq T \times T$ *is a precedence relation.*

T *is a disjoint set of XOR splits* $T_{\oplus s}$*, XOR joins* $T_{\oplus j}$*, AND splits* $T_{\wedge s}$*, AND joins* $T_{\wedge j}$*, sequential nodes* T_{seq} *and* $\{t_s, t_e\}$ *where* t_s *is the start node and* t_e *is the end node.*

Auxiliary functions are $inedge, outedge : T \longrightarrow \mathcal{P}(F)$*. Given a node* $t \in T$*, let* $inedge(t) = \{(x, y) \in F \mid y = t\}$ *and* $outedge(t) = \{(x, y) \in F \mid x = t\}$.

3.2.2 Structural constraints

Each workflow should satisfy the following constraints:

- the start node has no incoming edge and at least one outgoing edge

  $|inedge(t_s)| = 0 \wedge |outedge(t_s)| \geq 1$;

- the end node has at least one incoming edge and no outgoing edge

  $|inedge(t_e)| \geq 1 \wedge |outedge(t_e)| = 0$;

- each sequential node has one incoming and one outgoing edge

  $\forall t \in T_{seq} : |inedge(t)| = |outedge(t)| = 1$;

- each split node has one incoming and two or more outgoing edges

  $\forall t \in T_{\wedge s} \cup T_{\oplus s} : |inedge(t)| = 1 \wedge |outedge(t)| > 1$;

- each join node has two or more incoming and one outgoing edge

  $\forall t \in T_{\wedge j} \cup T_{\oplus j} : |inedge(t)| > 1 \wedge |outedge(t)| = 1$;

- all AND splits have a corresponding AND join

  If $t_{\wedge s} \in T_{\wedge s}$, $t_{\wedge j} \in T_{\wedge j}$ then

  $\exists t \in T : t_{\wedge s} F^* t \wedge t F^* t_{\wedge j}$

  where F^* denotes the reflexive-transitive closure of F;

- all XOR splits have a corresponding XOR join

  If $t_{\oplus s} \in T_{\oplus s}$, $t_{\oplus j} \in T_{\oplus j}$

  $\exists t \in T : t_{\oplus s} F^* t \wedge t F^* t_{\oplus j}$

  where F^* denotes the reflexive-transitive closure of F;

- each node is on a path from the start to the end node

  $\forall t \in T : t_s F^* t \wedge t F^* t_e$.

3.2.3 Instance subgraphs

Instance subgraphs represent the runtime behaviour of the workflow. Instance subgraphs are built inductively from a start node and the construction rules ensure that all active conjunctive parallel paths and single active disjunctive parallel paths are included within the summary of runtime behaviour. A valid and correct subgraph is defined as follows:

Definition 3

Let (T, F) *be a workflow graph. An instance subgraph is a tuple* (T', F') *such that:*

1. $T' \subseteq T$;
2. $F' \subseteq F$;
3. $t_s \in T'$ *and* $outedge\,(t_s) \subseteq F'$;
4. $t \in T'$ *and* $t \in T_{seq} \Rightarrow outedge\,(t) \subseteq F'$;
5. $t \in T'$ *and* $t \in T_{\wedge s} \cup T_{\wedge j}$ *and* $inedge\,(t) \subseteq F' \Rightarrow outedge\,(t) \subseteq F'$;
6. $t \in T'$ *and* $t \in T_{\oplus s} \cup T_{\oplus j}$ *and* $|inedge\,(t) \cap F'| = 1 \Rightarrow |outedge\,(t) \cap F'| = 1$;
7. $t \in T'$ *and* $t \in T_{\wedge j}$ *then* $inedge\,(t) \subseteq F'$
8. $t \in T'$ *and* $t \in T_{\oplus j}$ *then* $|inedge\,(t) \cap F'| = 1$
9. $(x, y) \in F' \Rightarrow x, y \in T'$;
10. $t \in T'$ *and* $inedge\,(t) \cap F' = \varnothing \Rightarrow t = t_s$.

The final definition ensures that only valid instance subgraphs are allowed by ensuring that the only task node within the subgraph that has no incoming edge is the start node.

3.2.4 Terminating instance subgraph

A terminating instance subgraph is a subgraph that includes the end node t_e. The existence of a terminating instance subgraph for a given workflow indicates that a path can be found from a start node to an end node for that workflow.

Definition 4

Let $W_{sub} = (T', F')$ be an instance subgraph for the workflow graph $W = (T, F)$ and let $t_e \in T$ be an end node. If $t_e \in T'$ and $inedge(t_e) \subseteq F'$ then W_{sub} is a terminating instance subgraph of the graph W.

3.3 Business process access control (BPAC)

In chapter 2 we introduce workflow-based access control as an extension of role-based access control. In this section we present a formal definition of business process access control (BPAC), our interpretation of workflow-based access control, followed by formal descriptions of the security properties: completeness, satisfiability, collusion and anonymity.

3.3.1 Constraints

We define constraints as per Li and Mitchell [62]. Firstly, we define a constraint domain, then we define constraint expressions as a conjunction of primitive constraint predicates within the constraint domain. Later in this chapter, we discuss constraint tests and constraint commands that provide tools for the manipulation of these constraint expressions.

3.3.2 Constraint domain

A constraint domain for a workflow W provides the toolkit with which constraint rules assigned to tasks within the workflow can be constructed. We define a constraint domain Φ as a collection of predicates, functions, names and variables together with their associated mappings and relations.

Definition 5

Given a workflow $W = (T, F)$, a set of agents A and a set of roles R then a constraint domain Φ over the workflow W is a 3-tuple $(\Sigma, \mathcal{D}, \mathcal{L})$. Σ is a signature comprising a set of constants and a collection

of predicate and function symbols, each with an associated "arity", indicating the number of arguments to the symbol. $\mathcal{D}$ is a Σ-structure comprising the following: a set D called the universe of the structure that includes all workflow task names i.e. $T \subseteq D$, the workflow name W and an infinite set of variables, a mapping from each constant to an element in D, a mapping from each predicate symbol in Σ of degree k to a k-ary relation over D and a mapping from each function symbol in Σ of degree k to a function from D^k to D. $\mathcal{L}$ is a set of quantifier-free first-order formulas over Σ, called the primitive constraints of the domain.

3.3.2.1 Constraint domain specifics

We assume that the unary predicate symbol $\neg$ is contained in Σ and is interpreted as negation in $\mathcal{D}$. We also assume that $\top$ (true) is in $\mathcal{L}$ and that $\mathcal{L}$ is closed under variable renaming.

3.3.3 Constraint expression

A constraint expression (referred to simply as a constraint) is a conjunctive formula associated with a task in a workflow. If the formula is satisfied by an agent together with the matching of the agent's role to the task role then the agent is permitted access to the task. We define a constraint expression in respect of a constraint domain Φ. The constraint expression is built from primitive constraint predicates in conjunction with the negations of primitive constraint predicates formulated over signature Σ within the constraint domain Φ.

Definition 6

Let Φ be a constraint domain. A constraint expression (in variables $x_1, \ldots, x_k$) is a finite conjunction $c = \varphi_0 \wedge \ldots \varphi_i \ldots \wedge \varphi_n \wedge \neg\psi_0 \wedge \ldots \neg\psi_j \ldots \wedge \neg\psi_m$ where each φ_i, $0 \leq i \leq n$, ψ_j, $0 \leq j \leq m$ is a primitive constraint predicate without negation in Φ. Further, the variables in φ_i and ψ_j are among $x_1, \ldots, x_k$.

A constraint expression c is alternatively described in set notation as:

$c = \{\varphi_0, \ldots, \varphi_i, \ldots, \varphi_n, \neg\psi_0, \ldots, \neg\psi_j, \ldots, \neg\psi_m\}$.

The set of all constraint expressions for a particular workflow W is C.

3.3.4 Evaluating constraints against a store

Our BPAC environment uses a memory of constraint terms called a store that is updated on a task-by-task basis. Evaluation of constraints against the store forms the basis of our constraints testing. We

represent a constraint store as a world and the store can contain primitive constraint terms without negation formulated over Σ within the constraint domain Φ of section 3.3.2.

Definition 7

Let $\mathbf{W}$ *be a set of worlds and let* $\mathcal{W} \in \mathbf{W}$. *If* φ, ψ *are primitive constraint predicates without negation and* c *is a constraint expression with no free variables then we define* $[[c]]_{\mathcal{W}}$ *by induction over the structure of* c *as follows:*

$$[[\varphi]]_{\mathcal{W}} = \begin{cases} \top & if\ \varphi \in \mathcal{W} \\ \bot & if\ \varphi \notin \mathcal{W} \end{cases}$$

$$[[\neg\psi]]_{\mathcal{W}} = \begin{cases} \top & if\ \psi \notin \mathcal{W} \\ \bot & if\ \psi \in \mathcal{W} \end{cases}$$

and if c_1 *and* c_2 *are constraint expressions then:*

$[[c_1 \wedge c_2]]_{\mathcal{W}} = [[c_1]]_{\mathcal{W}} \wedge [[c_2]]_{\mathcal{W}}$.

3.3.5 Workflow-based access control

Having defined the basic workflow we now incorporate access control within our workflow model. We utilise roles as per our definition of RBAC in section 2.3 and we assign the roles to agents and tasks. Consequently, for an agent to be allowed access to task resources an agent's role must match a task role. Additionally, we assign constraints as per section 3.3.3 to tasks that further restrict the access of agents to tasks and these constraints can be global or local and based upon static or dynamic information.

We extend workflow W to a 7-tuple to incorporate access control components. Our definition incorporates a task/role assignment function and a set of constraints together with a task/constraint assignment function that specify the conditions under which agents can be assigned to tasks. Also, we include a set of worlds wherein a world represents a store for constraint terms and a transition function that updates worlds.

Definition 8

We define an access control workflow as a 7-tuple $W = (T, F, TR, C, TC, \mathbf{W}, \delta)$ *where* T *is a set of tasks,* $F \subseteq T \times T$ *is a precedence relation,* $TR : T \longrightarrow R$ *is a task/role assignment function,* C *is a set of*

constraints as defined in section 3.3.3, $TC : T \times A \longrightarrow C$ *is a task/agent/constraint assignment function,* $\mathbf{W}$ *is a set of worlds associated with the workflow and* $\delta : T \times \mathbf{W} \longrightarrow \mathbf{W}$ *is a transition function. We define* R *as a set of roles and* A *as a set of agents. We represent agent/role assignment by the finite binary relation* $AR \subseteq A \times R$.

3.3.6 Terminating instance subgraph

We extend the definition of a terminating instance subgraph W_{sub} to a 7-tuple to incorporate access control components. As in the previous definition our definition incorporates a task/role assignment function, a set of constraints that specify the conditions under which agents can be assigned to tasks, a task/agent/constraint assignment function, a set of worlds and a transition function. The existence of a terminating instance subgraph for a workflow indicates that a path can be found from a start node to an end node for that workflow given the access control conditions.

Definition 9

Let $W = (T, F, TR, C, TC, \mathbf{W}, \delta)$ *be an access control workflow and let* t_e *be an end node then if there exists an instance subgraph* $W_{sub} = (T', F', TR, C, TC, \mathbf{W}, \delta)$ *such that* $T' \subseteq T$, $F' \subseteq F$, $t_e \in T'$ *and* $inedge(t_e) \subseteq F'$ *then* W_{sub} *is called a terminating instance subgraph for graph* W.

3.3.7 Assignment function (AW)

The assignment function AW represents the allocation of agents, subject to role assignments and constraints, to every task in a workflow.

Definition 10

Let workflow be $W = (T, F, TR, C, TC, \mathbf{W}, \delta)$. *An assignment function* $AW : T \rightarrow A$ *is called valid in set of worlds* $\mathbf{W}$ *if*

1. *let* AR *be an agent/role assignment relation,* TR *be a task/role assignment function and* a *be* $AW(t)$ *then for all* $t \in T$ $\exists a$ *such that* $a\, AR\, TR(t)$,
2. *let* TC *be a task/agent/constraint assignment function then for any task* t *and world* $\mathcal{W}$ *if* $[[TC(t, AW(t))]]_{\mathcal{W}} = \perp$ *then* $\delta(t, \mathcal{W}) = \mathcal{W}$.

3.3.8 Satisfiable assignment function (SAW)

The definition is similar to AW except that SAW is not a total function over T. We define the satisfiable assignment function (SAW) in respect of a terminating instance subgraph i.e. a path from t_s to t_e .

Definition 11

Let terminating instance subgraph be $W_{sub} = (T', F', TR, C.TC, \mathbf{W}, \delta)$. *A satisfiable assignment function* $SAW : T' \rightarrow A$ *is called valid in set of worlds* $\mathbf{W}$ *if*

1. *let* AR *be an agent/role assignment relation,* TR *be a task/role assignment function and* a' *be* $SAW(t)$ *then for all* $t' \in T'$ $\exists a'$ *such that* $a'\ AR\ TR(t')$,
2. *let* TC *be a task/agent/constraint assignment function then for any task* $t' \in T'$ *and world* $\mathcal{W}$ *if* $[[TC(t', SAW(t'))]]_{\mathcal{W}} = \perp$ *then* $\delta(t', \mathcal{W}) = \mathcal{W}$.

3.3.9 The transition function

The transition function δ provides for the updating of worlds in respect of workflow tasks to represent the dynamic modification of the constraint store. We present an example of the transition function that we subsequently use in our BPAC environment.

3.3.9.1 An example of the transition function

We devise an operator called 'test' that identifies whether or not a constraint expression associated with a particular task/node is satisfied by an agent:

we use the syntax $(t, test(c(a), \mathcal{W}))$ in our BPAC model to represent the verification of a constraint expression assigned to task t for some agent $a \in A$ against the store $\mathcal{W}$ where $c : A \longrightarrow \{\top, \perp\}$ is a constraint expression and $(t, test(c(a), \mathcal{W})) = [[TC(t, a)]]_{\mathcal{W}}$.

We devise operators called 'write' and 'delete' using the constraint syntax that interact with the store in respect of specific task nodes. The operators allow us to modify the store contents by adding or removing constraint expressions to or from the current world. For example if:

$(t, write(\{c_1, \ldots, c_n\}, \mathcal{W})), (t, test(c(a), \mathcal{W})), (t, delete(\{d_1, \ldots, d_m\}, \mathcal{W})), (t, write(\{e_1, \ldots, e_p\}, \mathcal{W}))$ is a list of operators called the policy rule assigned to task t where $c(a)$ is a constraint expression and $c_1, \ldots, c_n$, $d_1, \ldots, d_m$ and $e_1, \ldots, e_p$ are primitive constraint predicates without negation then:

$\delta(t, \mathcal{W}) = (((\mathcal{W} \cup \{c_1, \ldots, c_n\}) \setminus \{d_1, \ldots, d_m\}) \cup \{e_1, \ldots, e_p\})$

We use the list of constraint operators outlined above on a task-by-task basis to construct a workflow constraint policy within a BPAC workflow model.

3.3.10 Constructing the BPAC model

Based upon the definitions presented above the implementation of the full BPAC model comprises the following components:

- sets of tasks T, agents A and roles R are identified for the particular example;
- a workflow W is constructed from the tasks, adding dummy tasks if necessary such as a terminating task t_e to ensure that the workflow structure is consistent with the workflow definition;
- agent/role (AR) and task/role (TR) assignments are specified;
- task/constraint (TC) assignments are specified. Constraint expressions attached to tasks are determined using the policy language summarised below and contain a variable (usually represented as a) that takes an agent identity value at runtime.
- We assume by default that the store is always updated once an agent has been matched to a task and we do not therefore explicitly specify commands for these store updates. That is, a $(t, write(hasAccessed(a, t, W), \mathcal{W}))$ command is associated with all tasks in a workflow model. Additional store update commands e.g. $(t, write(\{c_1, \ldots, c_n\}, \mathcal{W}))$, $(t, delete(\{d_1, \ldots, d_m\}, \mathcal{W}))$ and $(t, write(\{e_1, \ldots, e_p\}, \mathcal{W}))$ as per section 3.3.9.1 above are assigned to tasks in order as necessary using the policy language summarised below. A collection of commands for an entire workflow constitutes the workflow constraint policy.

3.3.11 BNF for constraint handling in the BPAC environment

The policy language that we use for constraint handling in our BPAC modelling environment consistent with the workflow-based access control definitions outlined above is presented in table 3.1. A workflow constraint policy is built from optional constraint policies devised for each workflow task. Task constraint policies comprise a sequence of optional constraint operators: a 'write' rule followed by a 'test' rule then a 'delete' rule and finally another 'write' rule. We use the rule order to indicate that an agent constraint term can be written to the store either before or after a constraint test is performed. In practice, a 'write' before 'test' can occur if an agent is attempting to access a task and has satisfied the task/role

⟨*workflow constraint policy*⟩	::=	(⟨*task constraint policy*⟩[","])*
⟨*task constraint policy*⟩	::=	[⟨*write policy rule*⟩][","][⟨*test policy rule*⟩][","] [⟨*delete policy rule*⟩][","][⟨*write policy rule*⟩]
⟨*test policy rule*⟩	::=	"(" ⟨*task*⟩ "," ⟨*test formula*⟩ ")"
⟨*write policy rule*⟩	::=	"(" ⟨*task*⟩ "," ⟨*write formula*⟩ ")"
⟨*delete policy rule*⟩	::=	"(" ⟨*task*⟩ "," ⟨*delete formula*⟩ ")"
⟨*test formula*⟩	::=	⟨*test*⟩ "(" ⟨*conjunction*⟩ ["," ⟨*store*⟩] ")"
⟨*write formula*⟩	::=	⟨*write*⟩ "(" ⟨*positive conjunction*⟩ ["," ⟨*store*⟩] ")"
⟨*delete formula*⟩	::=	⟨*delete*⟩ "(" ⟨*positive conjunction*⟩ ["," ⟨*store*⟩] ")"
⟨*positive conjunction*⟩	::=	⟨*positive conjunction term*⟩ (⟨*and*⟩ ⟨*positive conjunction term*⟩)*
⟨*conjunction*⟩	::=	⟨*conjunction term*⟩ (⟨*and*⟩ ⟨*conjunction term*⟩)*
⟨*conjunction term*⟩	::=	⟨*predicate term*⟩ | ⟨*negation*⟩ ⟨*predicate term*⟩
⟨*positive conjunction term*⟩	::=	⟨*predicate term*⟩
⟨*predicate term*⟩	::=	⟨*predicate*⟩ "(" ⟨*base term*⟩ ")"
⟨*base term*⟩	::=	⟨*variable*⟩ ("," ⟨*variable*⟩)*
⟨*variable*⟩	::=	⟨*Id*⟩
⟨*predicate*⟩	::=	⟨*Id*⟩
⟨*task*⟩	::=	⟨*Id*⟩
⟨*store*⟩	::=	⟨*Id*⟩
⟨*test*⟩	::=	"*test*"
⟨*write*⟩	::=	"*write*"
⟨*delete*⟩	::=	"*delete*"
⟨*negation*⟩	::=	"**not**" | "¬"
⟨*and*⟩	::=	"**and**" | "∧"
⟨*Id*⟩	::=	⟨*letter*⟩ (⟨*letter*⟩ | ⟨*digit*⟩ | "_" | "−")*

Table 3.1: BNF for constraint handling in BPAC

requirements but has not yet satisfied the constraint requirements. Alternatively, a 'write' after 'test' indicates that the agent has fulfilled the constraint terms prior to an update of the store. A case in which 'write' before 'test' is used is the provision of a concurrent access constraint for parallel tasks. In order to ensure that an agent can access no more than one concurrent task we 'write' before 'test' an access constraint 'canAccess' in respect of a given task and we test for this constraint in the parallel tasks. An example of 'write' before 'test' is given in chapter 4.

3.3.12 Example - a simple 2-task workflow

Let us consider a very simple example of a BPAC model.

Let W be a workflow comprising two tasks $Task_1$ and $Task_2$ that are linked, perhaps, by the flow of an authorisation document from $Task_1$ to $Task_2$ so that $Task_2$ cannot commence until $Task_1$ has been completed and the correct documentation has been transferred from $Task_1$ to $Task_2$ (note that flow in this context might be the physical or computer-based transfer of a document from one agent to another or the placement of a document in a specific location for subsequent take-up by another agent). These tasks can be identified respectively as the initial task or node, t_s, followed by the final task, t_e, as per our

workflow definition above. Each task has a selection of resource access requirements that are associated with the task process. We have a set of agents $\{Agent_1, Agent_2, Agent_3\}$ and each agent is assigned a role based upon their position within the organisation. For the purpose of this simple example we have a set of available roles $\{clerk, supervisor, manager\}$. We represent agent/role assignment by the binary relation of agent/role assignments as follows:

$$\begin{aligned} AR \quad = \quad & \{(Agent_1, clerk)\,, \\ & (Agent_2, supervisor)\,, \\ & (Agent_3, manager)\} \end{aligned}$$

So agent $Agent_1$ has role $clerk$, agent $Agent_2$ has role $supervisor$ and agent $Agent_3$ has role $manager$. Next we define our access control policy as follows mapping each task $Task_i$ to its role via the task/role assignment function:

$$TR \quad = \quad \{(Task_1, clerk)\,, (Task_2, supervisor)\}$$

$(Task_1, clerk)$ models the fact that the agent mapped to task $Task_1$ must have role $clerk$ and $(Task_2, supervisor)$ the agent mapped to $Task_2$ must have role $supervisor$.

We associate a set of worlds representing a dynamic constraint store, $\mathbf{W}$, with the workflow W.

Over this RBAC-based access control model we consider dynamic segregation of duties constraints mapped to tasks as follows wherein a is the only variable:

$$(Task_2, test\,(\neg hasAccessed\,(a, Task_1, W)\,, \mathcal{W}_2))$$

as the workflow constraint policy i.e. the agent mapped to task $Task_2$ cannot have been mapped to task $Task_1$ in workflow W given constraint store $\mathcal{W}_2$ associated with the task.

If we consider this simple example then a valid assignment function AW and a valid satisfiable assignment function SAW for the workflow in set of worlds $\mathbf{W}$ both exist and are represented by the following graph:

$$\{(Task_1, Agent_1)\,, (Task_2, Agent_2)\}$$

with the delta function for each task based upon an initial store/world $\mathcal{W}_1 = \varnothing$ being:

$$\begin{aligned}
\delta\left(Task_1, \mathcal{W}_1\right) &= \{hasAccessed\left(Agent_1, Task_1, W\right)\} \\
\delta\left(Task_2, \mathcal{W}_2\right) &= \{hasAccessed\left(Agent_1, Task_1, W\right), \\
&\quad hasAccessed\left(Agent_2, Task_2, W\right)\}
\end{aligned}$$

as the store is implicitly updated for each task with constraint terms indicating which agent has accessed the task at runtime.

In workflows that are constructed from sequences and/or conjunctive parallelism there is no difference between satisfiable assignment functions and assignment functions. Differences arise when the workflow incorporates disjunctive parallelism and/or loops.

3.4 Security properties of workflows

3.4.1 Discussion of security properties associated with the 2-task workflow

Given the simplicity of the 2-task workflow model it is a straightforward observational exercise to identify the security properties of the workflow. We return to this example in chapter 8 where we analyse the security properties in a more formal manner using applied pi calculus tools.

3.4.2 Completeness of BPAC policies

Completeness as defined for our modelling environment is an example of the reachability property in which all components of a workflow-based access control can be completed using available tasks and agents, given some collection of constraints that govern agent/task interaction and a set of worlds **W**. Arguably this is the most basic property of workflow-based access control that one would expect to be attained for a given model. For complex workflows that incorporate disjunctive or conditional branching, for the workflow to be complete all tasks identified by the workflow manager including those within all branches of the workflow must be available and must be accessible by available agents.

Definition 12

Let workflow be $W = (T, F, TR, C, TC, \mathbf{W}, \delta)$. *Workflow* W *is complete if there exists a valid assignment function* $AW : T \rightarrow A$ *in set of worlds* $\mathbf{W}$.

3.4.2.1 2-task workflow completeness example

We can establish by observation that the simple workflow is complete under the definition of section 3.4.2 above. Given the role assignment, constraint and worlds $\mathbf{W}$, agent $Agent_1$ can access task $Task_1$ and $Agent_2$ can access $Task_2$ so workflow W is complete given the set of agents and the access control policy.

3.4.3 Satisfiability of BPAC policies

Satisfiability as defined for our modelling environment is an example of the reachability property in which a workflow-based access control or some portion thereof can be terminated using available tasks and agents and given some collection of constraints that govern agent/task interaction together with a set of worlds $\mathbf{W}$. For a workflow to be terminated in this context we say that the terminating process t_e must be triggered as a consequence of agent and task assignments. It is important to note that whilst a workflow may be identified as satisfiable this does not necessarily mean that the workflow is complete as per section 3.4.2 and that a workflow that is complete may be satisfiable by a number of different workflow paths and agent and task assignments.

Definition 13

Let workflow be $W = (T, F, TR, C.TC, \mathbf{W}, \delta)$. *Workflow* W *is satisfiable if there exists a terminating instance subgraph* $W_{sub} = (T', F', TR, C, TC, \mathbf{W}, \delta)$ *in respect of workflow* W *and there exists a valid satisfiable assignment function* $SAW : T' \rightarrow A'$ *in set of worlds* $\mathbf{W}$.

3.4.3.1 2-task workflow satisfiability example

We can establish by observation that the workflow is satisfiable under the definition in section 3.4.3 above. Given the role assignment, constraint and set of worlds $\mathbf{W}$, agent $Agent_1$ can access task $Task_1$, $Agent_2$ can access $Task_2$ and $Task_2$ is a terminating task/node consistent with the workflow constraint terms of section 3.2.2 so workflow W is satisfiable given the set of agents and the access control policy.

3.4.4 The security problem - collusion

The collusion metric is the minimum of the set of all of the cardinalities of different image elements of the satisfiable assignment function for a workflow W.

The simple collusion metric represents an attempt to quantify the "robustness" of a workflow in respect of the potential for some set of agents either to collectively secure control over the entire workflow or over some satisfiable path through the workflow or to pass over control of the entire workflow or some satisfiable path to some external third party. An estimate of the simple collusion metric is obtained by identifying the minimum number of agents that are required to ensure that a workflow or some path through the workflow is satisfiable. Intuitively, a workflow is robust against collusion if, through the proper implementation of segregation of duties, a relatively large number of agents is required to secure control of the workflow or some satisfiable path. Conversely, a workflow is weak against collusion if a small number of agents or perhaps just one agent alone can secure control over the workflow.

Definition 14

Let $W = (T, F, TR, C, TC, \mathbf{W}, \delta)$ *be a workflow. Let* SAW *be the set of all valid assignment functions for the workflow then we define the collusion metric* cm_W *to be:*

$$cm_W \triangleq \underset{saw \epsilon SAW}{\mathbf{Min}} \left\{ \# \bigcup_{t \epsilon T} SAW\,(t) \right\}$$

3.4.4.1 2-task workflow collusion metric example

We observe that the only possible pattern of agent/task access is that identified in the previous test of satisfiability as agent $Agent_3$ can access neither $Task_1$ nor $Task_2$, $Agent_1$ can access task $Task_1$ but not $Task_2$ and agent $Agent_2$ can access $Task_2$ but not $Task_1$. We can conclude, therefore, that the minimum number of agents required to complete W is two, i.e. the collusion metric for the workflow is two.

3.4.5 The security problem - anonymity

We define anonymity as a property of a BPAC model whereby some outside observer is unable to match an agent to tasks and workflows within the model. If anonymity is breached, this provides a shortcut towards gaining control over a workflow in that identified agents can be targeted by another agent or third party. A breach of anonymity can occur if an agent or outside observer can observe the functioning or otherwise of a workflow given the presence or absence of a targeted agent and can potentially associate the targeted agent with tasks in consequence. For example, if a workflow becomes deadlocked when

an agent is absent then an observer can infer that there is a likelihood that the agent is critical to the workflow and she might also be able to link the agent to a specific task. Within our BPAC modelling environment we say that anonymity in respect of some targeted agent is preserved for a workflow if that workflow is complete both with and without the presence of the agent.

Definition 15

Let $W = (T, F, TR, C, TC, \mathbf{W}, \delta)$ *be a workflow and* a *be an agent. Anonymity is preserved in respect of agent* a *if for all valid assignments* $AW : T \longrightarrow A$ *there exists a valid assignment* $AW' : T \longrightarrow A$ *such that* $AW'(t) = AW(t)$ *if* $AW(t) \neq a$ *and* $AW'(t) \neq a$ *for all tasks* $t \in T$.

3.4.5.1 2-task workflow anonymity example

We observe that, given the set of agents, task $Task_1$ can only be accessed by agent $Agent_1$ and that $Agent_1$ is critical to the completion of the simple workflow W. Anonymity is not preserved in respect of agent $Agent_1$ and task $Task_1$ as $Agent_1$ is directly associated with $Task_1$ and the completion of $Task_1$ can only have occurred because of the interaction between $Agent_1$ and $Task_1$. Likewise, anonymity is not preserved in respect of agent $Agent_2$ and task $Task_2$. From the perspective of workflow W we can argue that anonymity of $Agent_3$ is preserved as $Agent_3$ cannot be associated with any aspect of the workflow W.

3.4.6 Conclusion

In this section we define BPAC and security issues associated with workflow-based access control. Throughout, we present a very simple example of 2-task workflow-based access control and we discuss the security issues within the context of the 2-task model. In the next section we present a selection of workflow-based access control examples and we discuss why we consider existing access control modelling solutions inadequate for the purpose of modelling these examples and how we can apply BPAC to these examples.

3.5 Practical examples of access control problems

In this section we present two examples of access control problems using Cassandra and RW. We highlight the limitations of each approach that we seek to address with BPAC and its modelling environment.

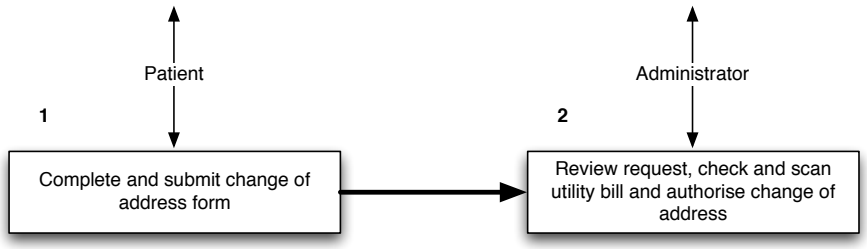


Figure 3.1: Data update workflow

3.5.1 Example 1: updating standing data in a medical practice

3.5.1.1 The problem

Medical Practice X is a small organisation operating within the National Health Service.

Consider the simple example of the requirement to update the standing data of a patient on the practice's computer-based records system, a change of address, say. A possible system might be as follows:

a patient has a very limited access to portions of her own patient records and to a selection of resources and processes. The patient fills out a change of address form on a computer terminal but on submission the address is only provisionally updated prior to verification.

The practice administrator checks the records and is required to verify the new address against some documentary evidence, a utility bill, say. A copy of the evidence is scanned and entered with the patient record update. The administrator is then able to update the patient records.

This example can be represented by the trivial two-task workflow shown in figure 3.1. The administrator need only access the patient update task contingent upon the satisfaction of the precondition, namely, the submission of the update request by the patient and conditional upon the administrator not being the patient herself. Furthermore, access to the initial task by the patient requires both the assignment of the patient role and access based upon the identity of the user. Within the context of the security model the aim of this business process is to ensure that the administrator does not have total control over valuable personal data. Most access control systems lack the combination of features that enable this security model: the dependency of access control on historical access control information and the application of global constraints based upon the user set as well as the role set. As an example, the definition of an access control policy set using Becker's Cassandra [13] is considered below. It should be noted that Cassandra is a policy language and does not incorporate a modelling and testing environment for access control policies.

3.5.1.2 An attempt to devise a policy ruleset for the problem using Cassandra

The following code snippets provide examples of Cassandra policy rules for the medical practice problem.

The policy ruleset (code snippets)

A policy ruleset for the example set out above is as follows:

$$\begin{aligned}&\mathsf{canActivate}\,(pat, \texttt{Request} - \texttt{update} - \texttt{patient} - \texttt{records}\,(pat)) \leftarrow\\ &\quad \mathsf{hasActivated}\,(pat, \texttt{Patient}\,())\,,\\ &\quad \mathsf{no} - \mathsf{main} - \mathsf{role} - \mathsf{active}\,(pat)\end{aligned}$$

That is the patient *pat* can activate the `Request` – `update` – `patient` – `records` () action provided that they have already activated the `Patient` () role and no other roles are active.

$$\begin{aligned}&\mathsf{canActivate}\,(admin, \texttt{Update} - \texttt{patient} - \texttt{records}\,(pat, admin)) \leftarrow\\ &\quad \mathsf{hasActivated}\,(admin, \texttt{Administrator}\,())\,,\\ &\quad \mathsf{canActivate}\,(pat, \texttt{Patient}\,())\,,\\ &\quad \mathsf{no} - \mathsf{main} - \mathsf{role} - \mathsf{active}\,(admin)\end{aligned}$$

states that administrator *admin* can activate the `Update` – `patient` – `records` () action provided that they have already activated the `Administrator` () role, that they have no other roles active and that the patient *pat* is capable of activating the `Patient` () role where:

$$\begin{aligned}&\mathsf{no} - \mathsf{main} - \mathsf{role} - \mathsf{active}\,(user) \leftarrow\\ &\quad \mathsf{count} - \mathsf{patient} - \mathsf{activations}\,(n, user)\,,\\ &\quad \mathsf{count} - \mathsf{admin} - \mathsf{activations}\,(n, user)\,,\\ &\quad \vdots\\ &\quad n = 0\end{aligned}$$

uses cardinality constraints to implement mutual exclusion and the cardinality constraints are defined as,

for example:

$$\begin{aligned}&\mathsf{count-patient-activations}\left(\mathsf{count}\left(u\right), user\right) \leftarrow \\ &\quad \mathsf{hasActivated}\left(user, \mathtt{Patient}\left(\right)\right) \\ &\quad \vdots\end{aligned}$$

The policy code above provides the administrator with patient record update rights to all patient records at all times provided that the administrator has activated her administrator role and has not activated any other roles. Therefore, the policy ruleset is not sufficient to provide the level of constraint required in the example i.e. the administrator is not sufficiently constrained so as to access the specific patient's details only. The scope of constraints that can be placed upon this access policy ruleset is limited although any number of constraints can be applied to the policy ruleset. Currently, the constraints are the role activation cardinality rule and a rule that states that the patient *pat* is able to activate the patient role. If we altered the rule using:

$$\begin{aligned}&\vdots \\ &\mathsf{hasActivated}\left(pat, \mathtt{Patient}\left(\right)\right), \\ &\vdots\end{aligned}$$

instead of:

$$\begin{aligned}&\vdots \\ &\mathsf{canActivate}\left(pat, \mathtt{Patient}\left(\right)\right), \\ &\vdots\end{aligned}$$

then the administrator would only be able to update the patient records if the patient had activated and not deactivated her patient role in the current session, which is clearly impractical.

The Cassandra policy language allows us to assign set membership constraints, temporal constraints and constraints based upon credentials. Whilst it is possible that a solution to this simple example

could be devised (e.g. using the patient to update some form of credential that provides a trigger to the administrator's update records policy, or by increasing the granularity of roles and actions) it is apparent that such a solution requires a counterintuitive and potentially complex interpretation of the Cassandra policy language.

3.5.1.3 Using BPAC to model the example

Now we consider the modelling of the data update example using BPAC.

We have a set of agents $A = \{Agent_1, Agent_2\}$, roles $R = \{patient, administrator\}$ and a set of worlds $\mathbf{W}$. We represent the workflow by 2 tasks $T = \{Task_1, Task_2\}$:

- $Task_1$ — complete and submit change of address form
- $Task_2$ — review request, check and scan utility bill and authorise change of address

and a single edge $F = \{(Task_1, Task_2)\}$.

Next, we populate initial store/world $\mathcal{W}_1 \in \mathbf{W}$ with patient identity constraint terms of the form $hasIdentity\,(Agent, ID_{pat})$ where ID_{pat} is a uniquely assigned identity for agent $Agent$:

$$\mathcal{W}_1 \quad = \quad \{hasIdentity\,(Agent_1, ID_{pat1})\,, hasIdentity\,(Agent_2, ID_{pat2})\}$$

We define agent/role assignment as follows:

$$AR \quad = \quad \{(Agent_1, patient)\,, (Agent_2, administrator)\}$$

and task/role assignment:

$$TR \quad = \quad \{(Task_1, patient)\,, (Task_2, administrator)\}$$

then constraints over the workflow tasks are specified as the following workflow constraint policy with a as a variable:

$$(Task_1, test\,(hasIdentity\,(a, ID_{pat1})\,, \mathcal{W}_1))\,,$$
$$(Task_2, test\,(\neg hasAccessed\,(a, Task_1, W)\,, \mathcal{W}_2))$$

The delta function for each task based upon the initial store/world $\mathcal{W}_1$ is:

$$\begin{aligned}
\delta\,(Task_1, \mathcal{W}_1) &= \{hasIdentity\,(Agent_1, ID_{pat1})\,, hasIdentity\,(Agent_2, ID_{pat2})\,, \\
&\quad hasAccessed\,(Agent_1, Task_1, W)\} \\
\delta\,(Task_2, \mathcal{W}_2) &= \{hasIdentity\,(Agent_1, ID_{pat1})\,, hasIdentity\,(Agent_2, ID_{pat2})\,, \\
&\quad \{hasAccessed\,(Agent_1, Task_1, W)\,, \\
&\quad hasAccessed\,(Agent_2, Task_2, W)\}
\end{aligned}$$

With this simple encoding we ensure that only the agent who is a *patient* and who has identity ID_{pat1} can access task $Task_1$ and can complete and submit a change of address form. Evaluation of the constraint test for task $Task_1$ against store $\mathcal{W}_1$ is successful for agent $Agent_1$ in this example. The request is reviewed under task $Task_2$ only once $Task_1$ has been completed and only by the agent who is an *administrator* and who has not completed task $Task_1$ herself. Evaluation of the constraint test for task $Task_2$ against store $\mathcal{W}_2$ is successful for agent $Agent_2$ in the example. By this method we successfully incorporate all of the security requirements specified in the outline of the problem in section 3.5.1.1 whilst avoiding potential SoD issues arising from role activation/deactivation and mutual exclusivity of roles.

3.5.1.4 Conclusion

Whilst this example would appear to be somewhat trivial, the point to note is that by applying a standard RBAC policy solution to the problem the administrator is still endowed with the freedom to view and amend any patient records without restriction. This problem arises because Cassandra uses activation and deactivation of roles, coupled with mutual exclusivity of role restrictions for SoD, and this approach lacks the granularity to apply SoD to a specific task. However, because Cassandra is a policy language only and there is no associated mechanism for verification and testing of policy languages then such limitations only become apparent by observation, or through the setting-up and testing of prototypes and real-world studies.

BPAC addresses some of the limitations identified above. Workflows can be used to temporally constrain access actions so that an administrator is only able to update patient records (via a task) if a request for such an update has been made beforehand (via a previous task). Also, dynamic segregation of duties constraints can be utilised to limit accessibility to specific patient records with reference to previous task activities e.g. an administrator can be restricted to the records identified in the patient request of the previous task.

Our BPAC modelling environment provides a mechanism for testing the security limitations of access policies, such as the simple example above, prior to live testing and implementation. This approach could potentially yield considerable resource savings and reduce the risk of security failure within an organisation. As the BPAC formalism is a workflow-based access control approach then our modelling environment is more expressive than traditional RBAC policy languages. Also, access policy creation can be addressed in a manner that is broadly isomorphic to BPM methods that are commonplace within organisations.

3.5.2 Example 2: submitting a paper to an academic publication

3.5.2.1 The problem

Journal Y is a popular academic publication produced by the Institute Z as quarterly issues of its collection of peer-reviewed papers. The business process for submission of papers for review, approval and publication is described by the workflow displayed in figs 3.2 and 3.3. Points of interest in this particular model are as follows:

- The editor cannot be the author of the paper.
- The reviewers should be different and cannot be the author of the paper. There should be a minimum of two reviewers.
- The editor can be a reviewer but in this case there needs to be at least two additional reviewers.
- A reviewer cannot access the resources (or the reviews) of her fellow reviewers.
- The editor collates the completed reviews.

An important component of this business process is that agents are unable to invoke dual roles within a session, for example author and reviewer or reviewer and reviewer. Additionally, it should not be possible for an agent to alter or manipulate records and resources that are made available as part of their task

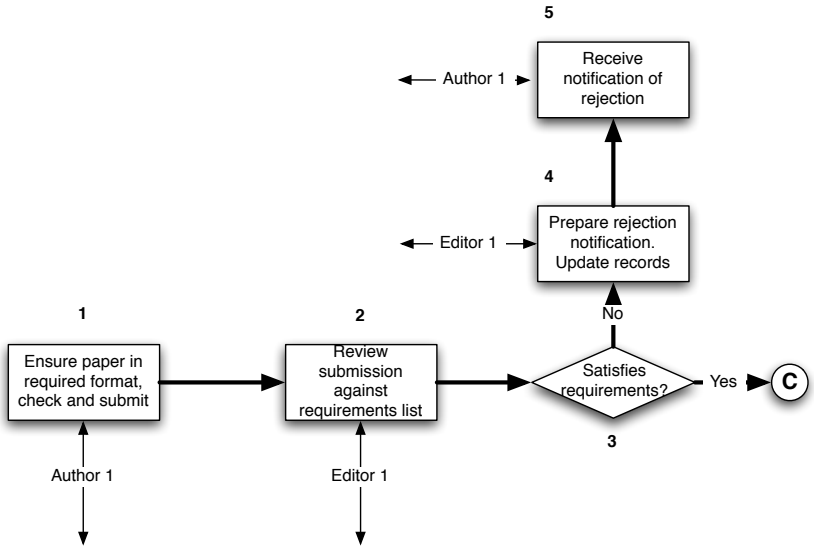


Figure 3.2: Paper submission and review workflow — part 1

execution once the paper has passed to the next task stage in the workflow. For example, once a reviewer has submitted her review then her involvement with the particular paper ceases and she cannot amend or alter records after the event and after she has discovered the names of the author or the names of the other reviewers.

3.5.2.2 An attempt to devise a policy model for the problem using RW

We consider the use of Nan Zhang's RW [104] for this particular problem and discuss a selection of the resulting policy rules.

The policy ruleset (code snippets)

Predicate terms are true ($\top$) or false ($\bot$). For instance $\texttt{author}(p, a)$ is true if a is the author of p and false otherwise. $\mathbf{w}$ is a mapping that rewrites the truth value of the predicate term to true or false. Formulas define conditions under which an agent denoted by variable x can read and overwrite the truth value of the variable represented by the instance predicate term.

A selection of policy rules using RW is set out below.

A review, written by agent a, of a paper p can be defined in respect of the writer of the review, being agent a, having the role of reviewer, who is not the paper's author and who has not submitted a review for the paper already:

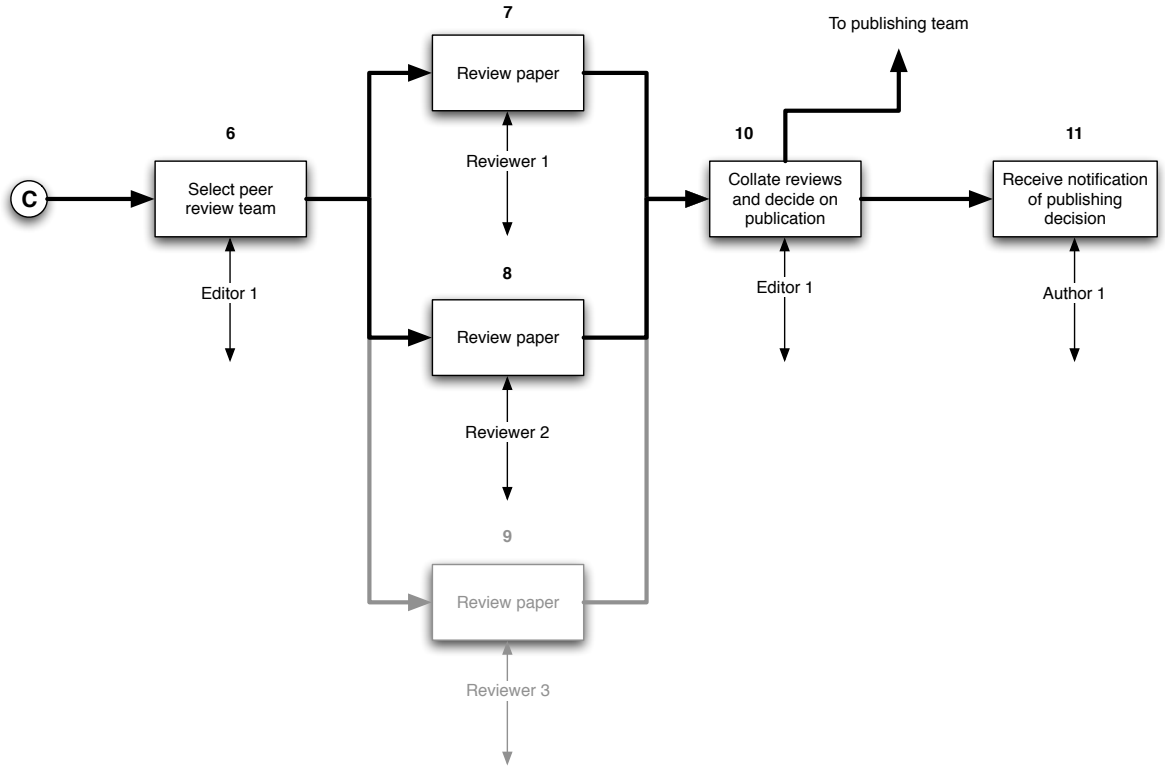


Figure 3.3: Paper submission and review workflow — part 2

$$\mathbf{w}\left(\texttt{review}\left(p,a\right),x\right) \quad \rightleftharpoons \quad \left(x = a\right) \wedge \texttt{reviewer}\left(p,x\right) \\ \wedge \neg\texttt{author}\left(p,x\right) \wedge \neg\texttt{submittedreview}\left(p,x\right)$$

and the policy language controls the access control management as well as the access control itself so that:

$$\mathbf{w}\left(\texttt{reviewer}\left(p,a\right),x\right) \quad \rightleftharpoons \quad \texttt{editor}\left(x\right) \wedge \neg\texttt{author}\left(p,a\right)$$

provides the editor with the policy to nominate and remove reviewers so long as she is the editor and the nominee reviewer is not the author of the paper. Note that the non-deterministic dynamic segregation of duties that is given in the example in respect of the editor and whether or not she can be a reviewer is merely absorbed into the decision-making process of the editor in the above policy rule. This is a more liberal and flexible interpretation of the access control than is required in the example as there is neither restriction on reviewer numbers nor prevention of the editor from being the sole reviewer.

$$\mathbf{r}\left(\texttt{submittedreview}\left(p,a\right),x\right) \quad \rightleftharpoons \quad \left(\texttt{editor}\left(x\right) \wedge \neg\texttt{author}\left(p,x\right)\right) \\ \vee\left(\left(x=a\right) \wedge \texttt{review}\left(p,x\right)\right)$$

States that either the editor, who is not the author of the paper, or the reviewer of the paper or both can read the submitted review. Whilst:

$$\mathbf{w}\left(\texttt{submittedreview}\left(p,a\right),x\right) \quad \rightleftharpoons \quad \left(x=a\right) \wedge \texttt{review}\left(p,x\right)$$

says that the agent who wrote the review can submit the review.

$$\mathbf{w}\left(\texttt{collatereviews}\left(p\right),x\right) \quad \rightleftharpoons \quad \texttt{editor}\left(x\right) \wedge \neg\texttt{author}\left(p,x\right) \\ \wedge\forall y \in \Sigma\left(\texttt{submittedreview}\left(p,y\right)\right. \\ \left.\wedge\texttt{reviewer}\left(p,y\right)\right)$$

States that the editor can collate reviews provided that all appointed reviewers have submitted the reviews.

These attempts at coding the access control demonstrate that the RW policy language is very compact and powerful despite its basic syntax. Sessions are defined via the named variables such as the paper identifier p so that for example the author argument $\texttt{author}\left(p,x\right)$ identifies the author for the given paper. Consequently, policy constraints can be granular and specific: $\texttt{author}\left(p,x\right)$, or general: $\texttt{author}\left(y,x\right)$, where y is a member of the set of submitted papers. However, there is a problem with the language in this particular example. The example coding above provides the editor with considerable power over the review and selection process. If we restrict the paper review process to exclude the editor, a static segregation of duties can be invoked through mutual exclusion. However, the scenario allows for the editor being a reviewer so this is not appropriate. Alternatively, the `reviewer()` predicate can be extended to provide one version for ordinary reviewers and another for the editor reviewer, `editorreviewer()`. Now, the policy rule for writing a review might look like the following:

$$\begin{aligned}\mathbf{w}\left(\texttt{review}\left(p,a\right),x\right) \quad \rightleftharpoons \quad & \left(x=a\right)\wedge\neg\texttt{submittedreview}\left(p,x\right)\\ & \wedge\left(\left(\texttt{editorreviewer}\left(p,b\right)\wedge\left(x=b\right)\right)\right.\\ & \vee\left(\left(\texttt{reviewer}\left(p,c\right)\wedge\left(x=c\right)\right.\right.\\ & \left.\left.\wedge\neg\exists y\in\Sigma\texttt{review}\left(p,y\right)\right)\right.\\ & \vee\left(\texttt{reviewer}\left(p,d\right)\wedge\left(x=d\right)\right.\\ & \left.\left.\wedge\exists z\in\Sigma\left(\texttt{review}\left(p,z\right)\wedge\left(z\neq x\right)\right)\right)\wedge\neg\texttt{editor}\left(x\right)\right)\\ & \wedge\neg\texttt{author}\left(p,x\right)\end{aligned}$$

This rule says that agent x can write a review if she has not already submitted a review and she is the editor-reviewer or she is a reviewer and not the editor and no one is writing a review or there are others writing reviews but not her. Whilst this rule should provide the required constraint it is rather more complex than previous rules and it is not intuitively associated with the underlying central business process.

What happens if we tweak the initial example requirements slightly? For example, the editor can collate submitted reviews provided that she has not submitted a review herself, otherwise another editor is required to collate the reviews (a dynamic SoD example). The policy rule for the collation of reviews can be amended to incorporate the additional constraint:

$$\begin{aligned}\mathbf{w}\left(\texttt{collatereviews}\left(p\right),x\right) \quad \rightleftharpoons \quad & \left(\left(\texttt{editor}\left(x\right)\wedge\neg\texttt{editorreviewer}\left(p,x\right)\right)\right.\\ & \vee\left(\texttt{editor}\left(x\right)\right.\\ & \wedge\exists y\in\Sigma\left(\texttt{editorreviewer}\left(p,x\right)\wedge\left(y\neq x\right)\right.\right.\\ & \left.\left.\left.\wedge\left(y\neq x\right)\right)\right)\right)\\ & \wedge\neg\texttt{author}\left(p,x\right)\\ & \wedge\forall z\in\Sigma\left(\texttt{submittedreview}\left(p,z\right)\right.\\ & \left.\wedge\texttt{reviewer}\left(p,z\right)\right)\end{aligned}$$

that says the editor can collate reviews provided that she is not an editor-reviewer or there is an editor-reviewer and she is not the editor-reviewer.

Ostensibly, we now have a set of rules that provide the access policy requirement. However, there is a

difficulty in that the original editor can potentially circumvent this additional restriction. Firstly, she can write a review, then secondly, she can legitimately resign her role as editor-reviewer, averting the collation constraint in consequence. This is the fundamental problem with utilising mutual exclusion to enact dynamic SoD and is why RBACs are not suited to implementing this business principle.

Unlike Cassandra discussed in the previous example RW incorporates a powerful modelling and testing environment that is able to identify security problems in access control policies. However, whilst this testing environment can identify flaws such as contradictions between policy rulesets, it is not so able to highlight failure or limitations of the application of business rules such as dynamic segregation of duties.

3.5.2.3 Using BPAC to model the example (code snippets)

We consider the modelling of the review section of this example using BPAC for this exercise.

We have a set of agents $A = \{Agent_1, Agent_2, Agent_3, \ldots\}$, roles $R = \{PCMember, editor, \ldots\}$ and a set of worlds $\mathbf{W}$. We represent the workflow by tasks $T = \{Task_1, \ldots, Task_{11}\}$:

- $Task_1$ — ensure paper in proper format , check and submit

  ⋮

- $Task_6$ — select peer review team
- $Task_7$ — review paper
- $Task_8$ — review paper
- $Task_9$ — review paper
- $Task_{10}$ — Collate reviews and decide on publication

  ⋮

We extend the set of task/nodes to include dummy (non-interactive) nodes to ensure that the workflow form is consistent with the workflow definitions and constraints in sections 3.2 and 3.3:

$T_{mod} = T \cup \{T_{node1}, T_{node2}, T_{node3}, T_{node4}, \ldots\}$

where $T_{node1}, \ldots, T_{node4}$ are dummy nodes representing splits and joins in the workflow and T_{mod} is the new task set for the model. We modify the workflow model so that it properly reflects the possibilities for paper review being performed either by two reviewers nominated by the editor or by the editor and two reviewers as shown in figure 3.4. Disjunctive splits and joins are identified in figure 3.4 by XOR labels and conjunctive splits and joins are unlabelled.

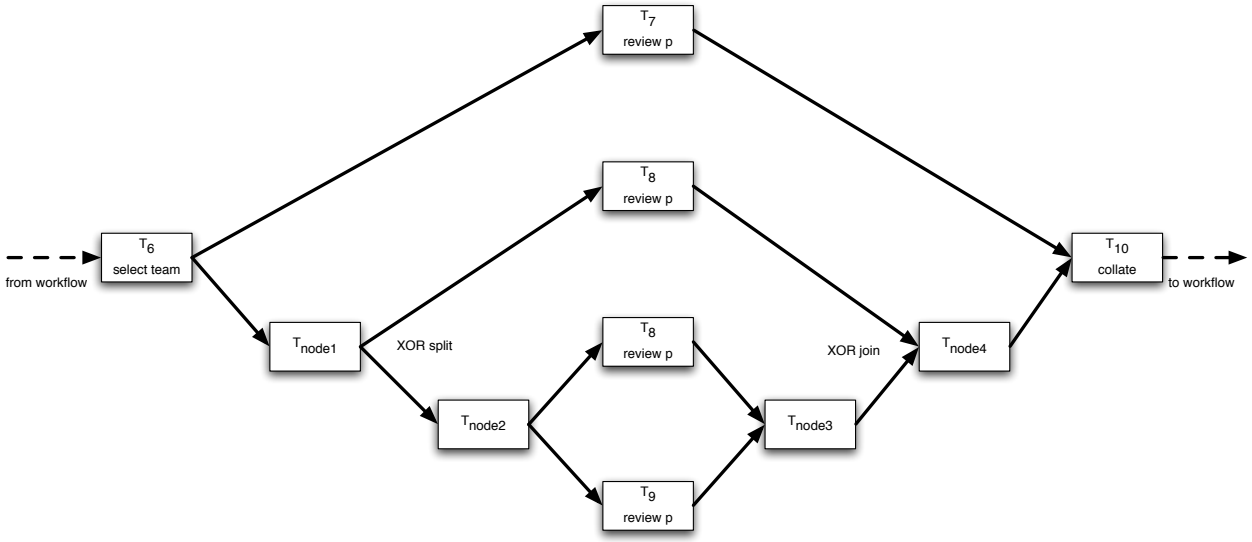


Figure 3.4: Paper submission and review BPAC segment

We define agent/role assignment as follows:

$$\begin{aligned} AR \quad = \quad & \{(Agent_1, member), (Agent_1, editor), (Agent_2, member), \\ & (Agent_2, reviewer), (Agent_3, member), (Agent_3, reviewer), \ldots\} \end{aligned}$$

and task/role assignment:

$$\begin{aligned} TR \quad = \quad & \{(Task_1, PCmember), (Task_6, editor), (Task_7, reviewer), \\ & (Task_8, reviewer), (Task_9, editor), \ldots\} \end{aligned}$$

then tests of constraints over workflow tasks are specified in respect of n worlds $\mathbf{W} = \{\mathcal{W}_1, \ldots \mathcal{W}_n\}$ as the following workflow constraint policy (extracts):

$$\ldots, (Task_6, test(\neg hasAccessed(a, Task_1, W), \mathcal{W}_6)),$$
$$(Task_7, test(\neg hasAccessed(a, Task_1, W)$$
$$\wedge \neg hasAccessed(a, Task_6, W)$$
$$\wedge \neg hasAccessed(a, Task_8, W)$$
$$\wedge \neg hasAccessed(a, Task_9, W), \mathcal{W}_7)),$$
$$(Task_8, test(\neg hasAccessed(a, Task_1, W)$$
$$\wedge \neg hasAccessed(a, Task_6, W)$$
$$\wedge \neg hasAccessed(a, Task_7, W)$$
$$\wedge \neg hasAccessed(a, Task_9, W), \mathcal{W}_8)),$$
$$(Task_9, test(\neg hasAccessed(a, Task_1, W)$$
$$\wedge \neg hasAccessed(a, Task_7, W)$$
$$\wedge \neg hasAccessed(a, Task_8, W), \mathcal{W}_9)), \ldots$$

With this encoding we ensure that only the editor accesses task $Task_6$ so long as she has not accessed task $Task_1$ and is not therefore the author of the paper. Subsequently, review tasks are either performed by two reviewers, neither of whom is the editor nor the author, or the editor plus two reviewers. None of the reviewers has already reviewed or is in the process of reviewing the paper. This encoding is consistent with the key points identified in section 3.5.2.1. It demonstrates that a workflow-based access control approach such as BPAC constrains access control through the application of traditional RBAC via task-specific, dynamic constraints and by the progress from one task to another within the workflow itself.

3.5.2.4 Conclusion

Enforcement of least privilege and segregation of duties within RW requires the addition of increasing numbers of constraint terms as the policy requirements become more complex. RW uses statically mutually exclusive roles to invoke SoD and as is pointed out in section 2.4.4, verification of enforceability of static SoD by SMERs is intractable.

RW's policy approach mixes roles, activities and role management. Consequently, setting up and maintaining constraints that extend across activities or tasks within the role or activity context and without

session histories is counterintuitive. Constraints are more naturally and economically applied by considering them within the centralised business process context.

Mutually exclusive roles are not, by themselves, sufficient to enforce dynamic segregation of duties. The example above demonstrates that, without session histories, mutual exclusion can be circumvented by establishing separate sessions through, for example, resigning roles.

The BPAC modelling environment addresses these limitations by shifting focus to centralised business processes. Applying constraints to linked tasks simplifies the constraint management process. Session histories maintained across business processes facilitates full dynamic SoD.

BPAC would be of limited usefulness if it was not able to correctly reveal security flaws that can also be identified by existing access control modelling environments, such as RW. In chapter 8 we demonstrate that the applied pi calculus implementation of BPAC can identify a security flaw in a simple conference management system, as can RW and then we proceed to demonstrate security tests that are not possible with RW such as the quantification of agent collusion and anonymity.

3.5.3 Practical examples — overall conclusions

A key problem with the application of RW, Cassandra and RBACs in general within complex workflows is that the policy languages are agent-centric or role-centric i.e. the policy language and constraints focus upon the definition of rules for agents or roles. Consequently, the implementation of least privilege and segregation of duties constraints becomes more complex as the number of discrete tasks increases. In effect, for each additional task, an additional constraint rule is added to the policy rules for each agent/role. Also, design and management of the agent-centric or role-centric access control model is counterintuitive. When managers consider countermeasures to security threats to their organisation they do so from the organisation's perspective and not from the employee's or agent's perspective. Access controls from an organisational perspective are concerned primarily with rules to ascertain whether or not processes or resources can be accessed by agents, that is, the rules attached to these processes or resources define some set of properties that must be associated with an agent if she is to be permitted access. Such rules can be attached directly to an organisation's documented structures such as its systems of financial control as exemplified by our BPAC modelling environment. Access control rules as exemplified by Cassandra and RW specify, on the other hand, permissions assigned to agents in respect of their capacity to access resources or to perform certain actions.

Without preconditions users may be able to exercise excessive levels of control over data and resources as we observe in the example in section 3.5.1 above. Our thesis proposal provides for preconditions through

simple workflow constructs and session histories.

Additionally, we observe through the examples above that dynamic segregation of duties, whilst a significant requirement within business processes, is not always properly enacted with role-based access control systems and the use of mutual exclusion of roles coupled with activation and deactivation of roles. Our implementation of BPAC circumvents this deficiency by utilising session histories across business processes.

Overall, therefore, our BPAC environment represents a paradigm shift towards a process-centric approach to access control modelling so that constraints are applicable across the process space. This alternative view simplifies the constraints process and attaches it to its natural environment: constraints are implementations of business rules and business rules are applied to the business processes of an organisation. Full, dynamic segregation of duties is enacted across business processes through session histories and strict least privilege is exercised through preconditions as workflows.

However, this is only part of the solution as the system also needs to incorporate mechanisms for formally verifying that the application of global constraints properly achieves the required security objectives.

3.6 Summary

In this chapter we introduce our understanding of BPAC and of the security issues that we associate with access control. We present a trivial 2-task workflow example to demonstrate how a workflow-based access control model works and we discuss the security issues: satisfiability, collusion and anonymity in respect of this trivial model. Subsequently, we present two practical examples of workflow-based access control and discuss why current policy languages or modelling environments are not sufficient for the modelling and analysis of these examples.

In the next chapter we present a simple practical workflow example in respect of a purchase order. We revisit this workflow example in subsequent chapters as we develop our implementation of BPAC and present the tools for security analysis.

Chapter 4

A simple workflow

4.1 Introduction

In a previous chapter we introduce the concept of workflow-based access control within the context of business process management, which we call BPAC. In this chapter we present a simple example of a workflow-based access control problem in BPAC, which we subsequently model in chapter 7 using the applied pi calculus modelling environment of chapter 6.

4.2 A simple purchase order workflow

Financial transactions within organisations tend to follow sequences of predetermined steps. Different agents of the organisation are required to perform processes that in total result in the completion of the transaction under consideration. These sequences of steps can be visualised as workflows. The assignment of agents to their respective processes is usually based upon a variety of properties associated with the agent such as expertise, location, availability and authority within the organisation. It is generally in the interest of the organisation that the most appropriate agents are assigned to the various processes. This ensures that the transaction is performed to a high standard, in as timely and cost-effective means as possible. As discussed above, role-based access control provides authorisation that matches appropriate agents to processes and within BPAC we assume that specific resources are assigned to each process.

In a previous chapter, we discuss segregation (or separation) of duties (SoD) as a mechanism for the prevention of fraudulent behaviour in respect of transactions within organisations. Except in the simplest of circumstances, role-based access control is not by itself best suited to the enforcement of SoD. If,

however, we combine the workflow properties of a transaction with role-based access control, we obtain an authorisation mechanism that properly addresses SoD.

Consider the simple access control workflow representing a purchase order within some organisation as set out in figure 4.1 and outlined in respect of task processes below. We assume that resources have been assigned to each task so that they can be completed satisfactorily by agents. The workflow W comprises a set of tasks $T = \{Task_1, \ldots, Task_6\}$ together with a precedence relation as follows:

$$\begin{aligned} F \quad = \quad & \{(Task_1, Task_2), (Task_2, Task_3), (Task_2, Task_4), (Task_3, Task_5), \\ & (Task_4, Task_5), (Task_5, Task_6)\} \end{aligned}$$

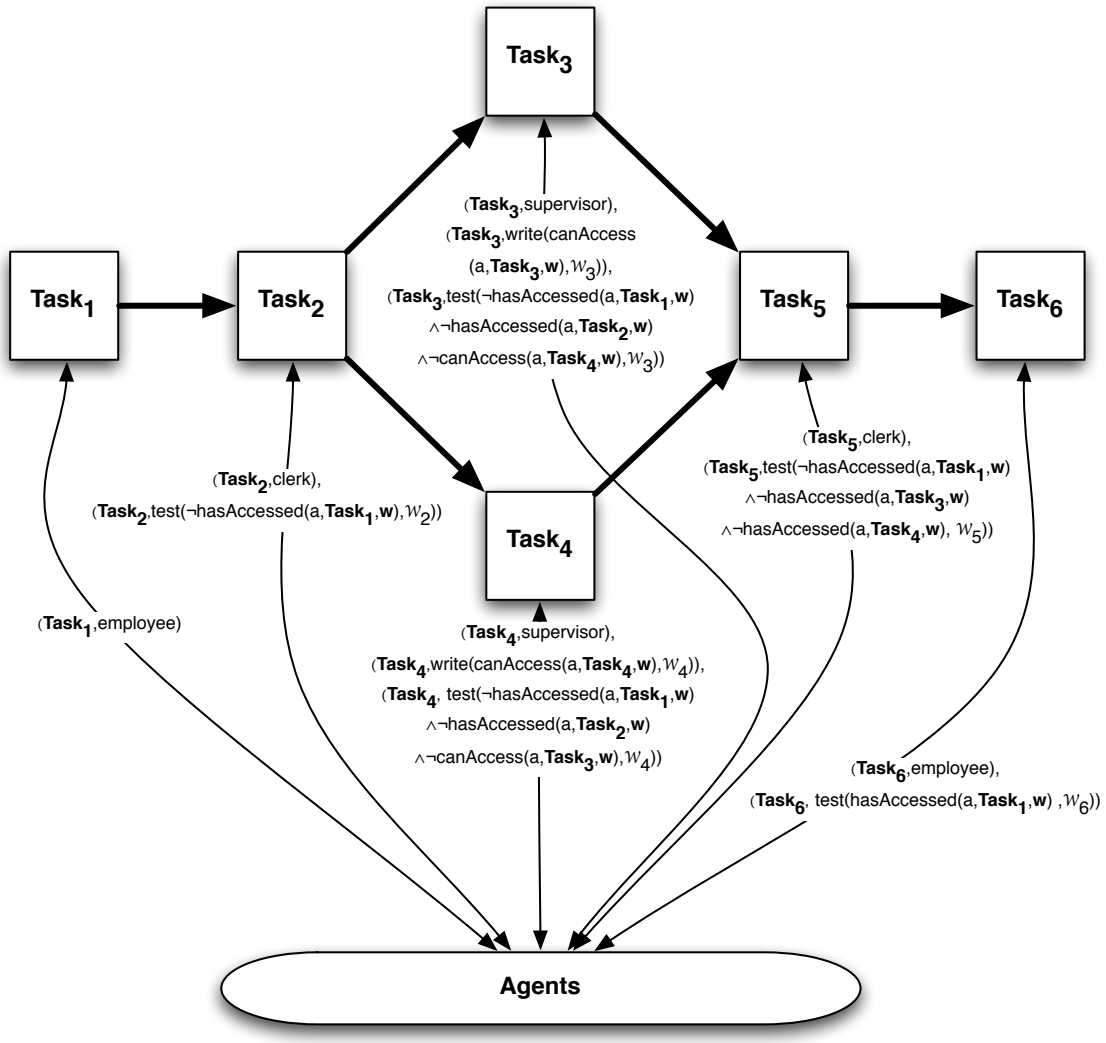
with tasks:

- $Task_1$ — an employee submits a request for a low-value item to the purchases department.
- $Task_2$ — A purchase clerk, who has not submitted the request, processes the request, checks it against available resources and prepares the purchase order.
- $Task_3$ and $Task_4$ — The purchase order is reviewed and approved by two different supervisors, neither of whom submitted or processed the request and is returned to a purchases clerk.
- $Task_5$ — The purchases clerk, who has not submitted the request and has not reviewed and approved the order, processes the order only once it has been reviewed and approved by both supervisors.
- $Task_6$ — The employee who originated the request receives the order item and confirms that the correct goods have been received in good condition.

Resources are assigned to each task to ensure their satisfactory completion and access to resources is mediated by access control over the tasks.

Within this example access control is defined by role assignment and by dynamic constraints based upon process history.

The workflow represents a sequence of connected tasks each requiring access by a single agent to perform the duties associated with these tasks. Agents need different levels of organisational seniority to perform the various tasks (agents are assigned to roles) and constraints are required to ensure that a minimum



Workflow details

Task$_1$ place order
Task$_2$ check order to resources
Task$_3$ review & approve payment
Task$_4$ review & approve payment
Task$_5$ process order
Task$_6$ confirm goods received

KEY

Task$_1$ ... **Task$_6$**	task identifiers
w	workflow identifier
Agents	set of agents
$w_2 \ldots w_6$	store/worlds

Figure 4.1: A simple purchase workflow

number of agents of the appropriate level of seniority are needed to complete the entire workflow for segregation of duties.

We define the following set of roles:

$$R = \{employee,\ clerk,\ supervisor\}$$

and set of agents:

$$A = \{Agent_1,\ Agent_2,\ Agent_3,\ Agent_4,\ Agent_5\}$$

and the binary relation of agent/role assignments is given as follows:

$$\begin{aligned} AR = \{&(Agent_1, employee),\\ &(Agent_2, employee),\ (Agent_2, clerk),\\ &(Agent_3, employee),\ (Agent_3, clerk),\\ &(Agent_4, employee),\ (Agent_4, clerk),\\ &(Agent_4, supervisor),\\ &(Agent_5, employee),\ (Agent_5, supervisor)\} \end{aligned}$$

As in the previous example of chapter 3 a basic RBAC constraint for this example is provided by the following task/role mapping:

$$\begin{aligned} TR = \{&(Task_1, employee),\ (Task_2, clerk),\\ &(Task_3, supervisor),\ (Task_4, supervisor),\\ &(Task_5, clerk),\ (Task_6, employee)\} \end{aligned}$$

So that $(Task_1, employee)$ models that the agent mapped to task $Task_1$ must have role *employee*, $(Task_2, clerk)$ the agent mapped to $Task_2$ must have role *clerk* etc.

Over this RBAC-based access control model we consider dynamic segregation of duties constraints as policy rules, in respect of a store represented by a set of worlds $\mathbf{W} = \{\mathcal{W}_1, \ldots, \mathcal{W}_6\}$, as follows:

$$
\begin{aligned}
&(Task_2, test\,(\neg hasAccessed\,(a, Task_1, W)\,, \mathcal{W}_2))\,,\\
&(Task_3, write\,(canAccess\,(a, Task_3, W)\,, \mathcal{W}_3))\,,\\
&(Task_3, test\,(\neg hasAccessed\,(a, Task_1, W)\\
&\wedge \neg hasAccessed\,(a, Task_2, W)\\
&\wedge \neg canAccess\,(a, Task_4, W)\,, \mathcal{W}_3))\,,\\
&(Task_4, write\,(canAccess\,(a, Task_4, W)\,, \mathcal{W}_4))\,,\\
&(Task_4, test\,(\neg hasAccessed\,(a, Task_1, W)\\
&\wedge \neg hasAccessed\,(a, Task_2, W)\\
&\wedge \neg canAccess\,(a, Task_3, W)\,, \mathcal{W}_4))\,,\\
&(Task_5, test\,(\neg hasAccessed\,(a, Task_1, W)\\
&\wedge \neg hasAccessed\,(a, Task_3, W)\\
&\wedge \neg hasAccessed\,(a, Task_4, W)\,, \mathcal{W}_5))\,,\\
&(Task_6, test\,(hasAccessed\,(a, Task_1, W)\,, \mathcal{W}_6))
\end{aligned}
$$

i.e. the agent mapped to task $Task_2$ cannot have been mapped to task $Task_1$ etc. By this method we avoid the complexity that arises with SMER matrices. However, we rely upon stateful resolution of our constraints when we test our access control model against the set of worlds $\mathbf{W}$. We note the use of a 'write' before 'test' as outlined in section 3.3.11 for parallel tasks $Task_3$ and $Task_4$, with the appropriate constraint tests to ensure that an agent accessing task $Task_3$ cannot concurrently access task $Task_4$and vice versa.

4.3 Summary

In this chapter we present a simple workflow-based access control model, which is representative of a typical financial transaction within an organisation, namely a low value purchase order. Following a narrative discussion of the financial transaction and its access control requirements, we then present the transaction in detail as a workflow-based access control process.

4.3 Summary

In the next chapter we detail the syntax and semantics of the applied pi calculus that forms the basis of our access control model. In chapter 6 we develop a modelling environment for BPAC based upon the applied pi calculus of chapter 5. In chapter 7 we apply the modelling environment of chapter 6 to the simple workflow example outlined in this chapter.

Chapter 5

Applied pi calculus for BPAC

5.1 Introduction

In the previous chapter we present a simple practical example of a workflow-based access control process in BPAC as motivation for the development of our modelling environment. In this chapter, we turn our attention to the basis for our modelling environment, namely, the applied pi calculus of Abadi and Fournet [3]. We summarise the syntax and semantics below and we extend the calculus with syntactic sugar so that our modelling environment is more concise and easier to read. Syntactic sugaring is a common technique for hiding the details of reusable blocks of process algebra code so as to aid human understanding and to simplify modelling and analysis with the code. Writing complex code can be error prone and sugaring can mitigate against these errors. The technique is exemplified by the definition of Casper [80] as a high level derivative of CSP that aids comprehension of protocol models and yet facilitates analysis of protocol examples.

5.2 Applied pi calculus

Abadi and Fournet's applied pi calculus is a powerful, more comprehensible extension of Milner's pi calculus. It adds functions, an equational theory and separate identification of names and variables to pi calculus whilst maintaining the underlying theory. The applied pi calculus has often been applied to analysis of security protocols, particularly in respect of authentication such as Abadi et al's modelling of Just-fast keying [4] and Ryan et al's analysis of anonymity [9] and privacy within electronic voting protocols [57, 35]. Its expressive syntax and ability to handle concurrent processes, coupled with powerful

L, M, N, T, U, V	::=	
	$a, b, c, \ldots, k, \ldots m, n, \ldots, s$	name
	x, y, z	variable
	$f(M_1, \ldots, M_l)$	function

$\widetilde{a}$ and $\widetilde{x}$ represent tuples over names and variables respectively.

Table 5.1: Terms

$pk(y)$	public key of y where y is a key seed
$sk(y)$	secret key of y where y is a key seed
$key(y)$	symmetric key of y where y is a key seed
$enc(x, y)$	encryption of x with key y
$dec(x, y)$	decryption of x with key y
$encs(x, y)$	signature encryption of x with key y
$decs(x, y)$	signature decryption of x with key y
ok	constant symbol
$true$	true symbol
$false$	false symbol
hit	hit symbol
$miss$	miss symbol
$\varnothing$	empty set
$fst((x, y))$	first element of a pair
$snd((x, y))$	second element of a pair
$fst((x, y, z))$	first element of a triple
$snd((x, y, z))$	second element of a triple
$thd((x, y, z))$	third element of a triple
$fst((x, y, z, w))$	first element of a 4-tuple
$snd((x, y, z, w))$	second element of a 4-tuple
$thd((x, y, z, w))$	third element of a 4-tuple
$fth((x, y, z, w))$	fourth element of a 4-tuple

Table 5.2: Function terms

analytical tools such as observational equivalence, make the applied pi calculus a suitable environment for BPAC.

5.3 Syntax and semantics

We define a signature Σ as comprising a finite set of function symbols together with an equational theory. Terms are defined in table 5.1 as per the applied pi calculus [3] and comprise infinite sets of variables and names together with function symbols f with arity l that range over Σ.

Function terms that are used within the modelling environment include terms for encryption and signature checking together with pair and tuple manipulation terms as outlined in table 5.2. Additionally, we use data function terms of arity n as per ProVerif to represent predicates [22]. These data functions are essentially named n-tuples that can be constructed and deconstructed by an attacker.

The equational theory consists of a finite set of equational axioms. It includes typical cryptographic equations for symmetric and asymmetric encryption together with tuple destructors as outlined in table 5.3.

$dec(enc(x, pk(y)), sk(y)) = x$	asymmetric encryption
$decs(encs(x, sk(y)), pk(y)) = x$	signature checking
$fst((x, y)) = x$	pair destructor — first element
$snd((x, y)) = y$	pair destructor — second element
$fst((x, y, z)) = x$	triple destructor — first element
$snd((x, y, z)) = y$	triple destructor — second element
$thd((x, y, z)) = z$	triple destructor — third element
$fst((x, y, z, w)) = x$	4-tuple destructor — first element
$snd((x, y, z, w)) = y$	4-tuple destructor — second element
$thd((x, y, z, w)) = z$	4-tuple destructor — third element
$fth((x, y, z, w)) = w$	4-tuple destructor — fourth element

Table 5.3: Equational axioms

P, Q, R	::=	
	$\mathbf{0}$	inaction
	$P \mid Q$	parallel composition
	$!P$	replication
	$\nu n.P$	name restriction
	$\overline{u}\langle N\rangle.P$	output
	$u(x).P$	input
	if $M = N$ **then** P **else** Q	conditional

Table 5.4: Plain processes

Plain processes (table 5.4) of the applied pi calculus are adopted for the modelling environment and are defined in the same manner as per Abadi and Fournet [3]. Specifically, the null process $\mathbf{0}$ does nothing, $P \mid Q$ represents the parallel running of two processes P and Q, $!P$ is multiple parallel processes P, $\nu n.P$ creates a new private name n and then runs as P, $\overline{u}\langle N\rangle.P$ sends message N on channel u then behaves as process P, $u(x).P$ receives a message on channel u then replaces variable x with the message in P and **if** $M = N$ **then** P **else** Q represents a typical conditional that branches on the equality or otherwise of M and N, proceeding as P or Q as appropriate.

Extended processes and their semantics are adopted in their entirety from Abadi and Fournet [3] without amendment as per table 5.5. The active substitution $\{^{M}/_{x}\}$ replaces all instances of variable x that it touches with M and usually arises as the consequence of the output of M to the environment.

When discussing the scope of names and variables we often refer to sets of free or bound names and variables. Binding of scope occurs through the application of the restriction νn or νx or input $a(x)$. The sets of bound names and variables in A are written $bn(A)$ (resp. $bv(A)$) and the sets of free names and variables in A are written $fn(A)$ (resp. $fv(A)$). An extended process is closed if all process variables are bound or defined by an active substitution. A frame (usually referred to as φ or ψ [3]) is an extended process constructed from $\mathbf{0}$ and active substitutions $\{^{M}/_{x}\}$.

A, B, C	::=	
	P	plain process
	$A \mid B$	parallel composition
	$\nu x.A$	variable restriction
	$\nu n.A$	name restriction
	$\{^{M}/_{x}\}$	active substitution

Table 5.5: Extended processes

5.3.1 Syntactic sugar

We define the following syntactic sugar to simplify the modelling process.

5.3.1.1 Internal process

We adopt τ to represent an undefined internal process. The exact nature of this internal process is not of importance to us as it is merely used as a sequential placeholder, particularly when a conditional statement is placed at the end of a process.

5.3.1.2 Trigger process

It is sometimes necessary to provide communication in which the actual message is not relevant to a process as a whole. For example, two sequential task processes, T_1 and T_2, say, that comprise a simple sequential workflow can be implemented as two parallel processes linked by a communication:

$$\nu t, a.\left(T_1.\bar{t}\left\langle a\right\rangle \mid t\left(x\right).T_2\right)$$

The new name a is arbitrary and does not occur in either T_1 or T_2. Consequently, we represent such communication as per Puhlmann and Weske [77]:

$$\nu t.\left(T_1.\bar{t} \mid t.T_2\right)$$

wherein $\bar{t}$ and t are private triggers and this representation is sufficient to enable us to model various workflow structures as outlined by Puhlmann and Weske [77].

5.3.1.3 Summation process

Modelling workflows may sometimes require non-deterministic branching to emulate decision processes. In traditional pi calculus this is represented by the summation operator +:

$$(\nu v.\overline{p}\langle v\rangle.v(x).P) + \overline{q}\langle w\rangle.Q$$

This process proceeds as follows: a fresh name v is declared and sent over channel p then v becomes a channel over which some message is received that substitutes for variable x in process P and the process $\overline{q}\langle w\rangle.Q$ is lost. Alternatively, name w is sent over channel q then the process proceeds as Q and process $\nu v.\overline{p}\langle v\rangle.v(x).P$ is lost.

We can replicate the summation operator in the above example in the applied pi calculus as follows:

$$\nu int.\left(\overline{int} \mid int.(\nu v.\overline{p}\langle v\rangle.v(x).P) \mid int.(\overline{q}\langle w\rangle.Q)\right)$$

Where int is a freshly declared trigger. Either $int.(\nu v.\overline{p}\langle v\rangle.v(x).P)$ or $int.(\overline{q}\langle w\rangle.Q)$ proceed depending upon which int is triggered.

In general, therefore, we use + as defined as follows:

$$P + Q \quad \triangleq \quad \nu int.\left(\overline{int} \mid int.P \mid int.Q\right)$$

where P and Q are normal processes.

5.3.1.4 Restricted active substitution

The restricted active substitution $\upsilon x.(\{^M/_x\} \mid P)$ is represented by **let** $x = M$ **in** P, which is more intuitive.

1. *Abelian monoid laws*
   $A \mid \mathbf{0} \equiv A$
   $A \mid B \equiv B \mid A$
   $A \mid (B \mid C) \equiv (A \mid B) \mid C$

2. *replication law*
   $!P \equiv !P \mid P$

3. *scope laws*
   $\nu u.\mathbf{0} \equiv \mathbf{0}$
   $\nu u.\nu v.A \equiv \nu v.\nu u.A$
   $\nu u.\,(A \mid B) \equiv A \mid \nu u.B$
   where $u \notin fv\,(A) \cup fn\,(A)$

4. *substitution laws*
   $\nu x.\,\{^{M}/_{x}\} \equiv \mathbf{0}$
   $\{^{M}/_{x}\} \mid A \equiv \{^{M}/_{x}\} \mid A\,\{^{M}/_{x}\}$
   $\{^{M}/_{x}\} \equiv \{^{N}/_{x}\}$
   when $\Sigma \vdash M = N$

Table 5.6: Structural congruences

5.3.1.5 Product operator

We use a product operator to represent multiple parallel processes of similar syntactic form so that:

$$\prod_{n} P_n \quad \triangleq \quad P_1 \mid P_2 \mid \ldots \mid P_n$$

5.3.2 Operational semantics

We adopt the structural equivalence forms of the applied pi calculus where the structural equivalence $\equiv$, is the least equivalence on extended processes that is closed under α conversion on variables and names, by application of evaluation contexts and is given by the definitions in table 5.6 [3]. Contexts and evaluation contexts are defined in section 5.3.4.

Furthermore, we adopt the internal reduction forms of the applied pi calculus where internal reduction $\rightarrow$ is the least relation on extended processes closed by structural congruence such that the expressions of table 5.7 hold.

5.3.3 Labelled operational semantics

The semantics of section 5.3.2 are extended to incorporate interactions between extended processes and their environment via an input label $a\,(M)$ or output labels of the form $\overline{a}\,\langle u \rangle$ or $(\nu u)\,\overline{a}\,\langle u \rangle$, where M is

communication
$\overline{a}\langle x\rangle.P \mid a(x).Q \to P \mid Q$

conditional
if $M = N$ **then** P **else** $Q \to P$
for ground terms M and N s.t. $\Sigma \vdash M = N$
if $M = N$ **then** P **else** $Q \to Q$
for ground terms M and N s.t. $\Sigma \nvdash M = N$

Table 5.7: Internal reduction

out-atom
$\overline{a}\langle u\rangle.P \xrightarrow{\overline{a}\langle u\rangle} P$

in
$a(x).P \xrightarrow{a(M)} P\{^{M}/_{x}\}$

open-atom
$$\frac{A \xrightarrow{\overline{a}\langle u\rangle} A' \quad u \neq a}{\nu u.A \xrightarrow{\nu u.\overline{a}\langle u\rangle} A'}$$

scope
$$\frac{A \xrightarrow{\alpha} A' \quad u \text{ does not occur in } \alpha}{\nu u.A \xrightarrow{\alpha} \nu u.A'}$$

par
$$\frac{A \xrightarrow{\alpha} A' \quad bv(\alpha) \cap fv(B) = bn(\alpha) \cap fn(B) = \varnothing}{A \mid B \xrightarrow{\alpha} A' \mid B}$$

struct
$$\frac{A \equiv B \quad B \xrightarrow{\alpha} B' \quad B' \equiv A'}{A \xrightarrow{\alpha} A'}$$

Table 5.8: Labelled operational semantics

a term containing names or variables and u is a channel name or name of base type as per Abadi and Fournet [3]. The labelled semantics rules are defined as per table 5.8.

5.3.4 Observational equivalence

We adopt the full definition of observational equivalence as per Abadi and Fournet [3]. Intuitively, observational equivalence is when two processes cannot be differentiated by an outside observer regardless of the context that the observer may apply to them. This is a powerful analytical tool for establishing secrecy properties of processes. We define the barb $A \Downarrow a$ wherein A is permitted to send a message on a. We define a context $C[\_]$ in the usual way as an expression with a hole. If $C[A]$ is closed then $C[\_]$

closes A. An evaluation context is a context with a hole that is not under a conditional, input, output or replication.

Definition 16

Observational equivalence ($\approx$) is the largest symmetric relation $\mathcal{R}$ between signature closed extended processes with the same domain such that $A\,\mathcal{R}\,B$ implies:

1. $A \Downarrow a$ *then* $B \Downarrow a$,
2. *if* $A \longrightarrow^* A'$ *then* $B \longrightarrow^* B'$ *and* $A'\,\mathcal{R}\,B'$ *for some* B' *and*
3. $C[A]\;\mathcal{R}\;C[B]$ *for closing evaluation contexts* $C[\_]$.

Next we adopt the rules for static equivalence and labelled bisimilarity in respect of the labelled operational semantics defined in section 5.3.3 above.

5.3.5 Static equivalence

Static equivalence is defined as per Abadi and Fournet [3]. Static equivalence is, in effect, the static part of observational equivalence and is therefore concerned with equivalence between processes (or more specifically their frames) up to their current state and is not concerned with the future transitions that the processes may subsequently undergo.

Definition 17

Two terms M and N are equal in the frame φ iff $\varphi \equiv \nu\tilde{n}.\sigma$, $M\sigma = N\sigma$ and $\{\tilde{n}\} \cap (fn(M) \cup fn(N)) = \varnothing$ for some names $\tilde{n}$ and substitution σ modulo the equational theory.

Definition 18

Two closed frames φ and ψ are statically equivalent, written $\varphi \approx_s \psi$, when $dom(\varphi) = dom(\psi)$ and when, for all terms M and N, we have $M\varphi = N\varphi$ iff $M\psi = N\psi$.

Two closed extended processes A and B are statically equivalent, written $A \approx_s B$, when their frames are statically equivalent.

5.3.6 Labelled bisimilarity

Observational equivalence requires quantification over all extended contexts and is very difficult to use. Labelled bisimilarity represents a more practical alternative to observational equivalence that utilises static equivalence to establish equivalence between frames and then compares the dynamics of processes under a set of interactions with an outside observer that is specified by the labelled semantics of section 5.3.3.

Definition 19

Labelled bisimilarity $(\approx_l)$ *is the largest symmetric relation* $\mathcal{R}$ *on closed extended processes such that* $A \mathcal{R} B$ *implies:*

1. $A \approx_s B$;
2. *if* $A \longrightarrow^* A'$ *then there exists* B' *s.t.* $B \longrightarrow^* B'$ *and* $A' \mathcal{R} B'$;
3. *if* $A \xrightarrow{\alpha} A'$ *and* $fv(\alpha) \subseteq dom(A)$ *and* $bn(\alpha) \cap fn(B) = \varnothing$, *then* $B \longrightarrow^* \xrightarrow{\alpha} \longrightarrow^* B'$ *and* $A' \mathcal{R} B'$

An important theorem per Abadi and Fournet is as follows:

Theorem 1

Observational equivalence is labelled bisimilarity $(\approx = \approx_l)$.

This theorem was introduced by Abadi and Fournet in their paper "Mobile Values, New Names and Secure Communication" [3] and we make use of this in our security analysis of chapter 8 without further discussion or proof.

5.4 Summary

In this chapter we present the syntax and semantics of the applied pi calculus together with such additions as we consider necessary for our access control modelling environment. Whilst the basic applied pi calculus of Abadi and Fournet [3] is sufficiently expressive for our needs we add some syntactic sugar to simplify the modelling process such as triggers and the sequential placeholder τ.

In the next chapter we present the core components of our implementation of BPAC in the applied pi calculus.

Chapter 6

The applied pi calculus implementation of BPAC

6.1 Introduction

In the previous chapter we describe the applied pi calculus as the basis for our model implementation of BPAC. In this chapter we present the core components of the access control modelling environment.

For the purpose of modelling workflow-based access control, we identify three core processes that we use to represent the key components of a workflow-based access control system: the agent, the task and the workflow manager. These three processes interact in a specified way that we call the standard core protocol. Also, we specify store and set handling processes that provide the toolkit for implementation of role-based access control with dynamic constraints based upon access history. Next, we discuss the initialisation process, which is a crucial component of the workflow modelling environment. This process initialises names used in the model, runs the set and store processes and pre-populates the stores as necessary. Finally, we present the translation of the BPAC environment to the applied pi calculus implementation followed by a summary of the complete model.

6.2 The programming environment: the applied pi calculus

We present the applied pi calculus, its syntax and semantics, together with the additional syntactic components required for our modelling environment in chapter 5. We adopt the equational theory specified in section 5.3 of chapter 5 and this provides us with functions that can be used, for example, to replicate

key creation, public key encryption and decryption and signature creation and verification as well as tuple creation and deconstruction.

6.3 The applied pi calculus model formalism

6.3.1 The standard core protocol

Figure 6.1 displays the process components that we use to create workflow-based access control models with the applied pi calculus. The interaction between these processes follows a prescribed form, which we refer to as the standard core protocol, and complex workflow examples are constructed from a number of standard core protocol interactions linked together by Puhlmann and Weske trigger constructs [77]. The full detail of the coding of the standard core protocol is presented in appendix A but the principle feature of the protocol is that agent, workflow manager and task processes communicate with each other using public key encryption over a public (unbound) channel. Access control policies are tested by the workflow manager against store processes via private communication channels. The stores are updated by task processes as necessary to provide stateful access control information for dynamic constraints.

6.3.2 Tasks

Task processes are representations of BPAC tasks within the applied pi calculus model. We are not concerned with modelling details of a specific task, that is, neither the details of work performed within a task nor the precise identity of resources associated with the task are of concern to us. However, we are interested in how these tasks interact with agent and workflow manager processes given a collection of access control constraints. Task processes interact with stores by writing access control state information to them over internal private communication channels.

6.3.3 Agents

Agent processes are used within the applied pi calculus workflow model to represent human or computer agents in the BPAC environment. A typical workflow model incorporates a number of unique agent processes that are capable of public channel encrypted interaction with workflow tasks via a workflow manager.

6.3.4 The workflow manager

The workflow manager process is a central mediator between agent, task and store processes. We do not have a direct analogy for this process within the BPAC model but we can consider this process as facilitating the application of the workflow-based access control policy within BPAC. The workflow manager comprises a number of sub-processes called task-calls that are linked together using triggers to represent the workflow structure of the BPAC model.

6.3.4.1 Task-call sub-processes

Task-call sub-processes implement access control policy in respect of specific task and agent process interactions. These sub-processes test access control constraints for agent processes seeking access to tasks against information in the stores, over internal private channels. If the constraint is satisfied, the sub-process initiates encrypted communication over a public channel between the agent and task processes, otherwise the process halts.

6.3.5 The stores

In order that we can properly model static and dynamic constraints over workflow models we set up a number of memory stores for our static and dynamic data. We can write information to these stores via suitable processes. Also, we can test information as data functions against the stored information and we can delete store entries if necessary. Usually, we adopt a store for our agent/role assignments, which we call R, and a dynamic constraint store S. Workflows modelled under BPAC enact access control policy rules through private channel interaction between workflow processes and the stores. We discuss stores in greater detail later in this chapter.

6.3.6 The initialisation process — $\mathcal{M}$

The initialisation process $\mathcal{M}$ has no analogy in the BPAC model. Within the applied pi calculus model we require certain names to be declared so that they are private in scope across the model. These names include key seeds for public key encryption as part of the core protocol that have to be kept private and secret from an outside observer of the workflow model. Also, the stores and supporting processes have to be run as ongoing parallel processes for the duration of the model run and these have to be pre-populated with standing information such as agent/role assignment. Process $\mathcal{M}$ satisfies this requirement prior to the running of the workflow model.

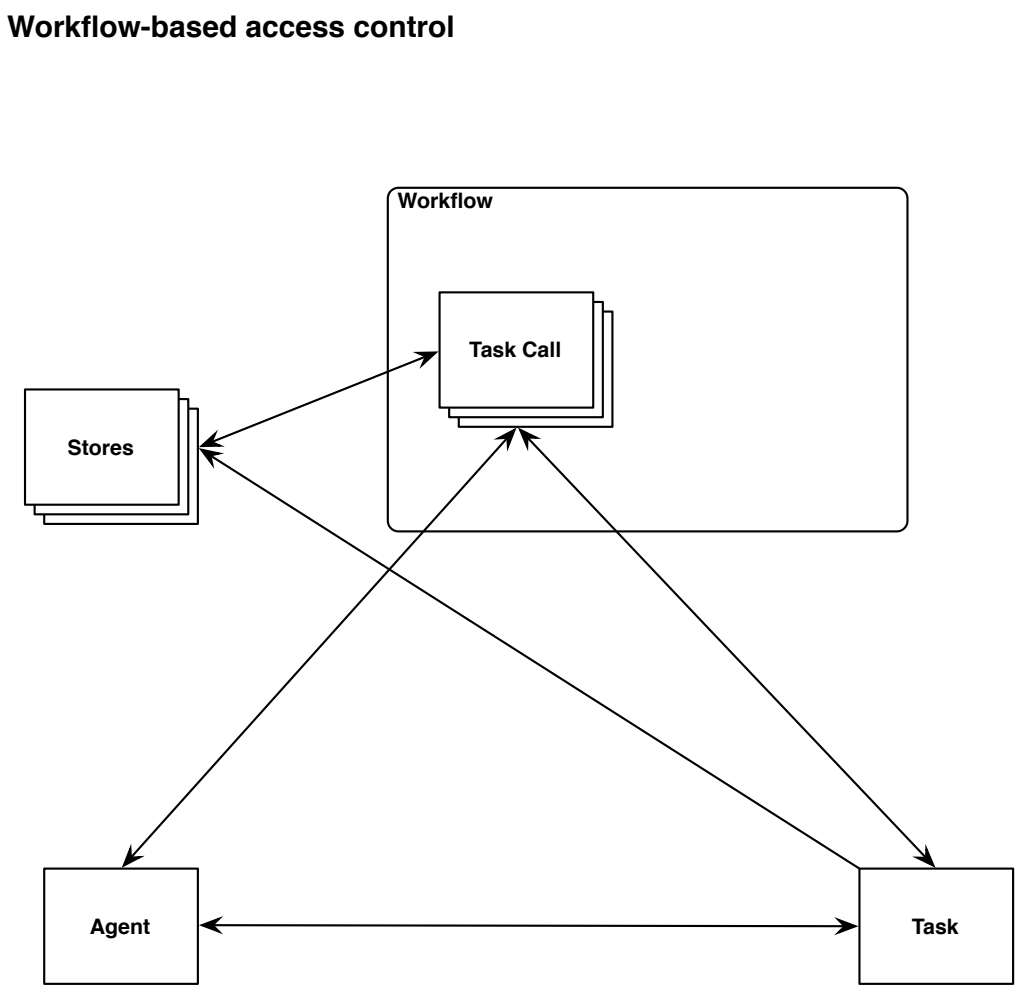


Figure 6.1: the standard core protocol

6.3.7 Summary

We outline the basic processes that are used to generate our workflow model in the applied pi calculus and we discuss how these processes represent the BPAC access control model. We return to each of these sections in more detail in the following sections as we present the macros used to build our representation of the workflow model in the applied pi calculus.

6.4 The standard core protocol

When the workflow manager W per section 6.7 interacts, via a single task-call sub-process, with a single task T_j per section 6.5 and a single agent A_i per 6.6, as indicated by figure 6.1, then this interaction comprises a number of encrypted communication steps over a public channel as discussed in detail in appendix A. We call this interaction the standard core protocol C. Complex workflows are built from a number of instances of the standard core protocol constructed by linking together multiple task-call sub-processes using Puhlmann and Weske trigger patterns [77].

6.5 Tasks

The task process represents a collection of resources associated with an atomic process within an organisation as per the BPAC model. For the purpose of our model, the task is represented as an applied pi calculus process that engages in encrypted communication over a public channel with the workflow manager and agents. Successful access to a task by an agent is represented by the passing of a message M_{T_j} from the j^{th} task process to the agent. If a process has access to the message M_{T_j}, it has access to the resources of task T_j. We use the following syntactic sugar to represent the core components of a task process T_j, the details of which are outlined in appendix A:

$$task\left\langle\left\langle ID_{T_j}, K_{T_j}, M_{T_j}, [\,], [\,]\right\rangle\right\rangle$$

where ID_{T_j} is an identity name for the task created by the workflow manager, K_{T_j} is a key seed associated with public key encrypted communication of the task process and M_{T_j} represents the task resources. The pair of holes indicated by [] are spaces for possible interactions between the task process and the stores. These spaces are optional. The detailed coding of the task process, as outlined in appendix A section A.5.1, includes as standard a write command to store S that records that an agent has accessed

the task: $\mathcal{W}(S)(hasAccessed(ID_A, ID_T, ID_W))$. Usually these holes are populated with a τ to indicate an internal process as the inbuilt write commands are generally sufficient.

The term ID_{T_j} is created and passed to the task process by the workflow manager process and the terms K_{T_j} and M_{T_j} are free in $task\langle\langle ID_{T_j}, K_{T_j}, M_{T_j}, [\,], [\,]\rangle\rangle$. Key seed K_{T_j} is declared new as part of the initialisation process $\mathcal{M}$ whilst M_{T_j} is a public name.

Based on the above, a task $Task_a$ without additional user-generated code is defined as follows:

$$Task_a \triangleq task\langle\langle ID_{T_a}, K_{T_a}, M_{T_a}, [\tau], [\tau]\rangle\rangle$$

and a task $Task_b$ that deletes a term $review(ID_A, p)$ from the store S and writes a new term $submitted(ID_A, p)$ to the store S via private channels is written as follows:

$$\begin{aligned} Task_b \triangleq\ & task\langle\langle ID_{T_b}, K_{T_b}, M_{T_b}, [\mathcal{D}(S)(review(ID_A, p))], \\ & [\mathcal{W}(S)(submitted(ID_A, p))]\rangle\rangle \end{aligned}$$

6.6 Agents

The agent process represents an employee (or perhaps a computer program) within an organisation who is required to interact with resources subject to access control constraints. Within our applied pi calculus modelling environment we represent the core components of an agent process A_i by the following syntactic sugar, a detailed explanation of which is presented in appendix A:

$$agent\langle\langle ID_{A_i}, K_{A_i}\rangle\rangle$$

where ID_{A_i} is an agent identifier name and K_{A_i} is a key seed associated with agent A_i. This agent process uses public key encryption, with private and public keys generated from the key seed K_{A_i}, to communicate over a public channel with workflow and task processes. Generally, ID_{A_i} represents a secret known by both the organisation and the agent/employee and K_{A_i} is likely to be known by the agent only. We declare both names as fresh at the start of the model run so that ID_{A_i} can be used within the stores for agent/role assignment and K_{A_i} remains constant over multiple instances of the agent process.

6.7 The workflow manager

The workflow manager initiates a task session in accordance with a workflow blueprint and coordinates the communication between the agent and the task. In particular, the workflow manager tests a communicating agent against the sets of constraints contained within the stores and allows or otherwise the direct interaction between agent and task processes.

A workflow is built up from parallel sub-processes, which we call task-calls. Task-calls are linked by internal communication that recreates workflow patterns in the style of Puhlmann and Weske [77].

6.7.1 Implementing constraints

As we discuss in previous chapters, workflow-based access control depends upon the verification of static and dynamic constraints. Static constraints include such properties as agent/role and task/role assignments and in the BPAC model per chapter 3 we specify agent/role assignments as pairs. For example, the pair:

$$(Agent_1, employee)$$

represents the fact that agent $Agent_1$ is mapped to the role *employee*. In the applied pi calculus modelling environment this data pair is represented by the data function, where data functions are as defined in section 5.3:

$$role\,(ID_{Agent_1}, employee)$$

An example of a task/role assignment pair given in chapter 4 is:

$$(Task_1, employee)$$

representing the access control constraint that the agent assigned to task $Task_1$ must have the role *employee*. In our modelling environment this access control constraint is represented by a test performed by a workflow manager process against a store of agent/role assignments:

$$\mathcal{T}(R)(role(id_A, employee)):$$

This can be interpreted as: test agent/role store R to see if there exists data function $role(id_A, employee)$ for some agent identity substituted for id_A (id_A is a variable that is assigned the identity name of the agent that is attempting to access a specific task) representing the agent/role assignment pair:

$$(Agent, employee)$$

and if true then continue. Note that the task identity is not included within the test syntax. This is because the test is incorporated within a task-call sub-process of the workflow process that calls the relevant task.

Dynamic constraints, such as agent/task access, are represented in a similar manner to static constraints only this time the data function representing the constraint is written to a different constraint store S during the workflow process and tests against the constraint store are performed within the task-call processes of the workflow manager.

An example of a dynamic constraint given in chapter 4 is:

$$(Task_2, test(\neg hasAccessed(a, Task_1, W), \mathcal{W}))$$

meaning that the agent that accesses task $Task_2$ cannot be the same agent who accessed task $Task_1$ under store/world $\mathcal{W}$.

In our modelling environment this constraint is implemented using a data function that represents the fact that an agent has accessed some task. The data function is written to the store S by a task process. A subsequent task-call sub-process within the workflow process tests for the existence of the data function within the store S. For example, our dynamic constraint given above is represented as follows:

Firstly, task process $Task_1$ writes to the store S that an agent, $Agent_1$, has accessed $Task_1$ within workflow W, using the following syntax:

$$\mathcal{W}(S)(hasAccessed(ID_{Agent_1}, ID_{Task_1}, ID_W))$$

where ID_{Agent_1}, ID_{Task_1} and ID_W are unique identifiers respectively for agent $Agent_1$, task $Task_1$ and workflow W. The term $hasAccessed()$ is a three argument data function such that:

$$hasAccessed(ID_{Agent_1}, ID_{Task_1}, ID_W)$$

can be interpreted as agent $Agent_1$ has accessed task $Task_1$ within workflow W.

Having written the data function to the store, a test on the existence of the data function within the store S can now be performed by the task-call sub-process of the workflow process W. For example, if we wish to constrain access to task $Task_2$ as per our example above, the task-call process within workflow W, which facilitates access to task $Task_2$, tests using the following syntax:

$$\mathcal{T}(S)(hasAccessed(id_A, ID_{T_1}, ID_W)) : \mathbf{0} :$$

This can be interpreted as: if the data function $hasAccessed(id_A, ID_{T_1}, ID_W)$ with an agent identity substituted for variable id_A exists in store S then halt the process otherwise continue.

6.7.2 Task-calls

Task-calls are the main working components of workflow processes. A task-call mediates the interaction between an agent and a task through interrogation of a number of stores. Task-calls never exist outside of a workflow manager process. We represent the task-call process by the following syntactic sugar:

$$taskCall\langle\langle ID_{T_m}, K_{T_m}, ID_W, K_W, []\rangle\rangle$$

Where ID_{T_m} is a fresh identity for a new task instance, K_T is the key seed associated with the task process and ID_W and K_W represent the identity and key seed associated with the workflow manager process W. The hole is used to specify the query or nest of queries that are applied against the stores.

An example of a task-call incorporated within a workflow manager process W (as indicated by the inclusion of identity ID_W and key seed K_W) using this syntax is:

$$\begin{aligned} TaskCall_2 \triangleq\ & \nu ID_{T_2}.taskCall\,\langle\langle ID_{T_2}, K_{T_2}, ID_W, K_W, \\ & [(\mathcal{T}\,(R)\,(role\,(ID_A, clerk)) : \\ & \mathcal{T}\,(S)\,(hasAccessed\,(ID_A, ID_{T_1}, ID_W)) : \mathbf{0} :)]\rangle\rangle \end{aligned}$$

where an instance of the task process identified by the key seed K_{T_2} is assigned a fresh identity ID_{T_2} and the task is advertised on the public channel. Once the task identity has been picked up by an agent process then the task manager checks the credentials of the agent against the stores using the $\mathcal{T}\,(R)$ and $\mathcal{T}\,(S)$ test processes. On a satisfactory outcome the workflow manager then facilitates communication between the successful agent and the task.

6.7.3 Combining task-calls to create workflows

In order to create our workflow-based access control models we combine a number of parallel task-calls represented by the sub-process $TaskCall$ using Puhlmann and Weske workflow modelling [77]. Firstly, we initialise channel triggers t_n for the workflow and task identities ID_T. The reason that these names are declared within the workflow process W and not within the initialisation process $\mathcal{M}$ is that we want to ensure that these names are always fresh for each workflow model instance. This ensures that multiple instances of workflows do not trigger each other and freshness of task identities contributes to the uniqueness property of our modelling environment, which is discussed in chapter 8.

A simple example of workflow manager coding is as follows:

Consider a simple workflow comprising four tasks, task 1 is followed by both tasks 2 and 3 in parallel then task 4 follows after the completion of both tasks 2 and 3. Our workflow manager for this particular workflow would be as follows assuming that there are no access constraints:

Control flow patterns	Name
$P_1.\overline{t_1} \mid t_1.Q_1$	Sequence
$P_1.\left(\overline{t_1} \mid \overline{t_2}\right) \mid t_1.Q_1 \mid t_2.Q_2$	Parallel split
$P_1.\overline{t_1} \mid P_2.\overline{t_2} \mid t_1.t_2.Q_1$	Synchronization
$P_1.\left(\overline{t_1} + \overline{t_2}\right) \mid t_1.Q_1 \mid t_2.Q_2$	Exclusive choice
$P_1.\overline{t_1} \mid P_2.\overline{t_1} \mid t_1.Q_1$	Simple merge

Table 6.1: Basic Puhlmann and Weske workflow constructs

$$
\begin{aligned}
W \triangleq\ & \nu t_2, t_3, t_{4a}, t_{4b}, ID_W, ID_{T_1}, ID_{T_2}, ID_{T_3}, ID_{T_4}. \\
& \left(taskCall\left\langle\left\langle ID_{T_1}, K_{T_1}, ID_W, K_W, []\right\rangle\right\rangle.\left(\overline{t_2} \mid \overline{t_3}\right)\right. \\
& \mid t_2.taskCall\left\langle\left\langle ID_{T_2}, K_{T_2}, ID_W, K_W, []\right\rangle\right\rangle.\overline{t_{4a}} \\
& \mid t_3.taskCall\left\langle\left\langle ID_{T_3}, K_{T_3}, ID_W, K_W, []\right\rangle\right\rangle.\overline{t_{4b}} \\
& \left.\mid t_{4a}.t_{4b}.taskCall\left\langle\left\langle ID_{T_4}, K_{T_4}, ID_W, K_W, []\right\rangle\right\rangle\right)
\end{aligned}
$$

This example demonstrates the coding for a workflow incorporating series and parallel components as follows:

following $taskCall\left\langle\left\langle ID_{T_1}, K_{T_1}, ID_W, K_W, []\right\rangle\right\rangle$, two parallel triggers $\left(\overline{t_2} \mid \overline{t_3}\right)$ are sent that are received by processes $t_2.taskCall\left\langle\left\langle ID_{T_2}, K_{T_2}, ID_W, K_W, []\right\rangle\right\rangle.\overline{t_{4a}}$ and $t_3.taskCall\left\langle\left\langle ID_{T_3}, K_{T_3}, ID_W, K_W, []\right\rangle\right\rangle.\overline{t_{4b}}$. On completion, both of these processes send triggers $\overline{t_{4a}}$ and $\overline{t_{4b}}$ and both of these have to be received by parallel process $t_{4a}.t_{4b}.taskCall\left\langle\left\langle ID_{T_4}, K_{T_4}, ID_W, K_W, []\right\rangle\right\rangle$ in order for it to continue.

There are numerous alternative coding constructs identified by Puhlmann and Weske to enable various complex workflow constructs, like path choice, multi-choice and multiple instances, to be encoded in the pi calculus (and the applied pi calculus by extension). We summarise the basic constructs in table 6.1 where P_1, P_2, Q_1 and Q_2 are plain processes and t_1 and t_2 are triggers.

6.8 Modelling the store

In section 6.3.5 we introduce stores as a key requirement for modelling workflow-based access control. A store provides the means by which static and dynamic segregation of duties is enforced across tasks within workflows. In essence, when an agent is permitted access to task resources then the agent identity, role and task identity are written to a store. Subsequent access attempts to task resources by agents can then be tested against the store as part of the task access control policy.

For example, we consider a simple workflow W comprising two sequential tasks, T_1 and T_2, two agents A_1 and A_2, agent roles *employee* and *clerk* with role *employee* assigned to A_1 and roles *employee* and *clerk* assigned to A_2. We define an access control policy as follows:

1. Task T_1 can be accessed by an agent with role *employee*
2. Task T_2 can be accessed by an agent with role *clerk*
3. Task T_2 cannot be accessed by an agent who has accessed task T_1

Policy rule 3. enforces dynamic segregation of duties across the workflow, ensuring that no one agent can have complete control of the workflow and workflow resources.

To model this access control example we require a set of agent/role assignments R that can be interrogated by the model and a store S. Constraints are represented as data functions as we discuss in section 6.7.1 above. In the case of task T_1, an agent attempts access to T_1 and the system queries the agent/role assignment for the agent against R. If the agent/role assignment satisfies the access rule 1, the system permits access for the agent and writes to the store S that the agent with role *employee* has accessed T_1.

Once access control has passed to T_2 and an agent attempts access to T_2 then the system verifies the agent/role assignment against R as before and additionally checks the store S as to whether or not the agent has accessed T_1. If the agent/role assignment satisfies rule 2 and if the agent has not accessed T_1, the system permits access for the agent and writes to the store that the agent with role *clerk* has accessed task T_2.

In applied pi calculus the store model requires processes to represent set handling:

6.8.1 Modelling sets

We use stores to retain both static and dynamic constraint information against which access control properties can be verified within an access control example. Initially, it was considered that a simple addition to our definition of function terms, coupled with a basic equational theory to provide idempotency and commutativity, as per Abadi and Blanchet [4, 22], would be sufficient for our store requirements. However, it was discovered that a membership deletion function is necessary for our modelling environment and such a function is inconsistent with the equational theory. Consequently, it was decided that sets would be implemented as lists with defined processes manipulating these lists. Strictly speaking, the sets produced by this method are represented by ordered lists where the order is based upon the temporal relationship of set updates.

We implement lists in the calculus as nested pairs so that the head of the list is the first element in the outermost pair, the tail of the list is the second element in the outermost pair and the store process initialises the list with an empty set symbol.

Definitions of key processes for set manipulation are given below. The processes are hierarchical in that *headSet* and *tailSet* extend the pair destructors that are defined within our applied pi calculus syntax in chapter 5 so as to properly output the empty set symbol if that is the only member of the list. The *memberSet* process utilises *headSet* and *tailSet* to identify whether or not an input message is present or otherwise in the list. The *insertMem* process utilises *memberSet* and adds an input message to a list if it is not already in the list. The *deleteMem* process utilise *memberSet*, the *headSet* and *tailset* processes to remove an input message from the list if it is in the list.

The processes as stated initially utilise an unbound channel *set* for communication with other processes and messages are not encrypted. Subsequently, fresh channels provided by the communicating processes are utilised. In practice, it is assumed that set handling is an internal server matter and channel *set* is restricted in scope over the extent of the access control model. Consequently, the contents of these processes are not exposed to an attacker.

6.8.1.1 headSet process

This process as listed below takes an input message and checks that the first component of the message matches the instruction term *head*. If the third component M matches the empty set term $\varnothing$ then, utilising the second component as a fresh channel set_1, the process returns a message $\varnothing$ otherwise the process returns the first element of M on channel set_1.

$$
\begin{aligned}
headSet \quad \triangleq \quad & set\,(instr, set_1, M) \\
& .\textbf{if } instr = head \textbf{ then} \\
& \textbf{if } M = \varnothing \textbf{ then } \overline{set_1}\,\langle \varnothing \rangle \\
& \textbf{else } \overline{set_1}\,\langle fst\,(M) \rangle
\end{aligned}
$$

6.8.1.2 tailSet process

This process as listed below takes an input message and checks that the first component of the message matches the instruction term *tail*. If the third component M matches the empty set term $\varnothing$ then, utilising

the second component as a fresh channel set_1, the process returns a message $\varnothing$ otherwise the process returns the second element of M on channel set_1.

$$
\begin{aligned}
tailSet \quad \triangleq \quad & set\,(instr, set_1, M) \\
& \textbf{if } instr = tail \textbf{ then} \\
& \textbf{if } M = \varnothing \textbf{ then } \overline{set_1}\,\langle\varnothing\rangle \\
& \textbf{else } \overline{set_1}\,\langle snd\,(M)\rangle
\end{aligned}
$$

6.8.1.3 memberSet process

The *memberSet* process represents an element member check against the head element in a list followed, if a match is not achieved, by a recursive procedure on the tail of the list.

The process listed below first creates an internal private channel set_1 then it takes an input message comprising four terms and checks that the first component of the message matches the instruction term *member*. Subsequently, the process creates a new instance of an internal parallel process by internally passing a message comprising the second, third and fourth elements of the original message. The second element represents a fresh channel, the third element represents a name to be checked for membership against the fourth element that represents a list. If the fourth element, M, matches the empty set term $\varnothing$ then, utilising the second component as a fresh channel set_1, the process returns a message $false$. Alternatively, the process makes a call to the *headSet* process with M and receives a message *headM* on a fresh channel (being the first element of the list), then a call to *tailSet* with M and receives a message *tailM* (being the tail of the list). Now the process checks the test variable against the head variable *headM* and sends message *true* on the fresh channel, otherwise a new instance of the process is initialised using message *tailM*.

$$
\begin{aligned}
memberSet \triangleq\ & \nu set_1.\,(set\,(instr, set_4, x_1, M_1) \\
& .\mathbf{if}\ instr = member\ \mathbf{then}\ \overline{set_1}\,\langle set_4, x_1, M_1\rangle\big) \\
& |!\,(set_1\,(mem, x, M)\,.\mathbf{if}\ M = \varnothing\ \mathbf{then}\ \overline{mem}\,\langle false\rangle \\
& \mathbf{else}\ \nu headch.\,\big(\overline{set}\,\langle head, headch, M\rangle\,.headch\,(headM)\big) \\
& .\nu tailch.\,\big(\overline{set}\,\langle tail, tailch, M\rangle\,.tailch\,(tailM)\big) \\
& .\mathbf{if}\ x = headM\ \mathbf{then}\ \overline{mem}\,\langle true\rangle \\
& \mathbf{else}\ \nu set_5.\,\big(\overline{set_1}\,\langle set_5, x, tailM\rangle\big)\big)\big)
\end{aligned}
$$

6.8.1.4 insertMem process

This process represents element insertion into a list provided that the element is not already present in the list.

On receipt of the message on the channel *set*, the process tests the first component of the message against the term *insert* and if true the *memberSet* process is called by sending a four element message over the *set* channel. The first element comprises the term *member*, the second element is a fresh return channel, the third element is the name to be inserted in the list and the fourth element is the list itself. If the returned variable matches the term *true*, the process outputs the list, otherwise the process returns the list paired with the inserted name.

$$
\begin{aligned}
insertMem \triangleq\ & (set\,(instr, set_4, x_1, M_1) \\
& .\mathbf{if}\ instr = insert\ \mathbf{then} \\
& \nu set_2.\,\big(\overline{set}\,\langle member, set_2, x_1, M_1\rangle\,.set_2\,(memM)\big) \\
& .\mathbf{if}\ memM = true\ \mathbf{then}\ \overline{set_4}\,\langle M_1\rangle\big) \\
& .\mathbf{else}\ \overline{set_4}\,\langle (x_1, M_1)\rangle\big)
\end{aligned}
$$

6.8.1.5 deleteMem process

The *deleteMem* process proceeds on broadly similar lines to the *memberSet* process already discussed above in that an empty set check is followed by a check against the head of the list and a return of the tail if there is a match. If there is not a match, a recursive process strips successive head elements from

the list until or otherwise a match is obtained. The list is then successively rebuilt without the deleted element if appropriate.

Following the initialisation of fresh internal channels, a message is received from some process on the *set* channel and if the first component of the message matches the term *delete*, a fresh parallel internal process is instantiated by sending the second component (representing a process-generated channel), third component (representing the name to be removed from the list) and the fourth component (representing the list). Subsequently, the process performs the empty set check as previously discussed and returns a message with value $\varnothing$, otherwise the process calls the *headSet* and *tailSet* processes to return respectively the head and tail of the list. Now the variable that takes the element to be deleted as a message value is compared to the variable returned from the *headSet* process and if true then the *tailSet* variable is returned. If the match is false, a fresh parallel delete process is called for the *tailSet* variable and following the return message from the fresh process the combined *headSet* variable and the return message is combined as a pair and returned. Ultimately, the reconstructed list as a pair is returned to the calling process.

$$
\begin{aligned}
deleteMem \triangleq\ & \nu set_1.\,(\nu set_2.\,(set\,(instr, set_4, x_1, M_1) \\
& .\mathbf{if}\ instr = delete\ \mathbf{then} \\
& \overline{set_1}\,\langle set_2, x_1, M_1\rangle\,.set_2\,(delM)\,.\overline{set_4}\,\langle delM\rangle\big) \\
& |!\,\big(set_1\,(set_3, x, M)\,.\mathbf{if}\ M = \varnothing\ \mathbf{then}\ \overline{set_3}\,\langle\varnothing\rangle \\
& \mathbf{else}\ \nu headch.\,\big(\overline{set}\,\langle head, headch, M\rangle\,.headch\,(headM)\big) \\
& .\nu tailch.\,\big(\overline{set}\,\langle tail, tailch, M\rangle\,.tailch\,(tailM)\big) \\
& .\mathbf{if}\ x = headM\ \mathbf{then}\ \overline{set_3}\,\langle tailM\rangle \\
& \mathbf{else}\ \nu set_5.\,\big(\overline{set_1}\,\langle set_5, x, tailM\rangle\,.set_5\,(nextM) \\
& .\overline{set_3}\,\langle(headM, nextM)\rangle\big)\big)\big)
\end{aligned}
$$

6.8.2 Encoding store processes

We model the historical stores within our modelling environment using an adaptation of the duplicate message filter defined by Abadi, Blanchet and Fournet for the modelling of the Just Fast Keying protocol in the applied pi calculus [4], together with the collection of set processes outlined in section 6.8.1. We model three stores for our particular model although there is no restriction on store numbers in principle. Store Q represents a public key server, store S accumulates dynamic access control information and store

R accumulates static access control information such as role assignments. It is envisaged that the stores be kept private and not exposed to an attacker. Consequently, internal store communication proceeds over fresh private channels and communication with protocol processes proceeds over channels restricted in scope over the entire model space.

The store processes interact with the processes of section 6.8.1 to perform set updates using the channel *set* that is restricted over the model space and fresh channels created by the stores.

System processes interact with the stores Q, R and S by sending a triple over the system-wide private channels *storeQ*, *storeR* or *storeS* comprising some data function representing a constraint predicate term, a 0-arity data function symbol, either *write*, *test* or *delete* and a channel for receiving the result of store processing. The result of store processing is a 0-arity data function symbol, either *done* or *hit* or *miss*.

All stores proceed in a similar way. The store creates an internal channel, then passes the empty set term $\varnothing$ over the channel to instantiate the ongoing store process. On receipt of this message the process is then in a wait state until a message triple is received on the external store communication channel, either *storeQ*, *storeS* or *storeR*, scoped over the model space. The variable representing the second element of the message is compared to a *write* term and if true then the InsertMem process is called and the updated store is received as the *newStore* variable. The *newStore* variable is then sent over the internal channel to create a fresh instance of the store process and term *done* is sent as a message back to the protocol process. If the second element is successfully compared to the *test* term, the *memberSet* process is called. If *true* is returned, the term *hit* is sent as a message back to the protocol process, otherwise *miss* is sent as a message instead. Finally, if the second element is successfully compared to the *delete* term, the *deleteMem* process is called and the updated store is received as the *memberDelete* variable. The *memberDelete* variable is sent over the internal channel and the term *done* is sent as a message back to the protocol process.

$$
\begin{aligned}
S \triangleq\ & \nu st.\left(\overline{st}\,\langle\varnothing\rangle \mid !\,(st\,(store)\,.storeS\,(z_1,z_2,z_3)\,.\right.\\
& (\textbf{if } z_2 = write \textbf{ then}\\
& \nu set_4.\left(\overline{set}\,\langle insert, set_4, z_1, store\rangle\,.set_4\,(newStore)\right).\\
& \overline{st}\,\langle newStore\rangle\,.\overline{z_3}\,\langle done\rangle\\
& \textbf{else if } z_2 = test \textbf{ then } \left(\overline{st}\,\langle store\rangle\,.\right.\\
& \nu set_5.\left(\overline{set}\,\langle member, set_5, z_1, store\rangle\,.set_5\,(memberTest)\right.\\
& \left.\left.\left.\left.\,.(\textbf{if } memberTest = true \textbf{ then } \overline{z_3}\,\langle hit\rangle \textbf{ else } \overline{z_3}\,\langle miss\rangle)\right)\right)\right)\\
& \textbf{else if } z_2 = delete \textbf{ then}\\
& \nu set_6.\left(\overline{set}\,\langle delete, set_6, z_1, store\rangle\,.set_6\,(memberDelete)\right.\\
& \left..\overline{st}\,\langle memberDelete\rangle\,.\overline{z_3}\,\langle done\rangle\right)\\
& \left.\left.\textbf{else } \overline{st}\,\langle store\rangle\right)\right)\right)
\end{aligned}
$$

and a similar process can be used to model the set of agent/role assignments:

$$
\begin{aligned}
R \triangleq\ & \nu rt.\left(\overline{rt}\,\langle\varnothing\rangle \mid !\,(rt\,(store)\,.storeR\,(z_1,z_2,z_3)\,.\right.\\
& (\textbf{if } z_2 = write \textbf{ then}\\
& \nu set_4.\left(\overline{set}\,\langle insert, set_4, z_1, store\rangle\,.set_4\,(newStore)\right).\\
& \overline{rt}\,\langle newStore\rangle\,.\overline{z_3}\,\langle done\rangle\\
& \textbf{else if } z_2 = test \textbf{ then } \left(\overline{rt}\,\langle store\rangle\,.\right.\\
& \nu set_5.\left(\overline{set}\,\langle member, set_5, z_1, store\rangle\,.set_5\,(memberTest)\right.\\
& \left.\left.\left.\left.\,.(\textbf{if } memberTest = true \textbf{ then } \overline{z_3}\,\langle hit\rangle \textbf{ else } \overline{z_3}\,\langle miss\rangle)\right)\right)\right)\\
& \textbf{else if } z_2 = delete \textbf{ then}\\
& \nu set_6.\left(\overline{set}\,\langle delete, set_6, z_1, store\rangle\,.set_6\,(memberDelete)\right.\\
& \left..\overline{rt}\,\langle memberDelete\rangle\,.\overline{z_3}\,\langle done\rangle\right)\\
& \left.\left.\textbf{else } \overline{rt}\,\langle store\rangle\right)\right)\right)
\end{aligned}
$$

and a public key server:

$$
\begin{aligned}
Q \triangleq\ & \nu qt.\left(\overline{qt}\left\langle\varnothing\right\rangle \mid !\left(qt\left(store\right).storeQ\left(z_1,z_2,z_3\right).\right.\right.\\
& \left(\textbf{if } z_2 = write \textbf{ then}\right.\\
& \nu set_4.\left(\overline{set}\left\langle insert, set_4, z_1, store\right\rangle.set_4\left(newStore\right)\right).\\
& \overline{qt}\left\langle newStore\right\rangle.\overline{z_3}\left\langle done\right\rangle\\
& \textbf{else if } z_2 = test \textbf{ then } \left(\overline{qt}\left\langle store\right\rangle.\right.\\
& \nu set_5.\left(\overline{set}\left\langle member, set_5, z_1, store\right\rangle.set_5\left(memberTest\right)\right.\\
& .\left(\textbf{if } memberTest = true \textbf{ then } \overline{z_3}\left\langle hit\right\rangle \textbf{ else } \overline{z_3}\left\langle miss\right\rangle\right)\left.\left.\left.\left.\right)\right)\right)\\
& \textbf{else if } z_2 = delete \textbf{ then}\\
& \nu set_6.\left(\overline{set}\left\langle delete, set_6, z_1, store\right\rangle.set_6\left(memberDelete\right)\right.\\
& .\left.\overline{qt}\left\langle memberDelete\right\rangle.\overline{z_3}\left\langle done\right\rangle\right)\\
& \left.\left.\textbf{else } \overline{qt}\left\langle store\right\rangle\right)\right)\right)
\end{aligned}
$$

6.8.3 Syntactic sugar for the interaction between processes and the stores

In section 6.8.2 above we discuss the coding of the stores and their interactions with set manipulation processes. In this section we discuss the interaction between protocol processes and the stores in greater detail and we adopt syntactic sugar to simplify the reading of these interactions.

Interaction between processes (the workflow manager and tasks) and the stores follows a consistent pattern within the modelling of the workflow: agents send a message to a store and receive a reply. In the case of a write instruction an agent identifies a store and sends a data function representing a constraint term to that store. The store replies with an acknowledgement. We use the following syntactic sugar to represent this agent/store behaviour:

$$
\begin{aligned}
\mathcal{W}\left(STORE\right)\left(dataFunction\left(a_1,\ldots,a_n\right)\right) =\ & \nu pteChannel.\\
& \left(\overline{storeX}\left\langle\right.\right.\\
& dataFunction\left(a_1,\ldots,a_n\right),\\
& \left.write, pteChannel\right\rangle\\
& .pteChannel\left(x\right))
\end{aligned}
$$

where $STORE$ is the specific store to which the constraint expression is to be written, $storeX$ is a store-specific channel restricted over the model space and not visible to an attacker and data function (as defined in section 5.3) $dataFunction\,(a_1,\ldots,a_n)$ represents a constraint term together with n arguments. For example:

$$\begin{aligned}\mathcal{W}\,(S)\,(hasAccessed\,(id_A,ID_T,ID_W)) \quad = \quad & \nu pteChannel.\,(\overline{storeS}\,\langle \\ & hasAccessed\,(id_A,ID_T,ID_W)\,, \\ & write, pteChannel\rangle \\ & .pteChannel\,(x))\end{aligned}$$

writes to store S on channel $storeS$ the constraint data function term:

$$hasAccessed\,(id_A,ID_T,ID_W)$$

indicating that agent A has accessed task T of workflow W.

The syntactic sugar for a test of a store's contents is slightly more complex as the store response is conditional upon the successful matching of the queried constraint data term to the store contents. We adopt the following:

$$\begin{aligned}\mathcal{T}\,(STORE)\,(dataFunction\,(a_1,\ldots,a_n)) : P : Q \quad = \quad & \nu pteChannel.\,(\overline{storeX}\,\langle \\ & dataFunction\,(a_1,\ldots,a_n)\,, \\ & storeProcess, \\ & pteChannel\rangle \\ & .pteChannel\,(x)\ \textbf{if}\ x = hit \\ & \textbf{then}\ P\ \textbf{else}\ Q)\end{aligned}$$

wherein the $STORE$, $storeX$ and $dataFunction\,(a_1,\ldots,a_n)$ terms are as before but now we have two outcomes P and Q depending upon the result of the comparison test, i.e. if the store returns hit for a match to the store, P executes, otherwise Q executes. If we require the process to stop on a successful

match but execute Q otherwise, the term will be:

$$\mathcal{T}\left(STORE\right)\left(dataFunction\left(a_1,\ldots,a_n\right)\right):\mathbf{0}:Q$$

and conversely an execution of P following a successful match or otherwise a process halt is given by:

$$\mathcal{T}\left(STORE\right)\left(dataFunction\left(a_1,\ldots,a_n\right)\right):P:\mathbf{0}$$

that we abbreviate to:

$$\mathcal{T}\left(STORE\right)\left(dataFunction\left(a_1,\ldots,a_n\right)\right):P$$

For the purpose of our model we substitute the internal process τ for P or Q as a continuity marker and we restrict our consequent and alternative patterns to $P = \tau$ when $Q = \mathbf{0}$ and $Q = \tau$ when $P = \mathbf{0}$.

The syntactic sugar for deletion of a data function representing a constraint term from a store is as follows:

$$\begin{aligned}\mathcal{D}\left(STORE\right)\left(dataFunction\left(a_1,\ldots,a_n\right)\right) \quad = \quad & \nu pteChannel.\left(\overline{storeX}\left\langle\right.\right.\\ & dataFunction\left(a_1,\ldots,a_n\right),\\ & delete, pteChannel\rangle\\ & .pteChannel\left(x\right))\end{aligned}$$

where $STORE$, $storeX$ and $dataFunction$ are defined as for the *write* and *test* processes above. An example of the *delete* process in which the constraint data function $reviewer\left(id_A, p\right)$ is removed from the store S if it is present is as below:

$$\begin{aligned}\mathcal{D}(S)(reviewer(id_A, p)) \quad = \quad & \nu pteChannel.\left(\overline{storeS}\langle\right. \\ & reviewer(id_A, p), \\ & delete, pteChannel\rangle \\ & .pteChannel(x))\end{aligned}$$

6.9 Initialising the modelling environment — $\mathcal{M}$

In the previous sections we present the various core processes that comprise the building blocks for the access control modelling environment. In this section we address the important task of initialisation for models at runtime. The process $\mathcal{M}$ sets up private names for various secrets that have experiment-wide scope, such as cryptographic key seeds, and this component of $\mathcal{M}$ is discussed further in section 6.9.1. Next, $\mathcal{M}$ starts the set and store processes, which are active throughout the experiment as per section 6.9.2. Finally, process $\mathcal{M}$ pre-populates the stores as appropriate for the experiment, for instance, agent/role assignments are written to the role store and this is discussed in section 6.9.3. Section 6.9.4 draws together these concepts and provides a definition of the initialisation process $\mathcal{M}$.

6.9.1 Declaration of names

The declaration of new names is an important part of the applied pi calculus and plays a significant role within the context of our workflow modelling environment. Certain names within the modelling environment are persistent throughout the model run whilst others are created on an ad hoc basis. Also, the scope of declared names may vary depending upon workflow requirements. In the case of ad hoc names such as nonces then these are declared within the body of the workflow code. However, names such as agent identities ID_A and agent key seeds K_A that are required to be persistent across the model run are declared at the commencement of the model run and scoped across the agent, task and workflow processes as appropriate. In the workflow example of chapter 4 above we have sets of agents, tasks and the workflow process that require the initialising of identities and key seeds, together with private channels which are used for the store processes. Our initialisation sequence is as follows:

$$\nu K_W, K_{Org}, K_{T_1}, K_{T_2}, K_{T_3}, K_{T_4}, K_{T_5}, K_{T_6},$$
$$ID_{A_1}, ID_{A_2}, ID_{A_3}, ID_{A_4}, ID_{A_5}, ID_{A_6},$$
$$K_{A_1}, K_{A_2}, K_{A_3}, K_{A_4}, K_{A_5}, K_{A_6},$$
$$set, storeQ, storeR, storeS$$

Clearly, the content of this initialisation sequence depends upon the nature of the modelling that we wish to perform. Generally this sequence always includes the declaration of private names to represent agent identities ID_A, cryptographic key seeds K_A assigned to each agent, cryptographic key seeds K_T assigned to each task process and cryptographic key seeds for the workflow, K_W, and the organisation, K_{Org}. Channel names are declared for the set and store processes *set*, *storeQ* etc.

6.9.2 Running the set and store processes

Set and store processes have to run continuously for the purpose of our modelling environment and so we run them as part of our initialisation, $\mathcal{M}$. We always run stores Q, R and S for the public key server, agent/role assignment and dynamic access control information respectively and we summarise these support processes as χ in our initialisation sequence.

6.9.3 Population of stores

Whilst some stores, particularly store S, are used to store dynamic information at runtime other stores, such as the agent/role assignment store R, require populating prior to model runtime as these stores represent broadly static information for the modelling process. For example, let us consider the simple workflow example of chapter 4. We recall that in this example the agent/role assignment is represented by the following relation:

$$
\begin{aligned}
AR \;=\; \{&(Agent_1, employee)\,,\\
&(Agent_2, employee)\,,\ (Agent_2, clerk)\,,\\
&(Agent_3, employee)\,,\ (Agent_3, clerk)\,,\\
&(Agent_4, employee)\,,\ (Agent_4, clerk)\,,\\
&(Agent_4, supervisor)\,,\\
&(Agent_5, employee)\,,\ (Agent_5, supervisor)\}
\end{aligned}
$$

so that agent $Agent_1$ is assigned role *employee*, $Agent_2$ is assigned role *employee* and role *clerk* etc. In order to model this particular workflow example we populate store R with agent/role assignments represented by data functions, prior to the running of workflow W, using the syntax of the previous section as follows:

$$
\begin{aligned}
&\mathcal{W}(R)(role(ID_{Agent_1}, employee))\\
&.\mathcal{W}(R)(role(ID_{Agent_2}, employee))\\
&.\mathcal{W}(R)(role(ID_{Agent_2}, clerk))\\
&.\mathcal{W}(R)(role(ID_{Agent_3}, employee))\\
&.\mathcal{W}(R)(role(ID_{Agent_3}, clerk))\\
&.\mathcal{W}(R)(role(ID_{Agent_4}, employee))\\
&.\mathcal{W}(R)(role(ID_{Agent_4}, clerk))\\
&.\mathcal{W}(R)(role(ID_{Agent_4}, supervisor))\\
&.\mathcal{W}(R)(role(ID_{Agent_5}, employee))\\
&.\mathcal{W}(R)(role(ID_{Agent_5}, supervisor))
\end{aligned}
$$

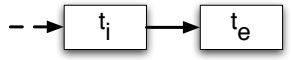
Once populated as above the store R retains its set of agent/role assignments for the duration of the model run unless they are deleted as part of the run.

6.9.4 The initialisation process $\mathcal{M}$

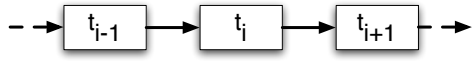
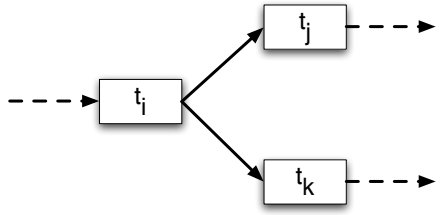
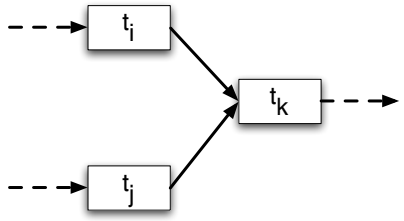
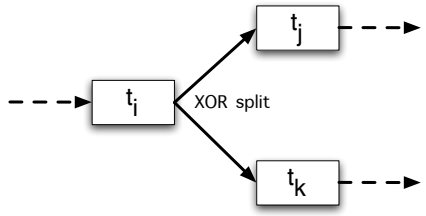
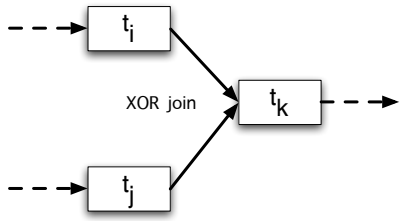
The initialisation process $\mathcal{M}$ for our workflow model combines the declaration of names per section 6.9.1, running the set and store processes of section 6.9.2 and populating the stores as appropriate as per section 6.9.3. Once $\mathcal{M}$ has been established then we can run any number of instances of our workflow model together with any other models that we may require for the purpose of security analysis.

6.10 Translation between BPAC and applied pi calculus

In this section we provide a full summary of the translation from the BPAC modelling environment of chapter 3 to the applied pi calculus implementation of the modelling environment presented in this chapter.

BPAC		Applied pi calculus
Component	Symbol/graphical representation	
Initialisation	No analogy in BPAC although pre-population of the store may sometimes be necessary.	$\mathcal{M}$ required to define names, run support processes, populate stores if necessary.
Task message	No analogy in BPAC	M_{T_i} message associated with tasks. Required to indicate agent access to tasks.
Workflow	W	$W = W_i \mid W_{i+1} \mid \ldots \mid T_i \mid T_{i+1}$
Role	r	$role$
Agent	a_i	$A_i = agent\left\langle\left\langle ID_{A_i}, K_{A_i}\right\rangle\right\rangle$
Task	t_i	$W_i \mid T_i$ where $W_i = taskCall\left\langle\left\langle ID_{T_i}, K_{T_i}, ID_W, K_W, []\right\rangle\right\rangle$ and $T_i = task\left\langle\left\langle ID_{T_i}, K_{T_i}, M_{T_i}, [], []\right\rangle\right\rangle$
Task start	 t_s	$W_s.\overline{t_i} \mid t_i.W_i.\overline{t_{i+1}} \mid \ldots \mid T_s \mid T_i \mid \ldots$
Task end	 t_e	$\ldots \mid t_i.W_i.\overline{t_e} \mid t_e.W_e \mid T_i \mid T_e \mid \ldots$

6.10 Translation between BPAC and applied pi calculus

BPAC		Applied pi calculus
Component	Symbol/graphical representation	
Task sequence	 $t_i = t_{seq}$	$\ldots \mid t_{i-1}.W_{i-1}.\overline{t_i} \mid t_i.W_i.\overline{t_{i+1}} \mid t_{i+1}.W_{i+1}.\overline{t_{i+2}} \mid \ldots$ $\mid T_{i-1} \mid T_i \mid T_{i+1} \mid \ldots$
Conjunctive split	 $t_i = t_{\wedge s}$	$\ldots \mid t_i.W_i.\left(\overline{t_j} \mid \overline{t_k}\right) \mid t_j.W_j.\overline{t_{j+1}} \mid t_k.W_k.\overline{t_{k+1}} \mid \ldots$ $\mid T_i \mid T_j \mid T_k \mid \ldots$
Conjunctive join	 $t_k = t_{\wedge j}$	$\ldots \mid t_i.W_i.\overline{t_{ka}} \mid t_j.W_j.\overline{t_{kb}} \mid t_{ka}.t_{kb}.W_k.\overline{t_{k+1}} \mid \ldots$ $\mid T_i \mid T_j \mid T_k \mid \ldots$
Disjunctive split	 $t_i = t_{\oplus s}$	$\ldots \mid t_i.W_i.\left(\overline{t_j} + \overline{t_k}\right) \mid t_j.W_j.\overline{t_{j+1}} \mid t_k.W_k.\overline{t_{k+1}} \mid \ldots$ $\mid T_i \mid T_j \mid T_k \mid \ldots$
Disjunctive join	 $t_k = t_{\oplus j}$	$\ldots \mid t_i.W_i.\overline{t_k} \mid t_j.W_j.\overline{t_k} \mid t_k.W_k.\overline{t_{k+1}} \mid \ldots$ $\mid T_i \mid T_j \mid T_k \mid \ldots$
Store	$\mathcal{W}$	$Q, R, S, \ldots$ Separate stores maintained for e.g. agent/role assignment in applied pi.
Agent/role assignment	(a_i, r_j)	$role\left(ID_{A_i}, r_j\right)$

6.10 Translation between BPAC and applied pi calculus

BPAC		Applied pi calculus
Component	Symbol/graphical representation	
Task/role assignment	(t_i, r_j)	$test = \mathcal{T}(R)\left(role\left(ID_{A_k}, r_j\right)\right)$: in $W_i = taskCall\left\langle\left\langle ID_{T_i}, K_{T_i}, ID_W, K_W, [test]\right\rangle\right\rangle$ for task T_i
Primitive constraint predicate	$\varphi = pred(x_1, \ldots, x_n)$ e.g. $hasAccessed(a, Task_i, W)$	$dataFunction(x_1, \ldots, x_n)$ e.g. $hasAccessed\left(ID_A, ID_{T_i}, ID_W\right)$
Negative constraint predicate	$\neg\varphi = \neg pred(x_1, \ldots, x_n)$ e.g. $\neg hasAccessed(a, Task_i, W)$	see constraint test below
Constraint expression	$c = \varphi_0 \wedge \ldots \varphi_i \ldots \wedge \varphi_n$ $\wedge\neg\psi_0 \wedge \ldots \neg\psi_j \ldots \wedge \neg\psi_m$	see constraint test below
Constraint test	$(t_i, test(c, \mathcal{W}))$ e.g. $(Task_i, test(hasAccessed(a, Task_1, W)$ $\wedge\neg hasAccessed(a, Task_2, W)$, $\wedge\neg hasAccessed(a, Task_3, W), \mathcal{W}))$,	$test = \mathcal{T}(S)(dataFunction(x_1, \ldots, x_n)) : \ldots$ $T(S)\left(dataFunction\left(x'_1, \ldots, x'_n\right)\right) : \mathbf{0} : \ldots$ in process W_i for task T_i where $T(S)() : \mathbf{0}$: is test for negation and colon means proceed to the next test. e.g. $\mathcal{T}(S)\left(hasAccessed\left(id_A, ID_{Task_1}, ID_W\right)\right)$: $\mathcal{T}(S)\left(hasAccessed\left(id_A, ID_{Task_2}, ID_W\right)\right) : \mathbf{0}$: $\mathcal{T}(S)\left(hasAccessed\left(id_A, ID_{Task_3}, ID_W\right)\right) : \mathbf{0}$: in W_i for task T_i
Write command	$(t_i, write(c, \mathcal{W}))$ e.g. $(t_i, write(hasAccessed(a, Task_i, W), \mathcal{W}))$	$write = \mathcal{W}(S)(dataFunction(x_1, \ldots, x_n)) \ldots..$ $\mathcal{W}(S)\left(dataFunction\left(x'_1, \ldots, x'_n\right)\right)$ in W_i or in task process T_i e.g. $\mathcal{T}(S)\left(hasAccessed\left(id_A, ID_{Task_i}, ID_W\right)\right)$ in task T_i

BPAC		Applied pi calculus
Component	Symbol/graphical representation	
Delete command	$(t_i, delete(c, \mathcal{W}))$ e.g. $(t_i, delete(review(a,p), \mathcal{W}))$	$delete = \mathcal{D}(S)(dataFunction(x_1, \ldots, x_n)) \ldots..$ $\mathcal{D}(S)\left(dataFunction\left(x'_1, \ldots, x'_n\right)\right)$ in task process T_i e.g. $\mathcal{D}(S)(review(id_A, p))$ in task T_i
Policy rule	$(t_i, write(\{c_1, \ldots, c_n\}, \mathcal{W})), (t_i, test(c, \mathcal{W}))$, $(t_i, delete(\{d_1, \ldots, d_m\}, \mathcal{W})), (t_i, write(\{e_1, \ldots, e_p\}, \mathcal{W}))$ followed by an implicit $(t, write(hasAccessed(a, t, W), \mathcal{W}))$ in t_i	$write.test$ in hole in W_i, $delete$ in 1st hole in T_i and $write$ in 2nd hole in T_i followed by a pre-programmed $\mathcal{W}(S)\left(hasAccessed\left(id_a, ID_{T_i}, ID_W\right)\right)$ in T_i

6.11 The complete model

Drawing together all of the modelling components discussed above we can now specify the complete workflow access control model.

All workflows are constructed as parallel multiples of the core protocol, being encrypted interactions over a public channel between the workflow manager, agent and task processes. We generalise the workflow access control model as follows:

Firstly, we run the initialisation process $\mathcal{M}$, then we run any number of instances of the workflow-based access control model $\mathbf{P}$, say, defined as follows:

$$\mathbf{P} \triangleq W \mid \prod_i !agent\langle\langle ID_{A_i}, K_{A_i}\rangle\rangle \mid \prod_n !task\langle\langle ID_{T_n}, K_{T_n}, M_{T_n}, [\,], [\,]\rangle\rangle$$

for i number of agent processes and n number of task processes and where

$$W \triangleq \prod_m \nu ID_{T_m}.(Trig1_m.taskCall\langle\langle ID_{T_m}, K_{T_m}, ID_W, K_W\,[\,]\rangle\rangle.Trig2_m)$$

represents the workflow manager with $Trig1_m$ and $Trig2_m$ being Puhlmann and Weske workflow trigger

sub-processes as discussed in section 6.7.3.

All m instances of $taskCall\,\langle\langle ID_{T_m}, K_{T_m}, ID_W, K_W\,[]\rangle\rangle$ in the workflow management process W, where context [] is populated with access control tests or τ, interact with a task $task\,\langle\langle ID_{T_n}, K_{T_n}, M_{T_n}, [], []\rangle\rangle$, with contexts populated with store write and delete instructions or τ, and an agent $agent\,\langle\langle ID_{A_i}, K_{A_i}\rangle\rangle$. Interaction is in accordance with the standard core protocol, which we discuss in detail in appendix A.

6.12 Well-formed workflows

A workflow model constructed purely in accordance with the format of section 6.11 above, without any additional code, is secure from the perspective of an outside observer. That is, an outside observer is unable to distinguish between workflow transitions and random chatter, as all communication that comprises the standard core protocol is encrypted over the public channel and any additional communication, such as in respect of the stores, is private. We call such a model a well-formed workflow.

6.13 Summary

In this chapter we present the basic building blocks of our applied pi calculus implementation of BPAC. We discuss the core process components: agents, tasks and workflows and we display the syntax of these core components. Additionally, we present the initialisation process $\mathcal{M}$, which creates and defines private names, stores and support processes and populates the stores with access control information as appropriate. Also, we summarise the translation from the BPAC environment of chapter 3 to the applied pi calculus implementation presented in this chapter.

Based upon this modelling environment, in the next chapter, we revisit our simple workflow access control example of chapter 4 and in the following chapter we investigate security testing and analysis using the applied pi calculus workflow model.

Chapter 7

Modelling the simple workflow in BPAC

7.1 Introduction

In the previous chapter we outline the basic components of our applied pi calculus implementation of BPAC. In this chapter we use this modelling environment to encode the workflow example presented in chapter 4.

7.2 The model

7.2.1 Defining the initialisation process $\mathcal{M}$

Our first task is to initialise the names that we will be using in the model and to setup the stores and their support processes. Firstly, we need to initialise some of the private names that are used throughout the model space, such as key seeds and agent identities. We initialise a workflow key seed K_W, which enables public key encrypted communication with the workflow manager process, and an organisation key seed K_{Org}, which enables general communication between processes within the organisation. Key seeds K_{Task_x} and K_{Agent_y} are initialised for targeted communication with the task process $Task_x$ and the agent process $Agent_y$. Channel names set and $storeQ$, $storeR$ and $storeS$ are declared for store process management.

The role assignment store R needs to be populated with the agent/role assignments. We recall that in BPAC the agent/role assignments for our simple workflow are summarised by the following relation:

$$\begin{aligned} AR \;=\; \{&(Agent_1, employee)\,,\\ &(Agent_2, employee)\,,\ (Agent_2, clerk)\,,\\ &(Agent_3, employee)\,,\ (Agent_3, clerk)\,,\\ &(Agent_4, employee)\,,\ (Agent_4, clerk)\,,\\ &(Agent_4, supervisor)\,,\\ &(Agent_5, employee)\,,\ (Agent_5, supervisor)\} \end{aligned}$$

for a set of roles:

$$R \;=\; \{employee,\ clerk,\ supervisor\}$$

and set of agents:

$$A \;=\; \{Agent_1,\ Agent_2,\ Agent_3,\ Agent_4, Agent_5\}$$

We assign these roles as follows in our applied pi calculus model:

$$role\,(ID_{Agent_1}, employee)$$
$$role\,(ID_{Agent_2}, employee)$$
$$role\,(ID_{Agent_2}, clerk)$$
$$role\,(ID_{Agent_3}, employee)$$
$$role\,(ID_{Agent_3}, clerk)$$
$$role\,(ID_{Agent_4}, employee)$$
$$role\,(ID_{Agent_4}, clerk)$$
$$role\,(ID_{Agent_4}, supervisor)$$
$$role\,(ID_{Agent_5}, employee)$$
$$role\,(ID_{Agent_5}, supervisor)$$

Where $role\,(x, y)$ is a data function that matches y to x and ID_{Agent_z} is a private name that is used to represent a unique identifier for the agent, $Agent_z$. So $\mathcal{M}$ is defined as:

$$\begin{aligned}
\mathcal{M} \triangleq\ & \nu K_W, K_{Org}, K_{Task_1}, K_{Task_2}, K_{Task_3}, K_{Task_4}, K_{Task_5}, \\
& K_{Task_6}, ID_{Agent_1}, ID_{Agent_2}, ID_{Agent_3}, ID_{Agent_4}, ID_{Agent_5}, \\
& K_{Agent_1}, K_{Agent_2}, K_{Agent_3}, K_{Agent_4}, K_{Agent_5}, \\
& set, storeQ, storeR, storeS. \\
& ((\mathcal{W}\,(R)\,(role\,(ID_{Agent_1}, employee)) \\
& .\mathcal{W}\,(R)\,(role\,(ID_{Agent_2}, employee)) \\
& .\mathcal{W}\,(R)\,(role\,(ID_{Agent_2}, clerk)) \\
& .\mathcal{W}\,(R)\,(role\,(ID_{Agent_3}, employee)) \\
& .\mathcal{W}\,(R)\,(role\,(ID_{Agent_3}, clerk)) \\
& .\mathcal{W}\,(R)\,(role\,(ID_{Agent_4}, employee)) \\
& .\mathcal{W}\,(R)\,(role\,(ID_{Agent_4}, clerk)) \\
& .\mathcal{W}\,(R)\,(role\,(ID_{Agent_4}, supervisor)) \\
& .\mathcal{W}\,(R)\,(role\,(ID_{Agent_5}, employee)) \\
& .\mathcal{W}\,(R)\,(role\,(ID_{Agent_5}, supervisor))) \mid \chi)
\end{aligned} \tag{7.1}$$

Where $\mathcal{W}(R)(x)$ is our "write" process that sends x to our store R and χ represents the stores and their support processes.

7.2.2 Agent processes

The agent processes are represented in the applied pi calculus implementation of BPAC as:

$$\begin{aligned} Agent_1 &\triangleq !agent\langle\langle ID_{Agent_1}, K_{Agent_1}\rangle\rangle \\ Agent_2 &\triangleq !agent\langle\langle ID_{Agent_2}, K_{Agent_2}\rangle\rangle \\ Agent_3 &\triangleq !agent\langle\langle ID_{Agent_3}, K_{Agent_3}\rangle\rangle \\ Agent_4 &\triangleq !agent\langle\langle ID_{Agent_4}, K_{Agent_4}\rangle\rangle \\ Agent_5 &\triangleq !agent\langle\langle ID_{Agent_5}, K_{Agent_5}\rangle\rangle \end{aligned} \tag{7.2}$$

Each agent is represented as multiple instances (via the replication operator) of the standard agent process as detailed in appendix A section A.5.2, identified with the specific agent via the agent identifiers $ID_{Agent_1} \ldots ID_{Agent_5}$ and the agent key seeds $K_{Agent_1} \ldots K_{Agent_5}$, which ensure that communication is isolated to individual agents.

7.2.3 Task processes

The task processes are represented as:

$$\begin{aligned} Task_1 &\triangleq !task\langle\langle ID_{Task_1}, K_{Task_1}, M_{Task_1}, [\tau], [\tau]\rangle\rangle \\ Task_2 &\triangleq !task\langle\langle ID_{Task_2}, K_{Task_2}, M_{Task_2}, [\tau], [\tau]\rangle\rangle \\ Task_3 &\triangleq !task\langle\langle ID_{Task_3}, K_{Task_3}, M_{Task_3}, [\tau], [\tau]\rangle\rangle \\ Task_4 &\triangleq !task\langle\langle ID_{Task_4}, K_{Task_4}, M_{Task_4}, [\tau], [\tau]\rangle\rangle \\ Task_5 &\triangleq !task\langle\langle ID_{Task_5}, K_{Task_5}, M_{Task_5}, [\tau], [\tau]\rangle\rangle \\ Task_6 &\triangleq !task\langle\langle ID_{Task_6}, K_{Task_6}, M_{Task_6}, [\tau], [\tau]\rangle\rangle \end{aligned} \tag{7.3}$$

Each task is represented as multiple instances of the standard task process as detailed in appendix A section A.5.1. $M_{Task_1} \ldots M_{Task_6}$ are the messages that act as labels for task resources, $ID_{Task_1} \ldots ID_{Task_6}$

are fresh task identities created by the workflow manager and $K_{Task_1} \ldots K_{Task_6}$ are the key seeds associated with each task, which ensure that communication can be isolated to each task process.

7.2.4 Workflow manager

As discussed in chapter 6 the workflow manager mediates the communication between agents and tasks, checks agent credentials against stores in accordance with access control policy, permits or otherwise communication between agents and tasks and implements the overall workflow structure.

We define the workflow manager process W in our modelling environment as follows:

$$
\begin{aligned}
W \triangleq\ & \nu ID_W, ID_{Task_1}, ID_{Task_2}, ID_{Task_3}, ID_{Task_4}, ID_{Task_5}, \\
& ID_{Task_6}, t_2, t_3, t_4, t_{5a}, t_{5b}, t_6, t_{end}. \\
& (taskCall\,\langle\langle ID_{Task_1}, K_{Task_1}, ID_W, K_W, \\
& [\mathcal{T}\,(R)\,(role\,(id_A, employee)) :]\rangle\rangle\,.\overline{t_2} \\
& \mid t_2.taskCall\,\langle\langle ID_{Task_2}, K_{Task_2}, ID_W, K_W, \\
& [\mathcal{T}\,(R)\,(role\,(id_A, clerk)) : \\
& \mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W)) : \mathbf{0} :]\rangle\rangle \\
& .\left(\overline{t_3} \mid \overline{t_4}\right) \\
& \mid t_3.taskCall\,\langle\langle ID_{Task_3}, K_{Task_3}, ID_W, K_W, \\
& [\mathcal{T}\,(R)\,(role\,(id_A, supervisor)) : \\
& \mathcal{W}\,(S)\,(canAccess\,(id_A, ID_{Task_3}, ID_W))\,. \\
& \mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W)) : \mathbf{0} : \\
& \mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_2}, ID_W)) : \mathbf{0} : \\
& \mathcal{T}\,(S)\,(canAccess\,(id_A, ID_{Task_4}, ID_W)) : \mathbf{0} :]\rangle\rangle \\
& .\overline{t_{5a}} \\
& \mid t_4.taskCall\,\langle\langle ID_{Task_4}, K_{Task_4}, ID_W, K_W, \\
& [\mathcal{T}\,(R)\,(role\,(id_A, supervisor)) : \\
& \mathcal{W}\,(S)\,(canAccess\,(id_A, ID_{Task_4}, ID_W))\,. \\
& \mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W)) : \mathbf{0} :
\end{aligned}
\tag{7.4}
$$

$$
\begin{aligned}
&\mathcal{T}(S)(hasAccessed(id_A, ID_{Task_2}, ID_W)) : \mathbf{0} : \\
&\mathcal{T}(S)(canAccess(id_A, ID_{Task_3}, ID_W)) : \mathbf{0} :]\rangle\rangle \\
&.\overline{t_{5b}} \\
&\mid t_{5a}.t_{5b}.taskCall\,\langle\langle ID_{Task_5}, K_{Task_5}, ID_W, K_W, \\
&[\mathcal{T}(R)(role(id_A, clerk)) : \\
&\mathcal{T}(S)(hasAccessed(id_A, ID_{Task_1}, ID_W)) : \mathbf{0} : \\
&\mathcal{T}(S)(hasAccessed(id_A, ID_{Task_3}, ID_W)) : \mathbf{0} : \\
&\mathcal{T}(S)(hasAccessed(id_A, ID_{Task_4}, ID_W)) : \mathbf{0} :]\rangle\rangle \\
&.\overline{t_6} \\
&\mid t_6.taskCall\,\langle\langle ID_{Task_6}, K_{Task_6}, ID_W, K_W, \\
&[\mathcal{T}(R)(role(id_A, employee)) : \\
&\mathcal{T}(S)(hasAccessed(id_A, ID_{Task_1}, ID_W)) :]\rangle\rangle)
\end{aligned}
$$

that we sometimes abbreviate for analysis purposes to:

$$
\begin{aligned}
W \triangleq\ & \nu ID_W, ID_{Task_1}, ID_{Task_2}, ID_{Task_3}, \\
& ID_{Task_4}, ID_{Task_5}, ID_{Task_6}, \\
& t_2, t_3, t_4, t_{5a}, t_{5b}, t_6. \\
& \big(W_1.\overline{t_2} \mid t_2.W_2.\big(\overline{t_3} \mid \overline{t_4}\big) \\
& \mid t_3.W_3.\overline{t_{5a}} \mid t_4.W_4.\overline{t_{5b}} \\
& \mid t_{5a}.t_{5b}.W_5.\overline{t_6} \mid t_6.W_6\big)
\end{aligned}
\tag{7.5}
$$

and a terminating version of W, W_{term} is as follows:

$$
\begin{aligned}
W_{term} \triangleq\ & \nu ID_W, ID_{Task_1}, ID_{Task_2}, ID_{Task_3}, \\
& ID_{Task_4}, ID_{Task_5}, ID_{Task_6}, \\
& t_2, t_3, t_4, t_{5a}, t_{5b}, t_6, t_{end}. \\
& \left(W_1.\overline{t_2} \mid t_2.W_2.\left(\overline{t_3} \mid \overline{t_4}\right)\right. \\
& \mid t_3.W_3.\overline{t_{5a}} \mid t_4.W_4.\overline{t_{5b}} \\
& \mid t_{5a}.t_{5b}.W_5.\overline{t_6} \mid t_6.W_6.\overline{t_{end}} \\
& \left. \mid t_{end}.\bar{t}\langle M_{end}\rangle\right)
\end{aligned} \tag{7.6}
$$

where the special terminating process $\bar{t}\langle M_{end}\rangle$ is added to the workflow, triggered by $\overline{t_{end}}$.

7.2.5 The complete workflow model

We specify a complete, well-formed model of the workflow $\mathbf{P_{purchase}}$, which incorporates the workflow manager W detailed above, agent and task processes, as follows:

$$
\begin{aligned}
\mathbf{P_{purchase}} \triangleq\ & W \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid Agent_1 \mid Agent_2 \mid Agent_3 \\
& \mid Agent_4 \mid Agent_5
\end{aligned} \tag{7.7}
$$

and the terminating model of the workflow is as follows:

$$
\begin{aligned}
\mathbf{P_{termp}} \triangleq\ & W_{term} \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid Agent_1 \mid Agent_2 \mid Agent_3 \\
& \mid Agent_4 \mid Agent_5
\end{aligned} \tag{7.8}
$$

7.3 Narrative explanation of the workflow model

Process $\mathcal{M}$ initialises identity values and key seeds (respectively ID_x and K_y where x and y are workflow/agent/task identifiers). Process $\mathbf{P_{purchase}}$ then proceeds. Workflow manager process W commences in parallel with multiple copies of processes for each agent and each task involved in the workflow. Process W proceeds as follows:

$$taskCall\,\langle\langle ID_{Task_1}, K_{Task_1}, ID_W, K_W, [\mathcal{T}\,(R)\,(role\,(id_A, employee))\,:]\rangle\rangle$$

initialises the task process for task $Task_1$, advertises task $Task_1$ to the agent pool representing organisation employees then, on communication of an agent with the workflow manager, tests the substituted agent identity variable id_A against store R to verify or otherwise whether the agent has the role *employee*. This test represents the standard role-based access control verification process. If the employee id_A is matched to the role *employee* using the data function $role\,(id_A, employee)$ within the store R, the process continues, otherwise the process halts. Following the test, the process initiates secure communication between the agent ID_A and the task $Task_1$ and the task process writes to store S the data function $hasAccessed\,(id_A, ID_{Task_1}, ID_W)$ to record that the agent has interacted with the task.

On completion of the task-call process the workflow manager triggers the next task-call process by sending a trigger on private channel t_2.

On receipt of the trigger on t_2 workflow manager W proceeds to:

$$\begin{aligned}&taskCall\,\langle\langle ID_{Task_2}, K_{Task_2}, ID_W, K_W,\\&[\mathcal{T}\,(R)\,(role\,(id_A, clerk))\,:\\&\ \mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W))\,:\mathbf{0}\,:]\rangle\rangle\end{aligned}$$

in which the task-call process initialises task $Task_2$ and advertises the task to the agent pool. When an agent communicates with the workflow manager W the task-call process applies the following tests on the agent's identity: first, the agent identity is tested against store R to match the agent to role *clerk* by $\mathcal{T}\,(R)\,(role\,(id_A, clerk))\,:$. This test represents the standard role-based access control verification process as before. If the match succeeds, the agent is tested against S by $\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W))\,:$ $\mathbf{0}\,:$. This test represents the dynamic access control verification process that ensures that an agent cannot

access task $Task_2$ if it has already accessed task $Task_1$. If the test against store S succeeds, the task-call process halts, otherwise it initiates communication between the agent and the task. The task process $Task_2$ writes $hasAccessed\,(ID_A, ID_{Task_2}, ID_W)$ to the store S indicating that the agent has accessed task $Task_2$ then the workflow process triggers a pair of task-call processes using parallel private trigger channels t_3 and t_4.

On receipt of the trigger on t_3 workflow manager W proceeds to:

$$
\begin{aligned}
&taskCall\,\langle\langle ID_{Task_3}, K_{Task_3}, ID_W, K_W,\\
&[\mathcal{T}\,(R)\,(role\,(id_A, supervisor)) :\\
&\mathcal{W}\,(S)\,(canAccess\,(id_A, ID_{Task_3}, ID_W))\,.\\
&\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W)) : \mathbf{0} :\\
&\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_2}, ID_W)) : \mathbf{0} :\\
&\mathcal{T}\,(S)\,(canAccess\,(id_A, ID_{Task_4}, ID_W)) : \mathbf{0} :]\rangle\rangle
\end{aligned}
$$

that continues as before but with task $Task_3$ initialised and advertised to the agent pool. An agent accesses the workflow process in response to the advertisement and the agent is tested against the stores using the following access control tests: firstly, the agent identity variable id_A is tested against store R to match the agent to role *supervisor* using data function $role\,()$, i.e. $\mathcal{T}\,(R)\,(role\,(id_A, supervisor)) :$. If there is a match to the store, the process first writes data function $canAccess\,(id_A, ID_{Task_3}, ID_W)$ to the store S and then tests id_A against a collection of dynamic access control rules written to store S that ensures that the agent has not accessed tasks $Task_1$, $Task_2$ and is not accessing $Task_4$. If test matches are achieved, the process halts, otherwise the workflow manager instigates communication between the agent and task $Task_3$. On completion of agent/task access the task process $Task_3$ writes that the agent identity substituted for id_A has accessed task $Task_3$ to store S and then the trigger t_{5a} is fired by the task-call process.

Concurrent with the processes outlined in the previous paragraph the workflow manager W, on receipt of the trigger t_4, continues as follows:

$$
\begin{aligned}
&taskCall\,\langle\langle ID_{Task_4}, K_{Task_4}, ID_W, K_W,\\
&[\mathcal{T}\,(R)\,(role\,(id_A, supervisor)):\\
&\mathcal{W}\,(S)\,(canAccess\,(id_A, ID_{Task_4}, ID_W))\,.\\
&\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W)):\mathbf{0}:\\
&\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_2}, ID_W)):\mathbf{0}:\\
&\mathcal{T}\,(S)\,(canAccess\,(id_A, ID_{Task_3}, ID_W)):\mathbf{0}:]\rangle\rangle
\end{aligned}
$$

This sequence proceeds in a similar manner to the previously discussed sequence, testing the agent for the *supervisor* role, writing $canAccess\,(id_A, ID_{Task_3}, ID_W)$ to the store S and checking that the agent has not accessed tasks $Task_1$, $Task_2$ and is not accessing $Task_3$. On completion of agent/task, access the task process $Task_4$ writes that the agent substituted for id_A has accessed task $Task_4$ to store S and then the trigger t_{5b} is fired by the task-call process.

The next task-call process sequence is triggered by the sequential receipt of triggers t_{5a} and t_{5b} so that task $Task_5$ can only be completed after both $Task_3$ and $Task_4$ have been completed. The workflow manager W proceeds as follows:

$$
\begin{aligned}
&|\,t_{5a}.t_{5b}.taskCall\,\langle\langle ID_{Task_5}, K_{Task_5}, ID_W, K_W,\\
&[\mathcal{T}\,(R)\,(role\,(id_A, clerk)):\\
&\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W)):\mathbf{0}:\\
&\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_3}, ID_W)):\mathbf{0}:\\
&\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_4}, ID_W)):\mathbf{0}:]\rangle\rangle
\end{aligned}
$$

In this process sequence the agent is tested for the role *clerk* and access to tasks $Task_1$, $Task_3$ and $Task_4$. Note that in this particular case our access-control policies allow for the agent permitted to access task $Task_5$ to be the same agent who accessed task $Task_2$ but this is not a necessary condition for access to task $Task_5$. On completion of task $Task_5$ the trigger t_6 is fired.

On receipt of trigger t_6 the workflow manager proceeds as follows:

$$taskCall\,\langle\langle ID_{Task_6}, K_{Task_6}, ID_W, K_W,$$
$$[\mathcal{T}\,(R)\,(role\,(id_A, employee)):$$
$$\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W)):]\rangle\rangle$$

In this process segment the agent is checked for role *employee* and is then matched to the agent who accessed task $Task_1$ via the test

$$\mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{Task_1}, ID_W)):$$

The dynamic access control requirement in this case is that only the agent who has accessed task $Task_1$ can access task $Task_6$.

When the terminating workflow W_{end} is substituted for W in $\mathbf{P_{purchase}}$ then the modified $\mathbf{P_{termp}}$ proceeds in the same manner as $\mathbf{P_{purchase}}$ except that on completion of task $Task_6$ the $\overline{t_{end}}$ trigger fires and the resulting terminating process sends message M_{end} over the public channel to indicate that the workflow process has reached its end.

7.4 Comments on the workflow model

The simple purchases workflow demonstrates encoding under the applied pi calculus implementation of BPAC. It is apparent from this example that our model environment is powerful and flexible and that segregation of duties, as specified in the access control policy, can be applied via interaction with the stores. However, it is also apparent that this power is realised at the expense of a degree of complexity of encoding and that complex models can become long and unwieldy. To counter this, it is anticipated that ultimately the BPAC modelling environment will be automated and potentially incorporated within a BPM program suite with a graphical interface so that the details of the code are not exposed to the user.

7.5 Summary

In the section above we present the access control model representation of our simple workflow-based access control example and outline how it properly encodes the access control constraints. In the next chapter we present security analysis with the applied pi calculus implementation of BPAC.

Chapter 8

Security analysis within BPAC

8.1 Introduction

In the previous chapter we present a simple workflow example encoded using the applied pi calculus implementation of BPAC. Now we turn our attention to security testing within the modelling environment. In the next section we discuss the security analysis requirements for access control models and this is followed by a selection of security tests. Tests such as satisfiability and reachability (sections 8.2.6 and 8.2.4) require the "lightweight" analysis tool of the abstraction test as introduced in section 8.2.3 below. Tests of anonymity as outlined in section 8.2.7 below require the full power of observational equivalence/labelled bisimilarity as per sections 5.3.4 and 5.3.6. In section 8.3 we present examples of the application of these security tests.

8.2 Security analysis for access control models

8.2.1 The security model

Within our modelling environment all interactions between agent, workflow manager and task processes are encrypted and executed over a public channel. Interactions between processes and the stores and between the stores and set processes are ordinarily carried out over private channels restricted over the scope of the model. In the applied pi calculus an attacker is represented by an arbitrary context that can read from or write to the public (unbound) channels. Ostensibly, there is nothing of interest in a well-formed workflow model as all that an attacker can see is random chatter over a public channel. However, we can engineer the selective leaking of information into the public domain from the model in a

variety of different ways and this enables us to analyse a vast collection of "what-if" scenarios for a given workflow model.

Given the analytical power and flexibility of the applied pi calculus when coupled with the attacker model outlined above, there is an enormous range of security analysis available to us within our BPAC implementation. We devise a suite of security tests in the applied pi calculus based upon the properties of BPAC defined in section 3.2.1 including:

- satisfiability analysis — satisfiability is defined in section 3.4.3 and is achieved if a workflow-based access control is observed to be terminable given some set of agents, tasks and authorisation constraints. Within the applied pi calculus implementation of BPAC we perform tests for satisfiability of workflow models using the abstraction test of section 8.2.3 below.

- Collusion analysis — we define collusion and the collusion metric in section 3.4.4. The collusion metric is the minimum number of agents who can collectively secure complete control over resources associated with a workflow. We perform collusion analysis in the applied pi calculus as an extension of satisfiability tests. The collusion metric is calculated for a workflow model from multiple satisfiability experiments over different subsets of agents.

- Agent/task reachability analysis — agent/task reachability refers to the workflow property that task resources can be accessed by an agent given that resources from some other workflow task can also be accessed by the agent. We discuss testing for agent/task reachability in further detail in section 8.2.6. Within a conference management example in section 8.3.2, we perform a variation on satisfiability analysis as an agent/task reachability test to establish whether or not a conference paper reviewer can gain sight of another review of the same paper prior to the execution of the review task on that particular paper.

- Agent anonymity and collusion resistance — anonymity is introduced in section 3.4.5. In real-world examples of systems of access control, protection against collusion is provided by non-disclosure of the identities of participating agents. In a conference review system, for example, if a reviewer does not know the identity of her fellow reviewers for a given paper, privacy-based access control protection is afforded against collusion of participating parties. Testing for privacy can be achieved by proving the observational equivalence of processes with interchanging participating agents as we outline in section 8.2.7 below and demonstrate in an example in section 8.3.1.4.

8.2.2 Uniqueness property for public channel messages within the modelling environment

Our modelling environment is constructed from an initialisation process and agent, task and workflow manager processes that interact with each other and with support processes, the stores, in a predefined way. This interaction of processes, a collection of labelled and internal transitions called the standard core protocol C, defined in section 6.4, provides the building blocks from which model examples of workflow-based access control can be constructed within our modelling environment. A prerequisite for the definition of our suite of analytical tools, which we can use to investigate our workflow models, is that no two labelled transitions within a well-formed model are the same. We refer to this particular property as uniqueness and this is presented in the next section. Now, at first sight, uniqueness would appear to be a trivial property. However, given the underlying complexity of the standard core protocol, there is always a possibility that duplicate messages could be sent over multiple instances of the protocol within an access control model. If the uniqueness property does not hold for any model, we cannot assume that encrypted private messages broadcast over the public channel are indistinguishable by an outside observer from fresh messages broadcast over the public channel.

8.2.2.1 The uniqueness property

Let $\mathbf{P}$ *be any secure well-formed workflow model as per section 6.12 comprising a number of linked cases of standard core processes,* $C_1, \ldots, C_m$ *as defined in section 6.4, initialised by process* $\mathcal{M}$ *as defined in section 6.9.4, such that* $\mathbf{P}$ *can undergo labelled transitions of the form* $\mathbf{P} \xrightarrow{\alpha^1} \ldots \xrightarrow{\alpha^n} \mathbf{P'}$ *then:*

1. $\alpha^i \neq \alpha^j$ *for* $i, j = 1$ *to* n *and* $i \neq j$
2. *For any two instances of* $\mathbf{P}$, $\mathbf{P^1}$ *and* $\mathbf{P^2}$, *say, initialised by process* $\mathcal{M}$, *such that* $\mathbf{P^1} \xrightarrow{\alpha_1^1} \ldots \xrightarrow{\alpha_1^n} \mathbf{P'^1}$ *and* $\mathbf{P^2} \xrightarrow{\alpha_2^1} \ldots \xrightarrow{\alpha_2^n} \mathbf{P'^2}$ *then* $\alpha_1^k \neq \alpha_2^l$ *for* $k, l = 1$ *to* n.

8.2.2.2 Outline of proof

The full details of the proof are presented in section B.1 of appendix B. In this section we outline the basic components of the proof.

1. All well-formed workflow models are constructed from the standard core protocol. Therefore we first compare all public transitions against each other within the standard protocol C and prove that no two public transitions are the same.

2. Next, we consider comparison of all public transitions between any two cases of the standard core protocol within some well-formed workflow model **P**. We need to consider how complex workflows are constructed from the standard core protocol, given that the standard core protocol is effectively the interaction of the processes *taskCall* (actually sub-processes within the workflow, W process), *task* and *agent*. If we consider any two standard protocol cases within the well-formed workflow model **P**: C_1 and C_2, say, initialised respectively by fresh identities ID_{T_a} and ID_{T_b} then:

   (a) all public communications are unique within C_1 as per 1 above.

   (b) All public communications are unique within C_2 as per 1 above.

   (c) All public communications within C_1, when compared to all similar public communications within C_2 are unique because any communication within C_1 contains a freshly declared task identity name ID_{T_a} and any communication within C_2 contains a freshly declared task identity name ID_{T_b} and as all functions $enc()$, $encs()$ and $sk()$ are collision-free then these communications can never be equal.

   (d) All public communications within C_1, when compared to all dissimilar public communications within C_2 are unique, the proof being similar to 1 above except that comparison of public transitions is performed across both cases of the core protocol.

3. Finally, we consider two instances $\mathbf{P_1}$ and $\mathbf{P_2}$, initialised by process $\mathcal{M}$, of the well-formed workflow model **P** and we argue by extension of 1 and 2 above and the freshness of task identities ID_T per each instance of the standard core protocol that all public transitions are unique within this comparison.

8.2.2.3 Application of the uniqueness property to workflow analysis.

The uniqueness property says that public messages generated by a well-formed workflow model are indistinguishable from random chatter by an outside observer. This is a powerful feature of the model that enables us to simplify our security tests.

If it was always the case that our workflow models were well-formed, these models would be totally uninteresting: a well-formed workflow is a black-box process that tells us nothing. However, our agent processes have been designed so that they can be adapted to leak acquired messages unencrypted into the public domain. This provides us with the ability to analyse all manner of "what-if" scenarios in respect of agents securing control over task resources via the various security tests on workflow models. The basic analytical component of our security tests is the abstraction test defined below.

8.2.3 The abstraction test

In chapter 5 we present an important analytical tool for the applied pi calculus called observational equivalence and its practical alternative technique called labelled bisimilarity. Proving observational equivalence is a two-way process such that for $A \approx B$ we prove that, for some external observer, there exists a B that satisfies the equivalence rules in respect of A then vice versa for A in respect of B. Within the context of our modelling environment, some of our security tests for any given access control protocol are unidirectional and existential, i.e. we need to identify or otherwise that there exists any example of a model trace that behaves like some abstract process trace. The burden of proving full observational equivalence is excessive and superfluous for these unidirectional security tests. Consequently, we use a lightweight comparison test in one direction only. We attempt to demonstrate that, for some external observer, there exists or otherwise a trace for some process A that is indistinguishable from the trace for some test process B. We call this test the abstraction test and we use the test for the reachability class of security problems for our access control models. If we can prove that a process **P** can behave like an abstract model **Q**, we say that **P** abstracts to **Q** and we write this as:

$$\mathbf{P} \quad \sim_{ab} \quad \mathbf{Q}$$

The abstraction test can be considered to be an applied pi calculus equivalent of the refinement test of CSP [80]. In refinement a comparison is made between two model descriptions, the specification and the implementation. If a check is successful then the specification is considered to be a refinement of the implementation. In the abstraction test the abstract reference model **Q** is analogous to the specification and the workflow model **P** is analogous to the implementation.

The formal definition of the abstraction test is as follows:

Definition 20

Abstraction $(\sim_{ab})$ *is a relation* $\mathcal{R}$ *on closed unique extended processes such that* $A \,\mathcal{R}\, B$ *implies:*

1. $A \approx_s B$;
2. *if* $B \longrightarrow^* B'$ *then* $\exists A'$ *s.t.* $A \longrightarrow^* A'$ *and* $A' \,\mathcal{R}\, B'$;
3. *if* $B \longrightarrow^* \xrightarrow{\alpha} \longrightarrow^* B'$, $fv(\alpha) \subseteq dom(B)$ *and* $bn(\alpha) \cap fn(A) = \varnothing$, *then* $\exists A'$ *s.t.* $A \longrightarrow^* \xrightarrow{\alpha} \longrightarrow^* A'$ *and* $A' \,\mathcal{R}\, B'$

The definition bears a likeness to the definition of labelled bisimilarity in chapter 5 except that it is unidirectional i.e. a similarity. We can further simplify this test by the application of the uniqueness principle in respect of the set of labelled transitions associated with the standard core protocol as outlined below.

8.2.3.1 Practical application of the abstraction test

We devise a simple strategy for using the abstraction test as part of our security analysis. Given some workflow model $\mathbf{P}$, initialised by $\mathcal{M}$, then we create a modified variant of $\mathbf{P}$ called $\mathbf{P'}$ that releases task messages of the form M_T to an attacker via the agent processes or via a special terminating process $t_{end}.\overline{t}\langle M_{end}\rangle$. We set up an abstract reference model $\mathbf{Q}$ that releases task messages also and we endeavour to find a trace of $\mathbf{P'}$ that is indistinguishable from the trace of $\mathbf{Q}$. If such a trace is found, we conclude that:

$$\mathbf{P'} \quad \sim_{ab} \quad \mathbf{Q}$$

We simplify the test by the application of the uniqueness theorem and syntactic sugar for the transition traces of the abstract reference model $\mathbf{Q}$ and the workflow variant $\mathbf{P'}$.

8.2.3.2 Defining the abstract reference model Q

The abstract reference model $\mathbf{Q}$ is the benchmark against which the workflow model is analysed and comprises sequential outputs of fresh names over the public channel together with the output of selected messages that represents the anticipated behaviour of the workflow model $\mathbf{P}$ or its derivatives. A typical example of reference model $\mathbf{Q}$ for the test of satisfiability would be formulated as:

$$\nu n_1.\overline{t}\langle n_1\rangle.\ldots.\nu n_i.\overline{t}\langle n_i\rangle.\overline{t}\langle M_{end}\rangle \tag{8.1}$$

for some i such that the output trace for $\nu n_1.\overline{t}\langle n_1\rangle.\ldots.\nu n_i.\overline{t}\langle n_i\rangle$ matches the output trace for the encrypted components of model $\mathbf{P}$ or its terminating derivative $\mathbf{P_{term}}$ and M_{end} is the message released on the public channel as part of the workflow terminating process within $\mathbf{P_{term}}$. The precise form of $\mathbf{Q}$ is not important to us provided that the trace is of the form:

$$\xrightarrow{\nu n_1.\overline{t}\langle n_1\rangle} \ldots \xrightarrow{\nu n_i.\overline{t}\langle n_i\rangle}\overline{t}\langle M_{end}\rangle \xrightarrow{} \tag{8.2}$$

as this is what we use as part of our analysis. We are able to use this in our abstraction test because the uniqueness property says that the public messages in a well-formed workflow model are indistinguishable from random chatter. More complex examples of **Q** can be devised that enable detailed analysis of workflow models.

8.2.3.3 Syntactic sugar for the transition trace for the abstract reference model

Sequential fresh messages are included in **Q** to represent all encrypted messages on the public channel in **P** and its derivatives that cannot be decrypted by an outside observer. In order to simplify our understanding of the transition steps for reference model **Q** we use the syntax $\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$ to represent a finite number of $\overset{\nu n.\bar{t}\langle n\rangle}{\longrightarrow}$ transitions. We are not concerned with knowing the exact number of $\overset{\nu n.\bar{t}\langle n\rangle}{\longrightarrow}$ steps in $\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$ for our experiments provided that the steps are matched on both sides of the comparison. We can do this because we are free in theory to devise **Q** with any finite number of sequential fresh messages and we can always therefore incorporate the exact number of fresh messages that match the private messages in **P**. These messages, like internal transactions, have no impact on our abstraction test other than to act as placeholders that match the private messages in the target model **P**. Also, we know from our uniqueness property that, so long as our model is properly constructed from the standard core protocol C and assuming that no cryptographic secrets are exposed to an outside observer, all of the core protocol transitions of the workflow model **P** and its derivatives are unique. In practice, whenever we devise **Q** for our experiments we simply assume that an undisclosed quantity of fresh messages precedes each instance of a public test message as in formula 8.1 above.

For instance, for some initialised model $\mathbf{P_{term}}$ derived from **P** by incorporating a terminating process of the form: $\mid t_{end}.\bar{t}\,\langle M_{end}\rangle$ within the workflow manager, a test for satisfiability against abstract reference model **Q** summarised as formula 8.1 above is as follows:

if a trace of $\mathbf{P_{term}}$ matches:

$$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$$

then

$$\mathbf{P_{term}} \sim_{ab} \mathbf{Q}$$

and $\mathbf{P_{term}}$ is satisfiable.

Alternatively, a trace of a simple serial workflow model of four tasks might be compared to:

$$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_1\rangle}{\longrightarrow}\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_2\rangle}{\longrightarrow}\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_3\rangle}{\longrightarrow}\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_4\rangle}{\longrightarrow} \tag{8.3}$$

that is: we attempt to find a trace of our workflow model that is indistinguishable from the trace 8.3 above.

A workflow model with parallel tasks T_2 and T_3 might be tested to see if it yields a trace comparable to:

$$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_1\rangle}{\longrightarrow}\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_2\rangle}{\longrightarrow}\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_3\rangle}{\longrightarrow}\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_4\rangle}{\longrightarrow} \tag{8.4}$$

and:

$$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_1\rangle}{\longrightarrow}\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_3\rangle}{\longrightarrow}\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_2\rangle}{\longrightarrow}\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_4\rangle}{\longrightarrow} \tag{8.5}$$

The test is successful only if we can demonstrate that the model can undergo transitions indistinguishable from both of the traces shown above. We abbreviate traces 8.4 and 8.5 as:

$$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_1\rangle}{\longrightarrow}\ \left(\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_2\rangle}{\longrightarrow}\ \wedge\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_3\rangle}{\longrightarrow}\right)\ \overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_4\rangle}{\longrightarrow} \tag{8.6}$$

using the $\wedge$ symbol to represent a conjunction of process traces i.e. both traces $\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_2\rangle}{\longrightarrow}$ and $\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_3\rangle}{\longrightarrow}$ have to occur prior to trace $\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\ \overset{\bar{t}\langle M_4\rangle}{\longrightarrow}$ occurring.

Alternatively, we might want to test our model to see if it performs like one trace or another. For example, given a four task workflow with parallel tasks T_2 and T_3 then it might be sufficient to establish that the model behaves like either of traces 8.4 or 8.5. We abbreviate the traces in this instance as follows using the disjunction operator $\vee$:

$$\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow} \stackrel{\bar{t}\langle M_1\rangle}{\longrightarrow} \left(\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow} \stackrel{\bar{t}\langle M_2\rangle}{\longrightarrow} \vee \stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow} \stackrel{\bar{t}\langle M_3\rangle}{\longrightarrow} \right) \stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow} \stackrel{\bar{t}\langle M_4\rangle}{\longrightarrow} \tag{8.7}$$

It is worth noting that two sequences of $\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$, i.e. $\stackrel{\nu n.\bar{t}\langle n\rangle\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$ can be summarised as $\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$ as two finite sequences of $\stackrel{\nu n.\bar{t}\langle n\rangle}{\longrightarrow}$ are behaviourally the same as one larger finite sequence of $\stackrel{\nu n.\bar{t}\langle n\rangle}{\longrightarrow}$ but we cannot arbitrarily deconstruct $\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$ into $\stackrel{\nu n.\bar{t}\langle n\rangle\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$ as $\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$ has to represent at least one instance of $\stackrel{\nu n.\bar{t}\langle n\rangle}{\longrightarrow}$ per our definition above.

8.2.3.4 Syntactic sugar for the transition trace for the standard core protocol

In order to simplify our analysis of the protocol model itself we use the following as syntactic sugar to represent a summarised transition trace for the standard core protocol C where C_i is the i^{th} instance of C, effectively representing all of the unique encrypted transitions that occur between the workflow sub-process W_q, the task process T_r and the agent process A_s :

$$W_q \stackrel{C_i}{\Leftrightarrow} T_r, A_s$$

This represents all of the labelled transitions for the core process. For a well-formed workflow these labelled transitions are indistinguishable from the same number of messages of the form:

$$\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$$

by the uniqueness property.

8.2.4 Satisfiability tests

We use the abstraction test to establish that an initialised workflow-based access control model **P** is satisfiable as defined in section 3.4.3.

Firstly, we make sure that our workflow model includes a terminating process that broadcasts message M_{end} on the public channel t. If not, then we extend the workflow manager model to create amended

model $\mathbf{P_{term}}$ by adding the following process to the end of the model, triggered by the Puhlmann and Weske workflow triggers as usual:

$$task_{end} \triangleq \overline{t}\langle M_{end}\rangle$$

i.e. message M_{end} can be output on the public channel only if the interaction of agents, tasks and the workflow manager results in the complete performance of the workflow as modelled by W.

Now we define a reference model $\mathbf{Q}$ in respect of its summarised labelled transitions that we specify as:

$$\stackrel{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}\stackrel{\overline{t}\langle M_{end}\rangle}{\longrightarrow}$$

where message M_{end} is output on the public channel and all other outputs on public channels are a finite number of fresh messages that precede it. If there exists $\mathbf{P_{term}}$ that is observed to undergo transitions that are indistinguishable from $\mathbf{Q}$, i.e.:

$$\mathbf{P_{term}} \sim_{ab} \mathbf{Q}$$

then it can be concluded that $\mathbf{P_{term}}$ is satisfiable and therefore the workflow-based access control policy is satisfiable for a given set of agents and tasks.

8.2.5 Collusion analysis and calculating the collusion metric

If we have established that the workflow-based access control policy is satisfiable by the application of satisfiability tests as per section 8.2.4, we know that there is a sufficient number of agent processes within the initialised terminated access control model $\mathbf{P_{term}}$ to ensure that the workflow can be completed. We define the simple collusion metric in section 3.4.4 as the minimum number of agents that can obtain complete control of all resources associated with a workflow. We can identify the simple collusion metric by repeatedly performing the satisfiability test over modifications to $\mathbf{P_{term}}$ containing subsets of the set of agent processes and by counting the minimum number of agent processes required to ensure satisfiability of the workflow model.

The simple collusion metric is a reasonable attempt at quantifying the robustness of a workflow against collusion. However, we can envisage a situation where agents are able to secure information about task resources indirectly, via intra-agent communication for example. In this case the simple collusion metric as defined might not be sufficient to properly represent the collusion resistance of a complete workflow-based access control model. Consequently, we define the extended collusion metric as being the minimum number of agents that can obtain complete control of all resources associated with a workflow by both direct and indirect means. In order to calculate the extended collusion metric we perform the following steps:

1. firstly, we define an abstract reference model $\mathbf{Q_c}$ by reference to its transition sequence:
   (a) we represent each and every task $task_i$ within the workflow model by an output transition of the task message on the public channel t, i.e. $\stackrel{\overline{t}\langle M_i\rangle}{\longrightarrow}$ for the i^{th} task and:
   (b) we precede each output transition $\stackrel{\overline{t}\langle M_i\rangle}{\longrightarrow}$ by a finite number of transition steps given by $\stackrel{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$ and:
   (c) we sequence the labelled transitions $\stackrel{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}\stackrel{\overline{t}\langle M_i\rangle}{\longrightarrow}$ so as to replicate the anticipated output from our workflow model so that $\mathbf{Q_c}$ is some process that undergoes a transition trace of the form:

$$\stackrel{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}\stackrel{\overline{t}\langle M_1\rangle}{\longrightarrow}\stackrel{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}\stackrel{\overline{t}\langle M_2\rangle}{\longrightarrow}\stackrel{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}\stackrel{\overline{t}\langle M_3\rangle}{\longrightarrow}\ldots$$

2. Next we attach a public output of messages to a selection of agent processes within $\mathbf{P_{term}}$ using a context of the form:

$$[\,] \mid !\left(t\left(x_{A_i}\right).\mathbf{let}\ m_{A_i} = fst\left(dec\left(x_{A_i}, sk\left(K_{A_i}\right)\right)\right)\ \mathbf{in}\ \overline{t}\left\langle m_{A_i}\right\rangle\right)$$

   for the i^{th} agent process within $\mathbf{P_{term}}$. This context receives message m_{A_i}, which is broadcast by the standard agent process encrypted under agent A_i's public key, then it outputs the decryption of this message on the public channel t.

3. Now we take the modified model from step 2 above, $\mathbf{P_{term}}'$, say, and we perform the abstraction test to ascertain whether or not:

$$\mathbf{P'_{term}} \quad \sim_{ab} \quad \mathbf{Q_c}$$

If this is satisfied, i.e. there exists a trace for $\mathbf{P'_{term}}$ that is indistinguishable from the trace of $\mathbf{Q_c}$, we have obtained a collusion set of agent processes for $\mathbf{P_{term}}$, being all of the agents that were modified to release their messages onto the public channel t.

4. We continue to perform steps 2 and 3 for different combinations of agent processes leaking messages and we obtain a minimum collusion set of agent processes for which:

$$\mathbf{P'_{term}} \quad \sim_{ab} \quad \mathbf{Q_c}$$

is satisfied. The number of agents within this set is the extended collusion metric for the workflow-based access control model $\mathbf{P_{term}}$.

8.2.6 Testing for reachability using satisfiability

The tests outlined above for satisfiability and the computation of the collusion metric are examples of tests for workflow reachability. In our test for satisfiability we focus upon the reachability problem: can an end state for a workflow model be reached given some initial state? In the collusion analysis tests we ask a similar question but we test for a more comprehensive set of model properties given a number of different initial states. A more general test for reachability is to identify one particular state of a workflow model and observe if some other state can be reached thereafter. Of particular interest in this regard is that tests for reachability can be devised that seek to ascertain whether or not at some particular state in a workflow model run an agent has accessed a particular task. In this context we consider reachability testing as a variation of satisfiability testing in which we attempt to identify whether or not specific task/agent assignments can be identified within a satisfiability experiment. Such a test is particularly useful, for example, for testing whether or not a reviewer has had access to a review of a paper prior to her reviewing that paper in a conference management system.

8.2.7 Observational equivalence tests for anonymity

Anonymity is defined in section 3.4.5 as the property that an outsider observer is unable to match agents to tasks or workflows within a workflow-based access control model and we test for anonymity using labelled bisimilarity of section 5.3.6.

We consider a workflow model $\mathbf{P}$ and the uniqueness theorem as discussed above that ensures that all private transitions are fresh. If we model a compromised agent A_i in $\mathbf{P}$ as the release of agent secrets, e.g.

the agent's identity ID_{A_i} and key seed K_{A_i}, to an attacker $Att[\,]$, anonymity is preserved if an attacker is unable to match agents to workflow tasks. We test for anonymity using labelled bisimilarity as follows:

for some target agent process A_j where $A_j \neq A_i$, if we define a modified workflow process $\mathbf{P''}$ as the workflow model process $\mathbf{P}$ less the agent processes A_i and A_j (we do not need to include compromised agent A_i as the attacker can fully replicate A_i's communication), anonymity is preserved if:

$$A_j \quad | \quad \mathbf{P''} \approx \mathbf{P''}$$

and therefore an attacker, via some compromised agent, cannot identify which agent performed some particular task within a workflow. All the attacker knows by reachability analysis is that particular tasks may have been performed.

Anonymity is breached in the following circumstance in accordance with our anonymity definition of section 3.4.5:

- If an agent A_j is critical to the workflow process then if compromised agent A_i interacts with some task dependent upon agent A_j's interaction with the workflow process, the omission of process A_j from the workflow model will result in compromised agent A_i being unable to complete its task and $A_j \mid \mathbf{P''} \not\approx \mathbf{P''}$.

This is the primary example of breach of anonymity for the purpose of this thesis, consistent with our BPAC model of chapter 3. It is observed that the concept of anonymity can be extended in the real world to encompass the effect of communication between agents. For example, a compromised agent tells an untrusted third party the identity of an agent accessing a particular task or task secrets are released to an untrusted third party through communicating agents. However, such extensions to the definition of anonymity are outside the scope of this thesis.

8.2.7.1 Syntactic sugar for traces of the standard core protocol

In section 8.2.3.4 we introduce the following syntax to represent a complete transition trace of the standard core protocol:

$$W_q \overset{C_i}{\Leftrightarrow} T_r, A_s$$

Whilst we use this notation for transitions when outputs are indistinguishable from fresh names over the public channel, we need to adopt a modified version of this trace if messages $mess_1, \ldots, mess_n$ are revealed on the public channel via the equational theory:

$$\left\{W_q \overset{C_i}{\Leftrightarrow} T_r, A_s\right\} \wr mess_1, \ldots, mess_n$$

We use this syntax to represent all traces for a single instance C_i of the standard core protocol that do not involve a compromised agent yet an attacker has access to the messages $mess_1, \ldots, mess_n$.

In respect of a compromised agent we expand the trace syntax so that the following represents a complete trace of the standard core protocol in which agent process A_s has been compromised by an attacker (indicated by $Att\,(A_s)$) and messages $mess_1, \ldots, mess_n$ have been released to the attacker:

$$\left\{W_q \overset{C}{\Leftrightarrow} T_r, Att\,(A_s)\right\} \wr mess_1, \ldots, mess_n$$

8.3 Some Examples of security tests

In this section we apply the security tests discussed above to BPAC examples that we model in the applied pi calculus. In the first example we return to the simple, two-task workflow that we introduce in chapter 3 and for the second example we analyse a flawed conference review model originally presented by Nan Zhang [104].

8.3.1 The two-task workflow example

Consider the model workflow W of section 3.3.12 comprising two sequential task processes, T_1 and T_2 and three agent processes, A_1, A_2 and A_3 and we assign roles to the agents using our model syntax as follows:

$role\,(ID_{A_1}, clerk)$

$role\,(ID_{A_2}, supervisor)$

$role\,(ID_{A_3}, manager)$

Firstly, we define our initialisation process, $\mathcal{M}$:

$$
\begin{aligned}
&\nu K_W, K_{Org}, K_{T_1}, K_{T_2}, ID_{A_1}, ID_{A_2}, ID_{A_3}, \\
&K_{A_1}, K_{A_2}, K_{A_3}, set, storeQ, storeR, storeS. \\
&(\mathcal{W}(R)(role(ID_{A_1}, clerk)) \\
&.\mathcal{W}(R)(role(ID_{A_2}, supervisor)) \\
&.\mathcal{W}(R)(role(ID_{A_3}, manager)) \mid \chi)
\end{aligned}
\tag{8.8}
$$

where $\mathcal{X}$ represents all of the private internal support processes, such as the set handling processes and the store processes Q, R and S. The $\mathcal{W}(R)$ processes write the agent/role assignments to the store R.

The agent processes are represented as:

$$
\begin{aligned}
A_1 &\triangleq !agent\langle\langle ID_{A_1}, K_{A_1}\rangle\rangle \\
A_2 &\triangleq !agent\langle\langle ID_{A_2}, K_{A_2}\rangle\rangle \\
A_3 &\triangleq !agent\langle\langle ID_{A_3}, K_{A_3}\rangle\rangle
\end{aligned}
$$

The task processes as:

$$
\begin{aligned}
T_1 &\triangleq !task\langle\langle ID_{T_1}, K_{T_1}, M_{T_1}, [\tau], [\tau]\rangle\rangle \\
T_2 &\triangleq !task\langle\langle ID_{T_2}, K_{T_2}, M_{T_2}, [\tau], [\tau]\rangle\rangle
\end{aligned}
$$

and the workflow manager process as:

$$
\begin{aligned}
W_{term} \triangleq\ & \nu t_2, t_{end}, ID_W, ID_{T_1}, ID_{T_2}. \\
& (taskCall\,\langle\langle ID_{T_1}, K_{T_1}, ID_W, K_W, \\
& [\mathcal{T}\,(R)\,(role\,(id_A, clerk))\,:]\rangle\rangle\,.\overline{t_2} \\
& \mid t_2.taskCall\,\langle\langle ID_{T_2}, K_{T_2}, ID_W, K_W, \\
& [\mathcal{T}\,(R)\,(role\,(id_A, supervisor))\,: \\
& \mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{T_1}, ID_W))\,:\mathbf{0}\,:]\rangle\rangle \\
& .\overline{t_{end}} \mid t_{end}.\bar{t}\,\langle M_{end}\rangle\big)
\end{aligned}
$$

that, for the purpose of our subsequent analysis, we abbreviate as follows:

$$
\begin{aligned}
W_{term} \triangleq\ & \nu t_2, t_{end}, ID_W, ID_{T_1}, ID_{T_2}. \\
& \big(\big(W_1.\overline{t_2}\big) \mid \big(t_2.W_2.\overline{t_{end}}\big) \mid \big(t_{end}.\bar{t}\,\langle M_{end}\rangle\big)\big)
\end{aligned}
\tag{8.9}
$$

where

$$
\begin{aligned}
W_1 \triangleq\ & taskCall\,\langle\langle ID_{T_1}, K_{T_1}, ID_W, K_W, \\
& [\mathcal{T}\,(R)\,(role\,(id_A, clerk))\,:]\rangle\rangle
\end{aligned}
\tag{8.10}
$$

and

$$
\begin{aligned}
W_2 \triangleq\ & taskCall\,\langle\langle ID_{T_2}, K_{T_2}, ID_W, K_W, \\
& [\mathcal{T}\,(R)\,(role\,(id_A, supervisor))\,: \\
& \mathcal{T}\,(S)\,(hasAccessed\,(id_A, ID_{T_1}, ID_W))\,:\mathbf{0}\,:]\rangle\rangle
\end{aligned}
\tag{8.11}
$$

and the basic model is then represented as:

$$
\mathbf{P_{term}} \triangleq W_{term} \mid T_1 \mid T_2 \mid A_1 \mid A_2 \mid A_3 \tag{8.12}
$$

Note that our model $\mathbf{P_{term}}$ contains a terminating process attached within W_{term} of the form:

$$[\,] \mid t_{end}.\bar{t}\,\langle M_{end}\rangle \tag{8.13}$$

where t_{end} is a trigger and M_{end} is the usual termination message to indicate that the workflow has been completed.

8.3.1.1 Testing the two-task workflow example for satisfiability

Given the two-task workflow-based access control model, initialised by process $\mathcal{M}$, detailed above we seek to establish that, given the access control constraints and given the available agents A_1, A_2 and A_3, a trace can be found in which the complete workflow can be executed.

Procedure

The procedure for this test is based upon the idea that we attempt to find a trace of our model $\mathbf{P_{term}}$ that behaves like a target model $\mathbf{Q}$. We use the procedure that we describe as the abstraction test as outlined in section 8.2.3 above.

We formulate the labelled transition behaviour of our test process $\mathbf{Q}$ as follows:

$$\stackrel{\nu n.\bar{t}\langle n\rangle \bar{t}\langle M_{end}\rangle}{\Longrightarrow \longrightarrow}$$

and we perform the abstraction test to establish or otherwise:

$$\mathbf{P_{term}} \quad \sim_{ab} \quad \mathbf{Q}$$

That is we seek to establish whether or not there exists a trace of $\mathbf{P_{term}}$ that is indistinguishable from the trace of $\mathbf{Q}$ in so far as an external observer is concerned.

Firstly, we rewrite $\mathbf{P_{term}}$ as follows:

$$\begin{aligned}\mathbf{P_{term}} &\triangleq W_{term} \mid T_1 \mid T_2 \mid A_1 \mid A_2 \mid A_3 \\ &= \nu t_2, ID_W, ID_{T_1}, ID_{T_2}. \\ &\quad \left(\left(W_1.\overline{t_2}\right) \mid \left(t_2.W_2.\overline{t_{end}}\right) \mid \left(t_{end}.\bar{t}\,\langle M_{end}\rangle\right)\right) \\ &\quad \mid T_1 \mid T_2 \mid A_1 \mid A_2 \mid A_3\end{aligned}$$

by substitution using 8.9 above.

By rewriting $\mathbf{P_{term}}$ in this way we indicate that $\mathbf{P_{term}}$ can be considered as a succession of core protocols. In this case there are two core protocols: the interaction between W_1 and either T_1 or T_2 and A_1 or A_2 or A_3, which we can refer to as core protocol instance C_1 and the interaction between W_2 and either T_1 or T_2 and A_1 or A_2 or A_3, which we can refer to as core protocol instance C_2.

Now we attempt to find a trace of $\mathbf{P_{term}}$ that is indistinguishable from the trace for $\mathbf{Q}$ (summarised below):

No.	$\mathbf{P_{term}}$		$\mathbf{Q}$
1	$W_1 \overset{C_1}{\Leftrightarrow} T_1, A_1$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
2	$W_2 \overset{C_2}{\Leftrightarrow} T_2, A_2$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
3	$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$		$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$
	success		

therefore

$$\mathbf{P_{term}} \sim_{ab} \mathbf{Q}$$

and there exists a trace wherein workflow sub-process W_1 interacts with task process T_1 and agent A_1 (core protocol instance C_1) and workflow sub-process W_2 interacts with task process T_2 and agent A_2 (core protocol instance C_2) followed by the terminating process $\bar{t}\,\langle M_{end}\rangle$ and this trace is similar to the trace of the abstract reference model $\mathbf{Q}$. We can conclude, therefore, that the workflow-based access control model is satisfiable using the specified access control constraints and given the set of agents $\{A_1, A_2, A_3\}$ and the set of tasks $\{T_1, T_2\}$.

8.3.1.2 The simple collusion metric

As we discuss in section 8.2.4, we can extend the satisfiability test described above to create a test that identifies the minimum number of agents required to ensure model satisfiability. That is, we perform the satisfiability test for various combinations of agents and we identify the smallest number of agents, the simple collusion metric, for which the satisfiability test succeeds. For our toy example above we perform the full test for satisfiability first, i.e. we test to establish or otherwise:

$$\begin{array}{rcl} \mathbf{P_{term}} & \triangleq & W_{term} \mid T_1 \mid T_2 \mid A_1 \mid A_2 \mid A_3 \\ \sim_{ab} & & \mathbf{Q} \end{array}$$

Where $\mathbf{Q}$ has the transition trace $\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$ as before. We know this to be true from section 8.3.1.1 above. Now we remove agent process A_1 and we test to establish or otherwise:

$$\begin{array}{rcl} \mathbf{P'_{term}} & \triangleq & W_{term} \mid T_1 \mid T_2 \mid A_2 \mid A_3 \\ \sim_{ab} & & \mathbf{Q} \end{array}$$

with $\mathbf{Q}$ as before and the resultant trace comparisons are:

No.	$\mathbf{P'_{term}}$		$\mathbf{Q}$
1	$W_1 \stackrel{C_1}{\Leftrightarrow} T_1, A_2$		$\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
2	fail		

and

No.	$\mathbf{P'_{term}}$		$\mathbf{Q}$
1	$W_1 \stackrel{C_1}{\Leftrightarrow} T_1, A_3$		$\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
2	fail		

therefore we conclude that

$$\mathbf{P'_{term}} \not\sim_{ab} \mathbf{Q}$$

as the terminating process $\bar{t}\langle M_{end}\rangle$ is never reached in the workflow model and the model is not satisfiable using the specified access control constraints and given the set of agents $\{A_2, A_3\}$ and the set of tasks $\{T_1, T_2\}$. Indeed, process $\mathbf{P'}$ can never proceed beyond step 1 above as neither A_2 nor A_3 are permitted to access task T_1 by the role assignments and access control rules set out in the example and T_1 will never be completed.

Repeating the test for:

$$\mathbf{P''_{term}} \triangleq W_{term} \mid T_1 \mid T_2 \mid A_1 \mid A_3$$

yields the same result in that $\mathbf{P''_{term}}$ cannot output M_{end} over the public channel although in this case we can obtain a trace in which T_1 is completed through interaction with agent A_1. However, if we run the test for:

$$\mathbf{P'''_{term}} \triangleq W_{term} \mid T_1 \mid T_2 \mid A_1 \mid A_2$$

then we obtain the trace comparison:

No.	$\mathbf{P'''_{term}}$		$\mathbf{Q}$
1	$W_1 \overset{C_1}{\Leftrightarrow} T_1, A_1$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
2	$W_2 \overset{C_2}{\Leftrightarrow} T_2, A_2$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
3	$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$		$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$
4	success		

and

$$\mathbf{P'''_{term}} \sim_{ab} \mathbf{Q}$$

therefore, the workflow-based access control model is satisfiable using the specified access control constraints and given the set of agents $\{A_1, A_2\}$ and the set of tasks $\{T_1, T_2\}$.

Now, given that

$$\mathbf{P'_{term}} \triangleq W_{term} \mid T_1 \mid T_2 \mid A_2 \mid A_3$$

and

$$\mathbf{P''_{term}} \triangleq W_{term} \mid T_1 \mid T_2 \mid A_1 \mid A_3$$

both failed the satisfiability tests we can say without further analysis that:

$$\mathbf{P^4_{term}} \triangleq W_{term} \mid T_1 \mid T_2 \mid A_1$$
$$\mathbf{P^5_{term}} \triangleq W_{term} \mid T_1 \mid T_2 \mid A_2$$
$$\mathbf{P^6_{term}} \triangleq W_{term} \mid T_1 \mid T_2 \mid A_3$$

also fail the satisfiability tests.

We can conclude, therefore, that the minimum number of required agents to ensure satisfiability of our workflow example, the simple collusion metric, is two and indeed the minimum set of agents required to ensure satisfiability is $\{A_1, A_2\}$.

8.3.1.3 Extending the collusion metric analysis

We have calculated the simple collusion metric for our toy workflow example above. The result is a fair representation of the resistance to collusion for the model, assuming that control of task resources cannot pass to agents by other means, such as intra-agent communication. If we have a situation where agents can communicate with each other then we need to consider the extended collusion metric per 8.2.5. We can model all manner of interactions between agents and tasks in addition to the workflow-based access control protocols using the applied pi calculus. For example, consider an adaptation of the toy example presented above:

agent A_1 communicates any messages it obtains via its interactions with workflow/task processes to agent A_2 via encrypted communication over the public channel. We can model this as follows, noting that the

agent process outputs its acquired messages, encrypted under its own public key, on the public channel t as part of the detailed encoding of agent processes per section A.5.2 in appendix A:

$$\begin{aligned}\mathbf{P^8_{term}} \triangleq\ & W_{term} \mid T_1 \mid T_2 \mid (A_1 \mid !\,(t\,(x_{A_1})\,.\mathbf{let}\ m_{A_1} = fst\,(dec\,(x_{A_1}, sk\,(K_{A_1})))\\ & \mathbf{in}\ \nu n_{nonce}.\overline{t}\,\langle enc\,((m_{A_1}, n_{nonce})\,, pub_{A_2})\rangle)\big)\\ & \mid (A_2 \mid !t\,(x_{A_2})) \mid A_3\end{aligned}$$

Wherein workflow process W_{term} is as specified in formula 8.9 above and the fresh nonce n_{nonce} is included to maintain the uniqueness property of public transitions. If we perform the simple collusion metric test again, we yield the same result as before: the minimum number of required agents to ensure satisfiability of our workflow example is two and the minimum set of agents required to ensure satisfiability is still $\{A_1, A_2\}$. However, because of the communication between agents A_1 and A_2 then agent A_2 has control of both tasks (as task messages M_{T_1} and M_{T_2}) and we say that the extended collusion metric is one in this example. Note that two agents are still required to perform the workflow but one agent can have complete control over the workflow.

In practice, analysing a model for the extended collusion metric requires multiple tests. First we test by assigning output messages to all of the agent processes, then we selectively apply outputs to agent processes and we check at each stage that a complete set of messages can be output as specified by a model $\mathbf{Q'_c}$ that behaves as follows:

$$\stackrel{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}\ \stackrel{\overline{t}\langle M_1\rangle}{\longrightarrow}\ \stackrel{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}\ \stackrel{\overline{t}\langle M_2\rangle}{\longrightarrow}\ \stackrel{\overline{t}\langle M_{end}\rangle}{\longrightarrow} \tag{8.14}$$

Experiment 1: agents A_1, A_2 and A_3 leak secrets

Our first experiment is performed on:

$$\begin{aligned}\mathbf{P^8_{term}} \triangleq\ & W_{term} \mid T_1 \mid T_2 \mid (A_1 \mid !\,(t\,(x_{A_1})\,.\mathbf{let}\ m_{A_1} = fst\,(dec\,(x_{A_1}, sk\,(K_{A_1})))\\ & \mathbf{in}\ \nu n_{nonce}.\overline{t}\,\langle enc\,((m_{A_1}, n_{nonce})\,, pub_{A_2})\rangle\,.\overline{t}\,\langle m_{A_1}\rangle)\big)\\ & \mid \big(A_2 \mid !\,\big(t\,(x_{A_2})\,.\mathbf{let}\ m_{A_2} = fst\,(dec\,(x_{A_2}, sk\,(K_{A_2})))\ \mathbf{in}\ \overline{t}\,\langle m_{A_2}\rangle\big)\big)\\ & \mid \big(A_3 \mid !\,\big(t\,(x_{A_3})\,.\mathbf{let}\ m_{A_3} = fst\,(dec\,(x_{A_3}, sk\,(K_{A_3})))\ \mathbf{in}\ \overline{t}\,\langle m_{A_3}\rangle\big)\big)\end{aligned}$$

being the usual workflow and task processes together with agents A_1, A_2 and A_3 all outputting their messages on the public channel t and agent A_1 communicating with agent A_2. Performing the analysis yields the following traces:

No.	$\mathbf{P^8_{term}}$		$\mathbf{Q'_c}$
1	$W_1 \overset{C_1}{\Leftrightarrow} T_1, A_1$		$\nu n.\overline{t}\langle n\rangle$ $\Longrightarrow$
2	$(t(x_{A_1}).\nu n_{nonce}.\overline{t}\langle enc((m_{A_1}, n_{nonce}), pub_{A_2})\rangle.\overline{t}\langle m_{A_1}\rangle)$ $\longrightarrow$		$\nu n.\overline{t}\langle n\rangle\overline{t}\langle M_1\rangle$ $\Longrightarrow$ $\longrightarrow$
3	$W_2 \overset{C_2}{\Leftrightarrow} T_2, A_2$		$\nu n.\overline{t}\langle n\rangle$ $\Longrightarrow$
4	$(t(x_{A_2}).\overline{t}\langle m_{A_2}\rangle)$ $\longrightarrow$		$\overline{t}\langle M_2\rangle$ $\longrightarrow$
5	$\overline{t}\langle M_{end}\rangle$ $\longrightarrow$		$\overline{t}\langle M_{end}\rangle$ $\longrightarrow$
	success		

therefore

$$\mathbf{P^8_{term}} \sim_{ab} \mathbf{Q'_c}$$

and the set of agents $\{A_1, A_2, A_3\}$ has access to all of the secrets that comprise the complete workflow. Note that, given the manner in which information is leaked from agents then the order in which the test sequence 8.14 is specified is irrelevant as the leaky agents in $\mathbf{P^8_{term}}$ can output secrets in any order.

Experiment 2: agents A_2 and A_3 leak secrets

Next, we perform the same test and we restrict outputs from the agent processes as follows:

$$\begin{aligned}\mathbf{P^9_{term}} \triangleq\ & W_{term} \mid T_1 \mid T_2 \mid (A_1 \mid !(t(x_{A_1}).\mathbf{let}\ m_{A_1} = fst(dec(x_{A_1}, sk(K_{A_1})))\\ & \mathbf{in}\ \nu n_{nonce}.\overline{t}\langle enc((m_{A_1}, n_{nonce}), pub_{A_2})\rangle))\\ & \mid (A_2 \mid !(t(x_{A_2}).\mathbf{let}\ m_{A_2} = fst(dec(x_{A_2}, sk(K_{A_2})))\ \mathbf{in}\ \overline{t}\langle m_{A_2}\rangle))\\ & \mid (A_3 \mid !(t(x_{A_3}).\mathbf{let}\ m_{A_3} = fst(dec(x_{A_3}, sk(K_{A_3})))\ \mathbf{in}\ \overline{t}\langle m_{A_3}\rangle))\end{aligned}$$

that is we no longer output secrets from agent A_1 although A_1 still communicates with A_2. The trace results are as follows:

No.	$\mathbf{P^9_{term}}$		$\mathbf{Q'_c}$
1	$W_1 \overset{C_1}{\Leftrightarrow} T_1, A_1$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
2	$\overset{t(x_{A_1}).\nu n_{nonce}.\bar{t}\langle enc((m_{A_1},n_{nonce}),pub_{A_2})\rangle}{\longrightarrow}$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
3	$\overset{t(x_{A_2}).\bar{t}\langle m_{A_2}\rangle}{\longrightarrow}$		$\overset{\bar{t}\langle M_{T_1}\rangle}{\longrightarrow}$
4	$W_2 \overset{C_2}{\Leftrightarrow} T_2, A_2$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
5	$\overset{t(x_{A_2}).\bar{t}\langle m_{A_2}\rangle}{\longrightarrow}$		$\overset{\bar{t}\langle M_{T_2}\rangle}{\longrightarrow}$
6	$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$		$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$
	success		

therefore

$$\mathbf{P^9_{term}} \sim_{ab} \mathbf{Q'_c}$$

and the set of agents $\{A_2, A_3\}$ has access to all of the secrets that comprise the complete workflow.

Experiment 3: agents A_1 and A_3 leak secrets

Next, we perform the test again but we restrict outputs from the agent processes as follows:

$$\begin{aligned}\mathbf{P^{10}_{term}} \triangleq\ & W_{term} \mid T_1 \mid T_2 \mid (A_1 \mid !(t(x_{A_1}).\mathbf{let}\ m_{A_1} = fst(dec(x_{A_1}, sk(K_{A_1})))\\ & \mathbf{in}\ \nu n_{nonce}.\bar{t}\langle enc((m_{A_1}, n_{nonce}), pub_{A_2})\rangle.\bar{t}\langle m_{A_1}\rangle)) \mid (A_2 \mid !t(x_{A_2}))\\ & \mid (A_3 \mid !(t(x_{A_3}).\mathbf{let}\ m_{A_3} = fst(dec(x_{A_3}, sk(K_{A_3})))\ \mathbf{in}\ \bar{t}\langle m_{A_3}\rangle))\end{aligned}$$

that is we no longer output secrets from agent A_2. The trace results are as follows:

No.	$\mathbf{P^{10}_{term}}$		$\mathbf{Q'_c}$
1	$W_1 \overset{C_1}{\Leftrightarrow} T_1, A_1$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
2	$\overset{t(x_{A_1}).\nu n_{nonce}.\bar{t}\langle enc((m_{A_1},n_{nonce}),pub_{A_2})\rangle.\bar{t}\langle m_{A_1}\rangle}{\longrightarrow}$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\overset{\bar{t}\langle M_{T_1}\rangle}{\longrightarrow}$
3	$W_2 \overset{C_2}{\Leftrightarrow} T_2, A_2$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
4	$\overset{t(x_{A_2})\mid\bar{t}\langle m_{A_3}\rangle}{\longrightarrow}$		fail

therefore we conclude that

$$\mathbf{P}^{\mathbf{10}}_{\mathbf{term}} \not\sim_{ab} \mathbf{Q}'_{\mathbf{c}}$$

and the agent set $\{A_1, A_3\}$ does not have access to all of the secrets that comprise the complete workflow.

Experiment 4: agents A_1 and A_2 leak secrets

We perform the test again but we restrict outputs from the agent processes as follows:

$$\begin{aligned}\mathbf{P}^{\mathbf{11}}_{\mathbf{term}} \triangleq\ & W_{term} \mid T_1 \mid T_2 \mid (A_1 \mid !\,(t\,(x_{A_1})\,.\mathbf{let}\ m_{A_1} = fst\,(dec\,(x_{A_1}, sk\,(K_{A_1})))\\ & \mathbf{in}\ \nu n_{nonce}.\bar{t}\,\langle enc\,((m_{A_1}, n_{nonce})\,, pub_{A_2})\rangle\,.\bar{t}\,\langle m_{A_1}\rangle\big)\big)\\ & \mid \big(A_2 \mid !\,\big(t\,(x_{A_2})\,.\mathbf{let}\ m_{A_2} = fst\,(dec\,(x_{A_2}, sk\,(K_{A_2})))\ \mathbf{in}\ \bar{t}\,\langle m_{A_2}\rangle\big)\big)\\ & \mid (A_3 \mid !t\,(x_{A_3}))\end{aligned}$$

that is we no longer output secrets from agent A_3. The trace results are as follows:

No.	$\mathbf{P}^{\mathbf{11}}_{\mathbf{term}}$		$\mathbf{Q}'_{\mathbf{c}}$
1	$W_1 \overset{C_1}{\Leftrightarrow} T_1, A_1$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
2	$\overset{t(x_{A_1}).\nu n_{nonce}.\bar{t}\langle enc((m_{A_1}, n_{nonce}), pub_{A_2})\rangle.\bar{t}\langle m_{A_1}\rangle}{\longrightarrow}$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\overset{\bar{t}\langle M_{T_1}\rangle}{\longrightarrow}$
3	$W_2 \overset{C_2}{\Leftrightarrow} T_2, A_2$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
4	$\overset{t(x_{A_2}).\bar{t}\langle m_{A_2}\rangle}{\longrightarrow}$		$\overset{\bar{t}\langle M_{T_2}\rangle}{\longrightarrow}$
5	$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$		$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$
	success		

therefore

$$\mathbf{P}^{\mathbf{11}}_{\mathbf{term}} \sim_{ab} \mathbf{Q}'_{\mathbf{c}}$$

and the agent set $\{A_1, A_2\}$ has access to all of the secrets that comprise the complete workflow.

Experiment 5: agent A_1 leaks secrets

Having restricted the output of secrets to any two agents, we perform the tests again only this time we restrict the output of secrets to single agents. Firstly, we restrict output to agent A_1 only:

$$\begin{aligned}\mathbf{P}^{\mathbf{12}}_{\mathbf{term}} \triangleq\ & W_{term} \mid T_1 \mid T_2 \mid (A_1 \mid !\,(t\,(x_{A_1})\,.\mathbf{let}\ m_{A_1} = fst\,(dec\,(x_{A_1}, sk\,(K_{A_1})))\\ & \mathbf{in}\ \nu n_{nonce}.\overline{t}\,\langle enc\,((m_{A_1}, n_{nonce})\,, pub_{A_2})\rangle\,.\overline{t}\,\langle m_{A_1}\rangle\big)\big)\\ & \mid (A_2 \mid !t\,(x_{A_2})) \mid (A_3 \mid !t\,(x_{A_3}))\end{aligned}$$

yielding the following trace results:

No.	$\mathbf{P}^{\mathbf{12}}_{\mathbf{term}}$		$\mathbf{Q}'_{\mathbf{c}}$
1	$W_1 \overset{C_1}{\Leftrightarrow} T_1, A_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
2	$t(x_{A_1}).\nu n_{nonce}\overline{t}\langle enc(\underset{\longrightarrow}{(m_{A_1}, n_{nonce})}, pub_{A_2})\rangle.\overline{t}\langle m_{A_1}\rangle$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}\overset{\overline{t}\langle M_{T_1}\rangle}{\longrightarrow}$
3	$W_2 \overset{C_2}{\Leftrightarrow} T_2, A_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
4	fail		

therefore we conclude that

$$\mathbf{P}^{\mathbf{12}}_{\mathbf{term}} \not\approx_{ab} \mathbf{Q}'_{\mathbf{c}}$$

and agent A_1 alone does not have access to all of the secrets that comprise the complete workflow.

Experiment 6: agent A_2 leaks secrets

Next, we restrict output to agent A_2 only:

$$\begin{aligned}\mathbf{P}^{\mathbf{13}}_{\mathbf{term}} \triangleq\ & W_{term} \mid T_1 \mid T_2 \mid (A_1 \mid !\,(t\,(x_{A_1})\,.\mathbf{let}\ m_{A_1} = fst\,(dec\,(x_{A_1}, sk\,(K_{A_1})))\\ & \mathbf{in}\ \nu n_{nonce}.\overline{t}\,\langle enc\,((m_{A_1}, n_{nonce})\,, pub_{A_2})\rangle\big)\big)\\ & \mid \big(A_2 \mid !\,\big(t\,(x_{A_2})\,.\mathbf{let}\ m_{A_2} = fst\,(dec\,(x_{A_2}, sk\,(K_{A_2})))\ \mathbf{in}\ \overline{t}\,\langle m_{A_2}\rangle\big)\big)\\ & \mid (A_3 \mid !t\,(x_{A_3}))\end{aligned}$$

with the following trace results:

No.	$\mathbf{P^{13}_{term}}$		$\mathbf{Q'_c}$
1	$W_1 \overset{C_1}{\Leftrightarrow} T_1, A_1$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
2	$\overset{t(x_{A_1}).\nu n_{nonce}.\bar{t}\langle enc((m_{A_1},n_{nonce}),pub_{A_2})\rangle}{\longrightarrow}$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
3	$\overset{t(x_{A_2}).\bar{t}\langle m_{A_2}\rangle}{\longrightarrow}$		$\overset{\bar{t}\langle M_{T_1}\rangle}{\longrightarrow}$
4	$W_2 \overset{C_2}{\Leftrightarrow} T_2, A_2$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
4	$\overset{t(x_{A_2}).\bar{t}\langle m_{A_2}\rangle}{\longrightarrow}$		$\overset{.\bar{t}\langle M_{T_2}\rangle}{\longrightarrow}$
5	$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$		$\overset{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$
	success		

therefore

$$\mathbf{P^{13}_{term}} \sim_{ab} \mathbf{Q'_c}$$

and agent A_2 alone has access to all of the secrets that comprise the complete workflow.

Experiment 7: agent A_3 leaks secrets

Finally, we restrict output to agent A_3 only:

$$\begin{aligned}\mathbf{P^{14}_{term}} \triangleq\ & W_{term} \mid T_1 \mid T_2 \mid (A_1 \mid !\,(t\,(x_{A_1})\,.\mathbf{let}\ m_{A_1} = fst\,(dec\,(x_{A_1}, sk\,(K_{A_1})))\\ & \mathbf{in}\ \nu n_{nonce}.\bar{t}\,\langle enc\,((m_{A_1}, n_{nonce})\,, pub_{A_2})\rangle)\big)\\ & \mid (A_2 \mid !t\,(x_{A_2})) \mid (A_3 \mid !\,(t\,(x_{A_3})\,.\mathbf{let}\ m_{A_3} = fst\,(dec\,(x_{A_3}, sk\,(K_{A_3})))\\ & \mathbf{in}\ \bar{t}\,\langle m_{A_3}\rangle)\big)\end{aligned}$$

with the following trace results:

No.	$\mathbf{P^{14}_{term}}$		$\mathbf{Q'_c}$
1	$W_1 \overset{C_1}{\Leftrightarrow} T_1, A_1$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
2	$\overset{t(x_{A_1}).\nu n_{nonce}.\bar{t}\langle enc((m_{A_1},n_{nonce}),pub_{A_2})\rangle}{\longrightarrow}$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
3	$W_2 \overset{C_2}{\Leftrightarrow} T_2, A_2$		$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}$
4	$\overset{t(x_{A_2})}{\longrightarrow}$		fail

therefore we conclude that

$$\mathbf{P}^{\mathbf{14}}_{\mathbf{term}} \quad \not\sim_{ab} \quad \mathbf{Q}'_{\mathbf{c}}$$

and agent A_3 alone does not have access to all of the secrets that comprise the complete workflow.

Conclusions

Given all of the above tests, we can conclude that the minimum number of agents that can have access to all of the workflow messages by direct and indirect means and can therefore exercise complete control over the workflow is one: agent A_2 has access to all of the messages. We conclude, therefore, that the example workflow has an extended collusion metric of one.

Clearly, in this toy example the calculation of the extended collusion metric does not yield results that are terribly surprising. However, in a more complex workflow model with interactions between certain agents (representing siblings, close friends or spouses perhaps) then the extended collusion metric can behave as an indicator as to the success or otherwise of the application of segregation of duties to workflow-based agent/task assignments.

8.3.1.4 Performing tests for anonymity

We return once again to our two-task workflow example of section 8.3.1 and this time we analyse the model for anonymity. For the purpose of this set of experiments we assume that one agent, A_2, has been compromised so that an attacker, $att[]$, has control over A_2's secrets: K_{A_2}, the key seed, an organisation-wide key seed K_{org} and the agent's identity ID_{A_2}. The results of these experiments are displayed in summary form using the syntax of section 8.2.7.1. Details of the experiment traces are included in appendix C.

Experiment 8

We perform the equivalence test to ascertain whether or not:

$$\mathbf{P}^{\mathbf{15}}_{\mathbf{term}} = W_{term} \mid T_1 \mid T_2 \mid A_1 \quad \approx \quad \mathbf{P}^{\mathbf{16}}_{\mathbf{term}} = W_{term} \mid T_1 \mid T_2 \mid A_3$$

that is we test the model with agent A_1 and the attacker behaving as agent A_2 against the model with agent A_3 and the attacker behaving as agent A_2.

The comparative reductions for $\mathbf{P^{15}_{term}}$ and $\mathbf{P^{16}_{term}}$ are summarised as follows (details in appendix C section C.1):

No.	$\mathbf{P^{15}_{term}}$	$\mathbf{P^{16}_{term}}$
1	$\left\{W_{term} \overset{C_1}{\Leftrightarrow} T_1, A_1\right\} \wr\, ID_{T_1}, pub_{T_1}$	$\left\{W_{term} \overset{C_1}{\Leftrightarrow} T_1, A_3\right\} \wr\, ID_{T_1}, pub_{T_1}$ FAIL

In this instance it is clear that labelled reduction of $W_{term} \mid T_1 \mid T_2 \mid A_3$ cannot proceed beyond T_1 as neither the attacker nor agent process A_3 can access T_1 due to access control constraints and

$$\mathbf{P^{15}_{term}} = W_{term} \mid T_1 \mid T_2 \mid A_1 \quad \not\approx \quad \mathbf{P^{16}_{term}} = W_{term} \mid T_1 \mid T_2 \mid A_3$$

So in this case agent process A_1 can be linked with the completion of a task or tasks on which task process T_2 depends and agent process A_1 is not anonymous in respect of the attacker.

Experiment 9

Let us now consider the case where we have an additional agent A_4 with role *clerk* within our set of agents in our model.

We perform an equivalence test to ascertain whether or not:

$$\mathbf{P^{19}_{term}} = W_{term} \mid T_1 \mid T_2 \mid A_1 \mid A_3 \quad \approx \quad \mathbf{P^{20}_{term}} = W_{term} \mid T_1 \mid T_2 \mid A_3 \mid A_4$$

The fourth agent process A_4 is defined as follows:

$$A_4 \quad \triangleq \quad !agent\,\langle\langle ID_{A_4}, K_{A_4}\rangle\rangle$$

The agent/role assignment for agent A_4 is given as:

$role\,(ID_{A_4}, clerk)$

and this is written to the store R using the following process:

$$\mathcal{W}(R)(role(ID_{A_4}, clerk))$$

The trace comparison from $\mathbf{P}^{\mathbf{19}}_{\mathbf{term}}$ to $\mathbf{P}^{\mathbf{20}}_{\mathbf{term}}$ can be summarised as follows (detailed labelled reductions and static equivalences for this experiment are displayed in appendix C section C.2):

$\mathbf{P}^{\mathbf{19}}_{\mathbf{term}}$		$\mathbf{P}^{\mathbf{20}}_{\mathbf{term}}$
$\left\{W_1 \overset{C_{11}}{\Leftrightarrow} T_1, A_1\right\} \wr ID_{T_1}, pub_{T_1}$		$\left\{W_1 \overset{C_{12}}{\Leftrightarrow} T_1, A_4\right\} \wr ID_{T_1}, pub_{T_1}$
$\left\{W_2 \overset{C_{21}}{\Leftrightarrow} T_2, Att(A_2)\right\} \wr ID_{T_2}, pub_{T_2}, M_{T_2}$		$\left\{W_2 \overset{C_{22}}{\Leftrightarrow} T_2, Att(A_2)\right\} \wr ID_{T_2}, pub_{T_2}, M_{T_2}$

then we perform the same exercise from $\mathbf{P}^{\mathbf{20}}_{\mathbf{term}}$ to $\mathbf{P}^{\mathbf{19}}_{\mathbf{term}}$.

Based upon the tests performed above we can conclude that:

$$\mathbf{P}^{\mathbf{19}}_{\mathbf{term}} \approx \mathbf{P}^{\mathbf{20}}_{\mathbf{term}}$$

and we can infer from this result that anonymity is preserved in respect of an agent interacting with task T_1 from the perspective of a compromised agent A_2. In other words, a compromised agent and an attacker know that some agent must have interacted with task T_1 but they do not know the identity of that agent in the workflow model under investigation.

8.3.1.5 Conclusions

We demonstrate the application of the suite of security analysis tests for the applied pi calculus implementation of BPAC in respect of a trivial workflow example. We observe from the results of these tests that the security problems of BPAC highlighted in chapter 3 such as satisfiability, collusion and anonymity are properly identified by our tests.

8.3.2 Analysing a simple conference management system

8.3.2.1 Introduction

In order to demonstrate the application of reachability analysis to BPAC models of workflow-based access control we consider the example of a simple flawed conference management system per Zhang [104]. We present the example in section 8.3.2.2, then we encode the model in our BPAC modelling environment per section 8.3.2.3 and perform an analysis based upon reachability/satisfiability testing in section 8.3.2.4.

8.3.2.2 Outline of the conference management system

We turn our attention to an access control example that demonstrates weaknesses in a naive conference management system. For this particular test we consider the following simple flawed example of a conference management system as per Nan Zhang [104]:

1. The chair C of the PC (Programme Committee) assigns papers to PC members for review.
2. A PC member a can read PC member b's review for paper p provided that p is not assigned to a.
3. If two members a and b are both assigned paper p for reviewing a can read b's review for p only if a has already submitted its review for p.
4. Having been given a paper p member a can give up being a reviewer before reviewing finished.

We model the basic conference management system as a workflow comprising two connected sub-workflows of five tasks each and we allocate static constraints based upon the rules outlined above. For example, rule 4 above is implemented by tasks T_{3a}, T_{3b} (give up review). Completion of either of tasks T_{3a} and T_{5a} results in the triggering of the second sub-workflow so as to represent the idea of a loop back to the start of the workflow. Additionally, we add dummy tasks T_{join} and T_{end} to the overall combined workflow model to ensure consistency with the workflow requirements of chapter 3, specifically, the matching of XOR split tasks with XOR join tasks and the addition of a terminating task to the combined workflow. These tasks do not represent any activity. The list of tasks is outlined in table 8.1 and a diagram of the workflows is presented in figure 8.1.

8.3.2.3 A BPAC model of the conference management system

As usual, we model the access control problem as an interaction between processes: a workflow manager W, a collection of i parallel tasks and a collection of j parallel agents, initialised by process $\mathcal{M}$. We

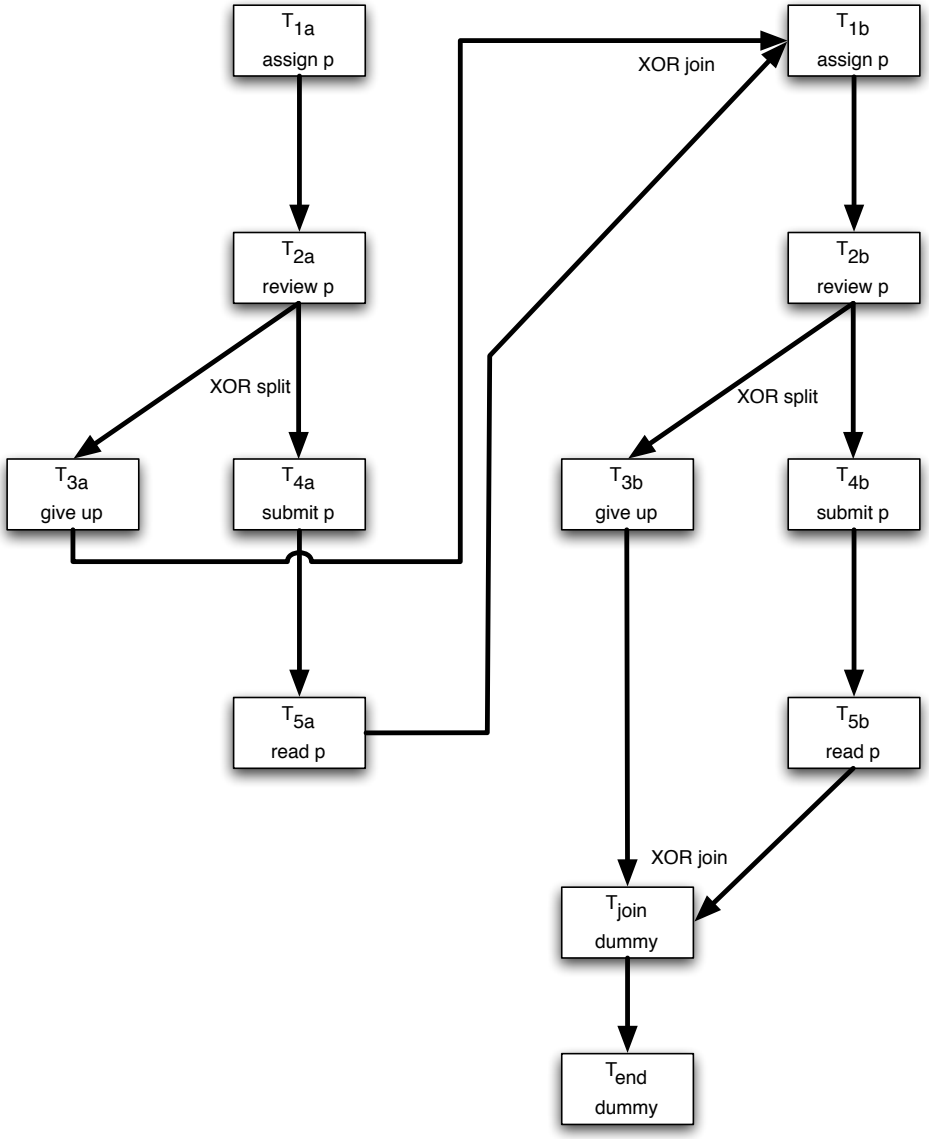


Figure 8.1: A flawed conference management system

Tasks	**Task detail**	**Constraint**
T_{1a}, T_{1b}	assign paper p to PC members for review	is a chair
T_{2a}, T_{2b}	review p	is assigned review of p
T_{3a}, T_{3b}	give up review of p	is assigned review of p
T_{4a}, T_{4b}	submit review of p	is assigned review of p
T_{5a}, T_{5b}	read review of p	is not assigned review of p OR has submitted a review of p
T_{join}	dummy join task	none
T_{end}	dummy end task	none

Table 8.1: List of tasks for the flawed conference management system

decide to use three agents in this model representing the chair C and members a and b as per the outline of the example in section 8.3.2.2. This number of agents is sufficient to allow the correct function of the protocol model. For instance, chair C assigns a review to member a and member b reads the review. The number is also sufficient to identify a flaw in the protocol such as agent a reading agent b's review without a submitting a review. We use the name p to represent the conference paper.

Firstly we define our initialisation process $\mathcal{M}$ as follows:

$$\begin{aligned}
&\nu K_W, K_{Org}, K_{T_1}, K_{T_2}, K_{T_3}, K_{T_4}, K_{T_5}, K_{join}, K_{end} \\
&K_{A_1}, K_{A_2}, K_{A_3}, ID_{A_1}, ID_{A_2}, ID_{A_3}, p, \\
&set, storeQ, storeR, storeS. \\
&(\mathcal{W}(R)(role(ID_{A_1}, PCMember)) \\
&.\mathcal{W}(R)(role(ID_{A_2}, PCMember)) \\
&.\mathcal{W}(R)(role(ID_{A_3}, PCMember)) \\
&.\mathcal{W}(R)(role(ID_{A_3}, Chair)) \mid \chi)
\end{aligned} \tag{8.15}$$

where, as usual, $\mathcal{X}$ represents all of the private internal support processes, such as the set handling processes and the store processes Q, R and S. Within $\mathcal{M}$ we assign $PCMember$ roles to the three agents that are participating in the model and we assign the $Chair$ to agent A_3.

We define the workflow manager as follows so that all of the constraints outlined in section 8.3.2.2 above are implemented:

$$
\begin{aligned}
W \triangleq\ & \nu ID_{T_1}, ID_{T_2}, ID_{T_3}, ID_{T_4}, ID_{T_5}, ID_{T_{join}}, ID_{T_{end}}, t_2, t_3, t_4, t_5, t_{join}, t_{end}. \\
& \left(t_W.taskCall\left\langle\left\langle ID_{T_1}, K_{T_1}, ID_W, K_W, [\mathcal{T}(R)(role(id_A, chairman)) :]\right\rangle\right\rangle.\overline{t_2}\right. \\
& \left| t_2.taskCall\left\langle\left\langle ID_{T_2}, K_{T_2}, ID_W, K_W, [\mathcal{T}(S)(review(id_A, p)) :]\right\rangle\right\rangle\right. \\
& .\left(\overline{t_3} + \overline{t_4}\right) \\
& \left| t_3.taskCall\left\langle\left\langle ID_{T_3}, K_{T_3}, ID_W, K_W, [\mathcal{T}(S)(review(id_A, p)) :]\right\rangle\right\rangle.\left(\overline{t_W} + \overline{t_{join}}\right)\right. \\
& \left| t_4.taskCall\left\langle\left\langle ID_{T_4}, K_{T_4}, ID_W, K_W, [\mathcal{T}(S)(review(id_A, p)) :]\right\rangle\right\rangle.\overline{t_5}\right. \\
& \left| t_5.\left(taskCall\left\langle\left\langle ID_{T_5}, K_{T_5}, ID_W, K_W, [\mathcal{T}(S)(review(id_A, p)) : \mathbf{0} :]\right\rangle\right\rangle\right.\right. \\
& \left.\left. + taskCall\left\langle\left\langle ID_{T_5}, K_{T_5}, ID_W, K_W, [\mathcal{T}(S)(submitted(id_A, p)) :]\right\rangle\right\rangle\right).\left(\overline{t_W} + \overline{t_{join}}\right)\right. \\
& \left| t_{join}.taskCall\left\langle\left\langle ID_{T_{join}}, K_{T_{join}}, ID_W, K_W, [\tau]\right\rangle\right\rangle.\overline{t_{end}}\right. \\
& \left| t_{end}.taskCall\left\langle\left\langle ID_{T_{end}}, K_{T_{end}}, ID_W, K_W, [\tau]\right\rangle\right\rangle\right)
\end{aligned}
$$

A complete model $\mathbf{P_{conf}}$ of the conference management system using the workflow manager and agent and task processes is as follows for the three participating agents and two runs of the workflow process:

$$
\begin{aligned}
\mathbf{P_{conf}} \triangleq\ & \nu t_W.\left(\overline{t_W} \mid W \mid W\right) \\
& \mid task\left\langle\left\langle ID_{T_{1a}}, K_{T_1}, M_{T_{1a}}, [\mathcal{W}(S)(review(id_A, p))], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{1b}}, K_{T_1}, M_{T_{1b}}, [\mathcal{W}(S)(review(id_A, p))], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{1c}}, K_{T_1}, M_{T_{1c}}, [\mathcal{W}(S)(review(id_A, p))], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{2a}}, K_{T_2}, M_{T_{2a}}, [\tau], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{2b}}, K_{T_2}, M_{T_{2b}}, [\tau], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{3a}}, K_{T_3}, M_{T_{3a}}, [\mathcal{D}(S)(review(id_A, p))], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{3b}}, K_{T_3}, M_{T_{3b}}, [\mathcal{D}(S)(review(id_A, p))], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{3c}}, K_{T_3}, M_{T_{3c}}, [\mathcal{D}(S)(review(id_A, p))], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{4a}}, K_{T_4}, M_{T_{4a}}, [\mathcal{D}(S)(review(id_A, p))]\right.\right. \\
& \left.\left., [\mathcal{W}(S)(submitted(id_A, p))]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{4b}}, K_{T_4}, M_{T_{4b}}, [\mathcal{D}(S)(review(id_A, p))]\right.\right. \\
& \left.\left., [\mathcal{W}(S)(submitted(id_A, p))]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{4c}}, K_{T_4}, M_{T_{4c}}, [\mathcal{D}(S)(review(id_A, p))]\right.\right. \\
& \left.\left., [\mathcal{W}(S)(submitted(id_A, p))]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{5a}}, K_{T_5}, M_{T_{5a}}, [\tau], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{5b}}, K_{T_5}, M_{T_{5b}}, [\tau], [\tau]\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle\left\langle ID_{T_{join}}, K_{T_{join}}, M_{T_{join}}, [\tau], [\tau]\right\rangle\right\rangle\right\rangle \\
& \mid task\left\langle\left\langle ID_{T_{end}}, K_{T_{end}}, M_{T_{end}}, [\tau], [\tau]\right\rangle\right\rangle \\
& \mid !agent\left\langle\left\langle ID_{A_1}, K_{A_1}\right\rangle\right\rangle \\
& \mid !agent\left\langle\left\langle ID_{A_2}, K_{A_2}\right\rangle\right\rangle \\
& \mid !agent\left\langle\left\langle ID_{A_3}, K_{A_3}\right\rangle\right\rangle
\end{aligned}
$$

Note that we have introduced multiple instances of the workflow manager and task processes within this model for the purpose of our investigation. Also, we have introduced multiple versions of the task resource messages M_{T_i} so that we can differentiate multiple task runs in our tests. Where a task process includes a store interaction there are three variants of the task process, recognising that the store interaction can be in respect of any of the three participating agents.

8.3.2.4 Analysing the model for dependency

A possible security test for this workflow model is to find out whether or not an agent can read another person's review of a paper that she is to review herself, in breach of rule 3 above. Within our model this security compromise can be represented by demonstrating that an agent can obtain the task message (M_{T_5}) for task 5 (T_5), followed by the task messages (M_{T_2} and M_{T_4}) for tasks 2 (T_2) and 4 (T_4) respectively. That is, some agent has access to task 5 and reads a review of paper p then subsequently she reviews paper p via task 2 and submits the review via task 4. Output of $M_{T_{end}}$ indicates that the workflow is satisfiable.

We define abstract reference model $\mathbf{Q_{conf}}$ as being a process that has the following transition steps:

$$\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\overset{\bar{t}\langle M_{T_{5a}}\rangle}{\longrightarrow}\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\overset{\bar{t}\langle M_{T_{2b}}\rangle}{\longrightarrow}\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\overset{\bar{t}\langle M_{T_{4b}}\rangle}{\longrightarrow}\overset{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\overset{\bar{t}\langle M_{T_{end}}\rangle}{\longrightarrow} \tag{8.16}$$

and we modify the process for agent A_2, $!agent\,\langle\langle ID_{A_2}, K_{A_2}\rangle\rangle$, in $\mathbf{P_{conf}}$ so that it can output its discovered task messages on the public channel t:

$$\begin{aligned} &!agent\,\langle\langle ID_{A_2}, K_{A_2}\rangle\rangle \mid (t\,(x_{A_{2a}})\,. \\ &\textbf{let}\ m_{A_{2a}} = fst\,(dec\,(x_{A_2}, sk\,(K_{A_2})))\ \textbf{in}\ \bar{t}\,\langle m_{A_{2a}}\rangle\,. \\ &t\,(x_{A_{2b}})\,.\textbf{let}\ m_{A_{2b}} = fst\,(dec\,(x_{A_2}, sk\,(K_{A_2})))\ \textbf{in}\ \bar{t}\,\langle m_{A_{2b}}\rangle\big) \end{aligned} \tag{8.17}$$

Note that on this occasion we have made the message disclosures serially dependent so that the second release of a discovered message can only occur after the first discovered message has been released. Now $\mathbf{P'_{conf}}$ represents the conference process $\mathbf{P_{conf}}$ with the original process for A_2 replaced by the modified process 8.17 above. If $\mathbf{P'_{conf}} \sim_{ab} \mathbf{Q_{conf}}$, we have demonstrated that rule 3 can be breached as this indicates that an agent can read another agent's review of the same paper before she has reviewed and submitted her own review of the paper. A trace comparison for this experiment is presented below:

No.	$\mathbf{P'_{conf}}$		$\mathbf{Q_{conf}}$
1	$W_1 \overset{C_{1a}}{\Leftrightarrow} T_{1a}, A_3$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
2	$W_2 \overset{C_{2a}}{\Leftrightarrow} T_{2a}, A_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
3	$W_4 \overset{C_{4a}}{\Leftrightarrow} T_{4a}, A_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
4	$W_5 \overset{C_{5a}}{\Leftrightarrow} T_{5a}, A_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
5	$\overset{t(x_{A_2}).\overline{t}\langle m_{A_2}\rangle}{\longrightarrow}$		$\overset{\overline{t}\langle M_{T_{5a}}\rangle}{\longrightarrow}$
6	$W_1 \overset{C_{1b}}{\Leftrightarrow} T_{1b}, A_3$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
7	$W_2 \overset{C_{2b}}{\Leftrightarrow} T_{2b}, A_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
8	$\overset{t(x_{A_2}).\overline{t}\langle m_{A_2}\rangle}{\longrightarrow}$		$\overset{\overline{t}\langle M_{T_{2b}}\rangle}{\longrightarrow}$
9	$W_4 \overset{C_{4b}}{\Leftrightarrow} T_{4b}, A_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
10	$\overset{t(x_{A_2}).\overline{t}\langle m_{A_2}\rangle}{\longrightarrow}$		$\overset{\overline{t}\langle M_{T_{4b}}\rangle}{\longrightarrow}$
11	$W_5 \overset{C_{5b}}{\Leftrightarrow} T_{5b}, A_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
12	$W_{join} \overset{C_{join}}{\Leftrightarrow} T_{join}, A_3$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
13	$W_{end} \overset{C_{end}}{\Leftrightarrow} T_{end}, A_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
14	$\overset{t(x_{A_2}).\overline{t}\langle m_{A_2}\rangle}{\longrightarrow}$		$\overset{\overline{t}\langle M_{T_{end}}\rangle}{\longrightarrow}$
	success		

Running the abstraction test does indeed identify that $\mathbf{P'_{conf}} \sim_{ab} \mathbf{Q_{conf}}$ i.e. an agent can read another agent's review of a paper p, before being assigned that paper for review, in breach of rule three.

8.3.2.5 Conclusions

We demonstrate with this model of access control for a naive conference management system that satisfiability tests based upon the abstraction test can be used to highlight particular flaws in the access control model within BPAC. We note that the example highlights issues with activation and deactivation of roles and that BPAC does not support this. However, we demonstrate that BPAC can model and analyse this particular example through the use of task-based dynamic constraints. We consider that such an approach is more appropriate as role assignment lacks the required granularity. Access control based solely upon the allocation of roles such as "reviewer", say, provides access to a whole class of resources, such as papers in the example, but our approach applies access control to the individual papers themselves.

It could be argued that the test material and the test itself is somewhat contrived and that real-world discovery of flaws would be rather more difficult than it appears from this example. Certainly, this example has been selected to provide a proof of concept with known flaws to demonstrate the test

process. Careful selection of test strategies and formulation of the abstract reference model would be required in practice.

8.4 Computational complexity

8.4.1 Introduction

In section 8.3 above we provide some toy examples of security analysis for models of workflow-based access control within our BPAC environment. As the example workflow model comprises only two tasks and three or four agents then the number of possible processes required for each test is maintained at manageable levels. In practice, the workflow models are likely to be much more complex and the number of processes required will be substantially greater. In this section we discuss the issue of computational complexity with regard to the collusion metric test. Discussions regarding the other security tests would follow a broadly similar theme.

8.4.2 Complexity of the extended collusion metric computation

In section 8.3.1.3 we compute the extended collusion metric for our two-task toy workflow example. A number of processes are required to perform this computation as experiments have to be performed for different combinations of agent processes. In general, we can state that the identification of the extended collusion metric depends upon the execution of multiple experiments based upon the number of agent processes within the model. Indeed in the worst case we are required to perform $2^{N_A} - 1$ experiments where N_A is the total number of participating agent processes within the model. Whilst a test with a worst time complexity of $O\left(2^{N_A}\right)$ is rather less than ideal, it is anticipated that, in practice, substantially fewer experiments would need to be performed, as we discuss below.

Discovery of a valid execution path for a workflow of tasks and a set of agents requires multiple executions of the order of $(N_A)^{N_T}$ where N_A is the total number of agents as before and N_T is the total number of tasks in the workflow model. Again we are faced with exponential time complexity. In practice, most of these outcomes are likely to fail due to access control constraints but for a workflow, task and agent model there may be numerous possible execution paths depending upon the number of agents that can successfully interact with the workflow manager and tasks. As we are going to have to perform many experiments for different agents releasing secrets then we only need to check the full number of outcomes once as the release of secrets is independent of the access control process.

A further problem to note is that the experiments themselves may have a multiplicity of outcomes. In our toy example the only satisfiable outcome is obtained from agent processes A_1 and A_2 and the subsequent experiments are relatively straightforward. However, in a more complex example, like our purchases workflow, there could be a number of valid agent mixes that lead to satisfiable outcomes.

Mitigating against computational explosion in extended collusion metric analysis

We suggest above that a problem with the extended collusion metric analysis as outlined above is worst time complexity of $O(2^n)$ in respect of the performance of experiments. In practice, time complexity can be reduced somewhat.

Given a number of agents N_A and a number of tasks N_T and given that no more than one agent can interact with a task via the standard core protocol then:

If $N_A \geq N_T$, we do not need to test for combinations of agents greater than or equal to N_T out of a total of N_A agents and only if $N_A < N_T$ do we need, potentially, to test up to N_A agents.

An organisation is likely to be interested in a minimum value for the collusion metric. That is, an organisation would seek to confirm that the collusion metric for a particular workflow-based access control model does not fall below some value. For example, given a number of agents and a workflow model it might be decided that a minimum collusion metric is three, say. The strategy, therefore, would be to perform experiments with single agent leaks of secrets, then experiments with pairs and if all tests are false and if the model passes the satisfiability test then we can conclude that the collusion metric is at least three. For a large set of agents a small minimum collusion metric produces the greatest computational savings with respect to the worst case time complexity.

It may be possible to reduce the size of the set of participating agents by removing those agents that clearly play no part in the workflow model. However, care needs to be taken as agents that are not directly associated with the workflow might nonetheless be a communication conduit or a process facilitator enabling other agents to obtain secrets. An agent might gain control over a number of task secrets and yet play no direct part in the workflow interaction.

A further possibility is to reduce the workflow to smaller components and perform experiments on these workflow components instead of on the workflow in its entirety.

8.4.3 Conclusion

We recognise that computational explosion is a problem with our security analysis tests within our modelling environment. We observe that there may be possible strategies to mitigate against this problem

and it is likely that improvements to our test methods could yield improved performance in this respect. Whilst we would like to use the applied pi calculus modelling environment to analyse complex access control problems it is not yet possible to implement this approach using an automated checking tool such as ProVerif. Additional work is required in this regard. For example, ProVerif does not implement observational equivalence fully and it is unable to perform our abstraction test. Also, ProVerif doe not provide a mechanism for retaining state histories as exemplified by the use of stores in our modelling environment. These issues have to be addressed if our modelling environment is to be implemented by an automated checking tool. Furthermore, the high level trace comparisons that we use as part of our security tests and which provide shortcuts to our protocol analysis also need to be implemented in an automated checking tool. An automated checking tool, when so amended, in addition to the use of optimising strategies, would provide a practical application of our BPAC modelling environment and its security tests in mitigation against the complexity issues discussed above.

8.5 Summary

In this chapter we present a discussion on the analysis of security properties within access control models using our modelling environment and this is followed by a detailed presentation of analysis techniques for a number of security tests. We introduce a lightweight test called the abstraction test and we demonstrate its use and flexibility over a number of security analysis examples for model systems of workflow-based access control. The security test for anonymity presented in section 8.2.7, with example experiments in 8.3.1.4, utilises the full power of observational equivalence/labelled bisimilarity, a key strength of the applied pi calculus. However, we identify complexity issues with the implementation of security tests to all but the smallest toy access control examples and further work is required in this respect, particularly with regard to automation of the modelling environment.

Chapter 9

Analysing the simple workflow

9.1 Introduction

In chapter 4 we introduce a simple BPAC purchases workflow example and we present the applied pi calculus coding of this example in chapter 7. In this chapter we return to our purchases workflow and we perform a collection of security tests on the BPAC workflow model. In section 9.2 we perform satisfiability checks and in section 9.3 we calculate the collusion metric for this particular example. In section 9.4 we perform anonymity tests on the purchases model.

9.2 Testing the purchases workflow for satisfiability

Given the terminating simple purchases workflow model $\mathbf{P_{termp}}$ (7.7), initialised by $\mathcal{M}$ (7.1), detailed in chapters 4 and 7 above, we seek to establish that, given the access control constraints and given the set of available agents:

$$A \quad = \quad \{Agent_1,\ Agent_2,\ Agent_3,\ Agent_4, Agent_5\}$$

a trace can be found in which the complete workflow can be executed.

9.2.1 Procedure

We attempt to find a trace of our model $\mathbf{P_{termp}}$ that is indistinguishable from the trace for an abstract reference model $\mathbf{Q_p}$. We use the procedure that we describe as the abstraction test as outlined in section 8.2.3 above.

Now we formulate the labelled transition behaviour of our abstract reference model $\mathbf{Q_p}$ as follows:

$$\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$$

that is, any number of fresh names broadcast over the public channel t are followed by the output of name M_{end} over t. We perform the abstraction test to establish or otherwise:

$$\mathbf{P_{termp}} \quad \sim_{ab} \quad \mathbf{Q_p}$$

that is we seek to establish whether or not there exists a trace of $\mathbf{P_{termp}}$ that behaves as $\mathbf{Q_p}$ in so far as an external observer is concerned.

Firstly, we rewrite $\mathbf{P_{termp}}$ as follows:

$$
\begin{aligned}
\mathbf{P_{termp}} \triangleq\ & W_{term} \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid Agent_1 \mid Agent_2 \mid Agent_3 \\
& \mid Agent_4 \mid Agent_5 \\
=\ & \nu ID_W, ID_{Task_1}, ID_{Task_2}, ID_{Task_3}, \\
& ID_{Task_4}, ID_{Task_5}, ID_{Task_6}, \\
& t_2, t_3, t_4, t_{5a}, t_{5b}, t_6, t_{end}. \\
& \left(W_1.\overline{t_2} \mid t_2.W_2.\left(\overline{t_3} \mid \overline{t_4}\right)\right. \\
& \mid t_3.W_3.\overline{t_{5a}} \mid t_4.W_4.\overline{t_{5b}} \\
& \mid t_{5a}.t_{5b}.W_5.\overline{t_6} \mid t_6.W_6.\overline{t_{end}} \\
& \left. \mid t_{end}.\bar{t}\langle M_{end}\rangle\right) \\
& \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid Agent_1 \mid Agent_2 \mid Agent_3 \\
& \mid Agent_4 \mid Agent_5
\end{aligned}
$$

by substitution using 7.5 above.

As in our examples in the previous chapters, this rewriting of $\mathbf{P_{termp}}$ highlights the core protocol transitions. In this case, there are six core protocol transitions being the interaction between W_1, one of $Task_1 \ldots Task_6$ and one of $Agent_1 \ldots Agent_5$, which we refer to as core protocol instance C_1; the interaction between W_2, one of $Task_1 \ldots Task_6$ and one of $Agent_1 \ldots Agent_5$, which we refer to as core protocol instance C_2 and so on through to protocol instance C_6.

9.2.2 Results

We attempt to find a trace of $\mathbf{P_{termp}}$ that behaves like $\mathbf{Q_p}$ (summarised below):

No.	$\mathbf{P_{termp}}$		$\mathbf{Q_p}$
1	$W_1 \overset{C_1}{\Leftrightarrow} Task_1, Agent_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
2	$W_2 \overset{C_2}{\Leftrightarrow} Task_2, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
3	$W_3 \overset{C_3}{\Leftrightarrow} Task_3, Agent_4$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
4	$W_4 \overset{C_4}{\Leftrightarrow} Task_4, Agent_5$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
5	$W_5 \overset{C_5}{\Leftrightarrow} Task_5, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
6	$W_6 \overset{C_6}{\Leftrightarrow} Task_6, Agent_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
7	$\overset{\overline{t}\langle M_{end}\rangle}{\longrightarrow}$		$\overset{\overline{t}\langle M_{end}\rangle}{\longrightarrow}$
	success		

We can conclude from the above trace comparison that:

$$\mathbf{P_{termp}} \sim_{ab} \mathbf{Q_p}$$

That is, there exists a trace where workflow sub-process W_1 interacts with task process $Task_1$ and agent $Agent_1$ (core protocol instance C_1), workflow sub-process W_2 interacts with task process $Task_2$ and agent $Agent_2$(core protocol instance C_2) and so on through to the terminating process $\overline{t}\,\langle M_{end}\rangle$ and this trace is similar to the trace of test process $\mathbf{Q_p}$.

9.2.3 Conclusion

Based upon the experiment outlined above we can conclude that the workflow-based access control model is satisfiable using the specified access control constraints and given the set of agents:

$$\{Agent_1, Agent_2, Agent_3, Agent_4, Agent_5\}$$

and the set of tasks:

$$\{Task_1, Task_2, Task_3, Task_4, Task_5, Task_6\}$$

9.3 The extended collusion metric for the simple purchases workflow

9.3.1 Procedure

Firstly, we modify our terminating workflow model $\mathbf{P_{termp}}$ so that all agents $Agent_1 \ldots Agent_5$ can output any messages received from interactions with the workflow manager W_{term} and tasks $Task_1 \ldots Task_6$, noting that the agent processes as per A.5.2 above output any received messages on public channel t encrypted under the agent's own public key:

$$
\begin{aligned}
\mathbf{P'_{termp}} \triangleq\ & W_{term} \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid Agent_1 \mid Agent_2 \mid Agent_3 \\
& \mid Agent_4 \mid Agent_5 \\
=\ & \nu ID_W, ID_{Task_1}, ID_{Task_2}, ID_{Task_3}, \\
& ID_{Task_4}, ID_{Task_5}, ID_{Task_6}, \\
& t_2, t_3, t_4, t_{5a}, t_{5b}, t_6, t_{end}. \\
& \big(W_1.\overline{t_2} \mid t_2.W_2.\big(\overline{t_3} \mid \overline{t_4}\big) \\
& \mid t_3.W_3.\overline{t_{5a}} \mid t_4.W_4.\overline{t_{5b}} \\
& \mid t_{5a}.t_{5b}.W_5.\overline{t_6} \mid t_6.W_6.\overline{t_{end}} \\
& \mid t_{end}.\overline{t}\langle M_{end}\rangle\big) \\
& \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid (Agent_1 \mid\mid !(t(x_{Agent_1}).\mathbf{let}\ m_{Agent_1} \\
& = fst(dec(x_{Agent_1}, sk(K_{Agent_1})))\ \mathbf{in}\ \overline{t}\langle m_{Agent_1}\rangle)) \\
& \mid (Agent_2 \mid\mid !(t(x_{Agent_2}).\mathbf{let}\ m_{Agent_2} \\
& = fst(dec(x_{Agent_2}, sk(K_{Agent_2})))\ \mathbf{in}\ \overline{t}\langle m_{Agent_2}\rangle)) \\
& \mid (Agent_3 \mid\mid !(t(x_{Agent_3}).\mathbf{let}\ m_{Agent_3} \\
& = fst(dec(x_{Agent_3}, sk(K_{Agent_3})))\ \mathbf{in}\ \overline{t}\langle m_{Agent_3}\rangle)) \\
& \mid (Agent_4 \mid\mid !(t(x_{Agent_4}).\mathbf{let}\ m_{Agent_4} \\
& = fst(dec(x_{Agent_4}, sk(K_{Agent_4})))\ \mathbf{in}\ \overline{t}\langle m_{Agent_4}\rangle)) \\
& \mid (Agent_5 \mid\mid !(t(x_{Agent_5}).\mathbf{let}\ m_{Agent_5} \\
& = fst(dec(x_{Agent_5}, sk(K_{Agent_5})))\ \mathbf{in}\ \overline{t}\langle m_{Agent_5}\rangle))
\end{aligned}
\tag{9.1}
$$

Next, we define our abstract model $\mathbf{Q'_p}$, which will be the target for our analysis using the format outlined in section 8.2.5. Our model $\mathbf{Q'_p}$ summarises all of the messages of our workflow model that could be observed on a public channel if they were released to the public channel. The labelled transition behaviour of our abstract model $\mathbf{Q'_p}$ is specified as follows:

$$\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_1\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_2\rangle}{\longrightarrow}\left(\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_3\rangle}{\longrightarrow}\wedge\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_4\rangle}{\longrightarrow}\right)\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_5\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_6\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_{end}\rangle}{\longrightarrow} \tag{9.2}$$

Note the conjunctive form representing the parallel output of messages M_3 and M_4. This requires us to perform two comparisons for each experiment, comparison against:

$$\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_1\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_2\rangle}{\longrightarrow}\left(\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_3\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_4\rangle}{\longrightarrow}\right)\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_5\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_6\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$$

and comparison against:

$$\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_1\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_2\rangle}{\longrightarrow}\left(\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_4\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_3\rangle}{\longrightarrow}\right)\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_5\rangle}{\longrightarrow}\stackrel{\nu n.\bar{t}\langle n\rangle}{\Longrightarrow}\stackrel{\bar{t}\langle M_6\rangle}{\longrightarrow}\stackrel{\bar{t}\langle M_{end}\rangle}{\longrightarrow}$$

in order to confirm full parallel workflow behaviour in our test model.

9.3.2 Experiment with $\mathbf{P'_{termp}}$ unmodified — five agent outputs

9.3.2.1 Experiment 1

Our first experiment compares traces of $\mathbf{P'_{termp}}$ as per 9.1 above with the trace of $\mathbf{Q'_p}$ as per 9.2 above. The results are as follows:

No.	$\mathbf{P'_{termp}}$		$\mathbf{Q'_p}$
1	$W_1 \overset{C_1}{\Leftrightarrow} Task_1, Agent_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
2	$(t(x_{Agent_1}).\overline{t}\langle m_{Agent_1}\rangle) \longrightarrow$		$\overline{t}\langle M_1\rangle \longrightarrow$
3	$W_2 \overset{C_2}{\Leftrightarrow} Task_2, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
4	$(t(x_{Agent_2}).\overline{t}\langle m_{Agent_2}\rangle) \longrightarrow$		$\overline{t}\langle M_2\rangle \longrightarrow$
5	$W_3 \overset{C_3}{\Leftrightarrow} Task_3, Agent_4$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
6	$(t(x_{Agent_4}).\overline{t}\langle m_{Agent_4}\rangle) \longrightarrow$		$\overline{t}\langle M_3\rangle \longrightarrow$
7	$W_4 \overset{C_4}{\Leftrightarrow} Task_4, Agent_5$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
8	$(t(x_{Agent_5}).\overline{t}\langle m_{Agent_5}\rangle) \longrightarrow$		$\overline{t}\langle M_4\rangle \longrightarrow$
9	$W_5 \overset{C_5}{\Leftrightarrow} Task_5, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
10	$(t(x_{Agent_2}).\overline{t}\langle m_{Agent_2}\rangle) \longrightarrow$		$\overline{t}\langle M_5\rangle \longrightarrow$
11	$W_6 \overset{C_6}{\Leftrightarrow} Task_6, Agent_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
12	$(t(x_{Agent_1}).\overline{t}\langle m_{Agent_1}\rangle) \longrightarrow$		$\overline{t}\langle M_6\rangle \longrightarrow$
13	$\overline{t}\langle M_{end}\rangle \longrightarrow$		$\overline{t}\langle M_{end}\rangle \longrightarrow$
	success		

and:

No.	$\mathbf{P'_{termp}}$		$\mathbf{Q'_p}$
1	$W_1 \overset{C_1}{\Leftrightarrow} Task_1, Agent_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
2	$(t(x_{Agent_1}).\overline{t}\langle m_{Agent_1}\rangle) \longrightarrow$		$\overline{t}\langle M_1\rangle \longrightarrow$
3	$W_2 \overset{C_2}{\Leftrightarrow} Task_2, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
4	$(t(x_{Agent_2}).\overline{t}\langle m_{Agent_2}\rangle) \longrightarrow$		$\overline{t}\langle M_2\rangle \longrightarrow$
5	$W_4 \overset{C_4}{\Leftrightarrow} Task_4, Agent_4$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
6	$(t(x_{Agent_4}).\overline{t}\langle m_{Agent_4}\rangle) \longrightarrow$		$\overline{t}\langle M_4\rangle \longrightarrow$
7	$W_3 \overset{C_3}{\Leftrightarrow} Task_3, Agent_5$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
8	$(t(x_{Agent_5}).\overline{t}\langle m_{Agent_5}\rangle) \longrightarrow$		$\overline{t}\langle M_3\rangle \longrightarrow$
9	$W_5 \overset{C_5}{\Leftrightarrow} Task_5, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
10	$(t(x_{Agent_2}).\overline{t}\langle m_{Agent_2}\rangle) \longrightarrow$		$\overline{t}\langle M_5\rangle \longrightarrow$
11	$W_6 \overset{C_6}{\Leftrightarrow} Task_6, Agent_1$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
12	$(t(x_{Agent_1}).\overline{t}\langle m_{Agent_1}\rangle) \longrightarrow$		$\overline{t}\langle M_6\rangle \longrightarrow$
13	$\overline{t}\langle M_{end}\rangle \longrightarrow$		$\overline{t}\langle M_{end}\rangle \longrightarrow$
	success		

therefore we can conclude that:

$$\mathbf{P'_{termp}} \quad \sim_{ab} \quad \mathbf{Q'_{p}}$$

and the set of agents $\{Agent_1, Agent_2, Agent_3, Agent_4, Agent_5\}$ has access to all of the secrets that comprise the complete workflow.

9.3.3 Experiments with $\mathbf{P'_{termp}}$ modified — four agent outputs

9.3.3.1 Experiment 2: removing $Agent_1$ output

Our second experiment compares:

$$
\begin{aligned}
\mathbf{P''_{termp}} \triangleq\ & W_{term} \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid Agent_1 \mid Agent_2 \mid Agent_3 \\
& \mid Agent_4 \mid Agent_5 \\
=\ & \nu ID_W, ID_{Task_1}, ID_{Task_2}, ID_{Task_3}, \\
& ID_{Task_4}, ID_{Task_5}, ID_{Task_6}, \\
& t_2, t_3, t_4, t_{5a}, t_{5b}, t_6, t_{end}. \\
& \left(W_1.\overline{t_2} \mid t_2.W_2.\left(\overline{t_3} \mid \overline{t_4}\right)\right. \\
& \mid t_3.W_3.\overline{t_{5a}} \mid t_4.W_4.\overline{t_{5b}} \\
& \mid t_{5a}.t_{5b}.W_5.\overline{t_6} \mid t_6.W_6.\overline{t_{end}} \\
& \left. \mid t_{end}.\overline{t}\langle M_{end}\rangle\right) \\
& \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid Agent_1 \\
& \mid (Agent_2 \mid !\,(t\,(x_{Agent_2})\,.\mathbf{let}\ m_{Agent_2} \\
& = fst\,(dec\,(x_{Agent_2}, sk\,(K_{Agent_2})))\ \mathbf{in}\ \overline{t}\,\langle m_{Agent_2}\rangle)) \\
& \mid (Agent_3 \mid !\,(t\,(x_{Agent_3})\,.\mathbf{let}\ m_{Agent_3} \\
& = fst\,(dec\,(x_{Agent_3}, sk\,(K_{Agent_3})))\ \mathbf{in}\ \overline{t}\,\langle m_{Agent_3}\rangle)) \\
& \mid (Agent_4 \mid !\,(t\,(x_{Agent_4})\,.\mathbf{let}\ m_{Agent_4} \\
& = fst\,(dec\,(x_{Agent_4}, sk\,(K_{Agent_4})))\ \mathbf{in}\ \overline{t}\,\langle m_{Agent_4}\rangle)) \\
& \mid (Agent_5 \mid !\,(t\,(x_{Agent_5})\,.\mathbf{let}\ m_{Agent_5} \\
& = fst\,(dec\,(x_{Agent_5}, sk\,(K_{Agent_5})))\ \mathbf{in}\ \overline{t}\,\langle m_{Agent_5}\rangle))
\end{aligned}
$$

i.e. agent $Agent_1$ does not output any acquired messages on t, to $\mathbf{Q'_p}$ and the results are as follows:

No.	$\mathbf{P''_{termp}}$		$\mathbf{Q'_p}$
1	$W_1 \overset{C_1}{\Leftrightarrow} Task_1, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
2	$(t(x_{Agent_2}).\overline{t}\langle m_{Agent_2}\rangle) \longrightarrow$		$\overline{t}\langle M_1\rangle \longrightarrow$
3	$W_2 \overset{C_2}{\Leftrightarrow} Task_2, Agent_3$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
4	$(t(x_{Agent_3}).\overline{t}\langle m_{Agent_3}\rangle) \longrightarrow$		$\overline{t}\langle M_2\rangle \longrightarrow$
5	$W_3 \overset{C_3}{\Leftrightarrow} Task_3, Agent_4$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
6	$(t(x_{Agent_4}).\overline{t}\langle m_{Agent_4}\rangle) \longrightarrow$		$\overline{t}\langle M_3\rangle \longrightarrow$
7	$W_4 \overset{C_4}{\Leftrightarrow} Task_4, Agent_5$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
8	$(t(x_{Agent_5}).\overline{t}\langle m_{Agent_5}\rangle) \longrightarrow$		$\overline{t}\langle M_4\rangle \longrightarrow$
9	$W_5 \overset{C_5}{\Leftrightarrow} Task_5, Agent_3$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
10	$(t(x_{Agent_3}).\overline{t}\langle m_{Agent_3}\rangle) \longrightarrow$		$\overline{t}\langle M_5\rangle \longrightarrow$
11	$W_6 \overset{C_6}{\Leftrightarrow} Task_6, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
12	$(t(x_{Agent_2}).\overline{t}\langle m_{Agent_2}\rangle) \longrightarrow$		$\overline{t}\langle M_6\rangle \longrightarrow$
13	$\overline{t}\langle M_{end}\rangle \longrightarrow$		$\overline{t}\langle M_{end}\rangle \longrightarrow$
	success		

and:

No.	$\mathbf{P''_{termp}}$		$\mathbf{Q'_p}$
1	$W_1 \overset{C_1}{\Leftrightarrow} Task_1, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
2	$(t(x_{Agent_2}).\overline{t}\langle m_{Agent_2}\rangle) \longrightarrow$		$\overline{t}\langle M_1\rangle \longrightarrow$
3	$W_2 \overset{C_2}{\Leftrightarrow} Task_2, Agent_3$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
4	$(t(x_{Agent_3}).\overline{t}\langle m_{Agent_3}\rangle) \longrightarrow$		$\overline{t}\langle M_2\rangle \longrightarrow$
5	$W_4 \overset{C_4}{\Leftrightarrow} Task_4, Agent_4$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
6	$(t(x_{Agent_4}).\overline{t}\langle m_{Agent_4}\rangle) \longrightarrow$		$\overline{t}\langle M_4\rangle \longrightarrow$
7	$W_3 \overset{C_3}{\Leftrightarrow} Task_3, Agent_5$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
8	$(t(x_{Agent_5}).\overline{t}\langle m_{Agent_5}\rangle) \longrightarrow$		$\overline{t}\langle M_3\rangle \longrightarrow$
9	$W_5 \overset{C_5}{\Leftrightarrow} Task_5, Agent_3$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
10	$(t(x_{Agent_3}).\overline{t}\langle m_{Agent_3}\rangle) \longrightarrow$		$\overline{t}\langle M_5\rangle \longrightarrow$
11	$W_6 \overset{C_6}{\Leftrightarrow} Task_6, Agent_2$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
12	$(t(x_{Agent_2}).\overline{t}\langle m_{Agent_2}\rangle) \longrightarrow$		$\overline{t}\langle M_6\rangle \longrightarrow$
13	$\overline{t}\langle M_{end}\rangle \longrightarrow$		$\overline{t}\langle M_{end}\rangle \longrightarrow$
	success		

therefore we can conclude that:

$$\mathbf{P''_{termp}} \sim_{ab} \mathbf{Q'_{p}}$$

and the set of agents $\{Agent_2, Agent_3, Agent_4, Agent_5\}$ has access to all of the secrets that comprise the complete workflow.

Now, as we have successfully found a trace comparison for four agents we do not need to consider any other cases for four agents.

9.3.4 Experiments with $\mathbf{P'_{termp}}$ modified — three agent outputs

9.3.4.1 Experiment 3: removing $Agent_1$ and $Agent_2$ output

Our third experiment compares:

$$
\begin{aligned}
\mathbf{P}'''_{\mathbf{termp}} \triangleq\ & W_{term} \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid Agent_1 \mid Agent_2 \mid Agent_3 \\
& \mid Agent_4 \mid Agent_5 \\
=\ & \nu ID_W, ID_{Task_1}, ID_{Task_2}, ID_{Task_3}, \\
& ID_{Task_4}, ID_{Task_5}, ID_{Task_6}, \\
& t_2, t_3, t_4, t_{5a}, t_{5b}, t_6, t_{end}. \\
& \left(W_1.\overline{t_2} \mid t_2.W_2.\left(\overline{t_3} \mid \overline{t_4}\right)\right. \\
& \mid t_3.W_3.\overline{t_{5a}} \mid t_4.W_4.\overline{t_{5b}} \\
& \mid t_{5a}.t_{5b}.W_5.\overline{t_6} \mid t_6.W_6.\overline{t_{end}} \\
& \left. \mid t_{end}.\overline{t}\langle M_{end}\rangle\right) \\
& \mid Task_1 \mid Task_2 \mid Task_3 \\
& \mid Task_4 \mid Task_5 \mid Task_6 \\
& \mid Agent_1 \mid Agent_2 \\
& \mid (Agent_3 \mid !(t(x_{Agent_3}).\mathbf{let}\ m_{Agent_3} \\
& = fst(dec(x_{Agent_3}, sk(K_{Agent_3})))\ \mathbf{in}\ \overline{t}\langle m_{Agent_3}\rangle)) \\
& \mid (Agent_4 \mid !(t(x_{Agent_4}).\mathbf{let}\ m_{Agent_4} \\
& = fst(dec(x_{Agent_4}, sk(K_{Agent_4})))\ \mathbf{in}\ \overline{t}\langle m_{Agent_4}\rangle)) \\
& \mid (Agent_5 \mid !(t(x_{Agent_5}).\mathbf{let}\ m_{Agent_5} \\
& = fst(dec(x_{Agent_5}, sk(K_{Agent_5})))\ \mathbf{in}\ \overline{t}\langle m_{Agent_5}\rangle))
\end{aligned}
$$

i.e. agents $Agent_1$ and $Agent_2$ do not output any acquired messages on t, to $\mathbf{Q}'_{\mathbf{p}}$ and the trace comparison results are as follows:

No.	$\mathbf{P'''_{termp}}$		$\mathbf{Q'_p}$
1	$W_1 \overset{C_1}{\Leftrightarrow} Task_1, Agent_3$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
2	$\overset{(t(x_{Agent_3}).\overline{t}\langle m_{Agent_3}\rangle)}{\longrightarrow}$		$\overset{\overline{t}\langle M_1\rangle}{\longrightarrow}$
3	$W_2 \overset{C_2}{\Leftrightarrow} Task_2, Agent_4$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
4	$\overset{(t(x_{Agent_4}).\overline{t}\langle m_{Agent_4}\rangle)}{\longrightarrow}$		$\overset{\overline{t}\langle M_2\rangle}{\longrightarrow}$
5	$W_3 \overset{C_3}{\Leftrightarrow} Task_3, Agent_5$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
6	$\overset{(t(x_{Agent_5}).\overline{t}\langle m_{Agent_5}\rangle)}{\longrightarrow}$		$\overset{\overline{t}\langle M_3\rangle}{\longrightarrow}$
7	fail		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$

and:

No.	$\mathbf{P'''_{termp}}$		$\mathbf{Q'_p}$
1	$W_1 \overset{C_1}{\Leftrightarrow} Task_1, Agent_3$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
2	$\overset{(t(x_{Agent_3}).\overline{t}\langle m_{Agent_3}\rangle)}{\longrightarrow}$		$\overset{\overline{t}\langle M_1\rangle}{\longrightarrow}$
3	$W_2 \overset{C_2}{\Leftrightarrow} Task_2, Agent_4$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
4	$\overset{(t(x_{Agent_4}).\overline{t}\langle m_{Agent_4}\rangle)}{\longrightarrow}$		$\overset{\overline{t}\langle M_2\rangle}{\longrightarrow}$
5	$W_4 \overset{C_4}{\Leftrightarrow} Task_4, Agent_5$		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$
6	$\overset{(t(x_{Agent_5}).\overline{t}\langle m_{Agent_5}\rangle)}{\longrightarrow}$		$\overset{\overline{t}\langle M_4\rangle}{\longrightarrow}$
7	fail		$\overset{\nu n.\overline{t}\langle n\rangle}{\Longrightarrow}$

therefore we can conclude that:

$$\mathbf{P'''_{termp}} \quad \not\sim_{ab} \quad \mathbf{Q'_p}$$

and the set of agents $\{Agent_3, Agent_4, Agent_5\}$ does not have access to all of the secrets that comprise the complete workflow.

9.3.4.2 Further experiments

We repeat experiment 3 above only this time we remove the output for $Agent_1$ and $Agent_3$ from 9.1 above. We discover that $\{Agent_2, Agent_4, Agent_5\}$ does not have access to all of the secrets that comprise the complete workflow. Performing the experiments on all of the remaining combinations of 9.1 with two agent outputs removed we find that:

$\{Agent_2, Agent_3, Agent_5\}$,

$\{Agent_2, Agent_3, Agent_4\}$,

$\{Agent_1, Agent_4, Agent_5\}$,

$\{Agent_1, Agent_3, Agent_5\}$,

$\{Agent_1, Agent_3, Agent_4\}$,

$\{Agent_1, Agent_2, Agent_5\}$,

$\{Agent_1, Agent_2, Agent_4\}$,

$\{Agent_1, Agent_2, Agent_3\}$

do not have access to all of the secrets that comprise the complete workflow. In other words, no set of three agents has access to all of the secrets that comprise the complete workflow.

9.3.5 Conclusion

Given that we have established that there exists a set of four agents for which the comparison test is successful and there is no set of three agents for which the comparison test is successful then we conclude that the extended collusion metric for our purchases workflow is four. In other words, at least four agents would need to collude in order to obtain complete control over the purchases workflow. Judgements as to the level of security of the purchase workflow indicated by the collusion metric depend upon the circumstances of the workflow such as the interaction with other workflows, the importance or significance of the workflow within an organisation and the management attitude to risk. However, in this particular case it can be observed that a collusion metric of four for a six-task workflow represents, in most circumstances, a high level of security and an effective implementation of segregation of duties. Note that we would have obtained the same result had we calculated the simple collusion metric as there is no interaction between agents in the example.

9.4 Anonymity testing

We perform anonymity tests on the simple access control model of chapter 7.

9.4.1 Critical agent tests

9.4.1.1 Procedure

Firstly, let us consider our example workflow from chapter 7:

$$\begin{aligned}\mathbf{P_{purchase}} \triangleq\ & W \mid Task_1 \mid Task_2 \mid Task_3 \\ & \mid Task_4 \mid Task_5 \mid Task_6 \\ & \mid Agent_1 \mid Agent_2 \mid Agent_3 \\ & \mid Agent_4 \mid Agent_5\end{aligned}$$

initialised by $\mathcal{M}$, and assume that no agents have been compromised. Let us test our model to see if $Agent_1$ is critical to the model. If we define $\mathbf{P^4_{purchase}}$ as our purchase model $\mathbf{P_{purchase}}$ without agent process $Agent_1$:

$$\begin{aligned}\mathbf{P^4_{purchase}} \triangleq\ & W \mid Task_1 \mid Task_2 \mid Task_3 \\ & \mid Task_4 \mid Task_5 \mid Task_6 \\ & \mid Agent_2 \mid Agent_3 \\ & \mid Agent_4 \mid Agent_5\end{aligned}$$

Then if $\mathbf{P_{purchase}} \approx \mathbf{P^4_{purchase}}$ we can conclude that $Agent_1$ is not critical to $\mathbf{P_{purchase}}$.

9.4.1.2 Results

We run comparison traces and static equivalences for all permutations of agent processes interacting with the workflow manager and task processes comparing model $\mathbf{P_{purchase}}$ to $\mathbf{P^4_{purchase}}$ and $\mathbf{P^4_{purchase}}$ to $\mathbf{P_{purchase}}$. A selection of traces is detailed below.

$\mathbf{P_{purchase}}$		$\mathbf{P^4_{purchase}}$
$W_1 \overset{C_{11}}{\Leftrightarrow} Task_1, Agent_1$		$W_1 \overset{C_{12}}{\Leftrightarrow} Task_1, Agent_3$
$W_2 \overset{C_{21}}{\Leftrightarrow} Task_2, Agent_2$		$W_2 \overset{C_{22}}{\Leftrightarrow} Task_2, Agent_2$
$W_3 \overset{C_{31}}{\Leftrightarrow} Task_3, Agent_4$		$W_3 \overset{C_{32}}{\Leftrightarrow} Task_3, Agent_4$
$W_4 \overset{C_{41}}{\Leftrightarrow} Task_4, Agent_5$		$W_4 \overset{C_{42}}{\Leftrightarrow} Task_4, Agent_5$
$W_5 \overset{C_{51}}{\Leftrightarrow} Task_5, Agent_2$		$W_5 \overset{C_{52}}{\Leftrightarrow} Task_5, Agent_2$
$W_6 \overset{C_{61}}{\Leftrightarrow} Task_6, Agent_1$		$W_6 \overset{C_{62}}{\Leftrightarrow} Task_6, Agent_3$

and a similar exercise can be performed for all of the alternative traces:

$\mathbf{P_{purchase}}$		$\mathbf{P^4_{purchase}}$
$W_1 \overset{C_{11}}{\Leftrightarrow} Task_1, Agent_1$		$W_1 \overset{C_{12}}{\Leftrightarrow} Task_1, Agent_3$
$W_2 \overset{C_{21}}{\Leftrightarrow} Task_2, Agent_3$		$W_2 \overset{C_{22}}{\Leftrightarrow} Task_2, Agent_2$
$W_3 \overset{C_{31}}{\Leftrightarrow} Task_3, Agent_4$		$W_3 \overset{C_{32}}{\Leftrightarrow} Task_3, Agent_4$
$W_4 \overset{C_{41}}{\Leftrightarrow} Task_4, Agent_5$		$W_4 \overset{C_{42}}{\Leftrightarrow} Task_4, Agent_5$
$W_5 \overset{C_{51}}{\Leftrightarrow} Task_5, Agent_3$		$W_5 \overset{C_{52}}{\Leftrightarrow} Task_5, Agent_2$
$W_6 \overset{C_{61}}{\Leftrightarrow} Task_6, Agent_1$		$W_6 \overset{C_{62}}{\Leftrightarrow} Task_6, Agent_3$

etc. and vice versa and all traces that do not complete such as:

$\mathbf{P_{purchase}}$		$\mathbf{P^4_{purchase}}$
$W_1 \overset{C_{11}}{\Leftrightarrow} Task_1, Agent_4$		$W_1 \overset{C_{12}}{\Leftrightarrow} Task_1, Agent_4$
$W_2 \overset{C_{21}}{\Leftrightarrow} Task_2, Agent_2$		$W_2 \overset{C_{22}}{\Leftrightarrow} Task_2, Agent_2$
$W_3 \overset{C_{31}}{\Leftrightarrow} Task_3, Agent_5$		$W_3 \overset{C_{32}}{\Leftrightarrow} Task_3, Agent_5$
access to $Task_4$ fails		access to $Task_4$ fails

and vice versa and we can conclude that $\mathbf{P_{purchase}} \approx \mathbf{P^4_{purchase}}$ and agent process $Agent_1$ is not critical to the workflow model. As $Agent_1$ is not critical to the workflow then we can say that $Agent_1$ is not identifiable by an external observer from workflow behaviour i.e. $Agent_1$ is anonymous to the external observer.

9.4.1.3 Conclusion

We demonstrate above that $Agent_1$ is not critical to our simple workflow model and we can perform similar experiments for the remaining agents within the model. We find that agent processes $Agent_2$ and

$Agent_3$ are not critical to the simple workflow model but processes $Agent_4$ and $Agent_5$ are critical and identifiable and are not anonymous to an outside observer in this context. That is we can conclude that anonymity is preserved in respect of our model and agents $Agent_1$, $Agent_2$ and $Agent_3$ but we cannot conclude that anonymity is preserved in respect of our model and agents $Agent_4$ and $Agent_5$.

9.4.2 Anonymity tests with a compromised agent

9.4.2.1 Procedure

Let us now consider a scenario wherein one of the agents in our model $\mathbf{P_{purchase}}$ releases its secrets, the agent's key seed and the organisation key seed, to an outside third party. If there are sufficient agents and the model workflow access control protocol is well formed, it should not be possible for the outside third party to recover any information that associates agents, other than the compromised agent, with any task or any portion of a workflow. We say that anonymity is preserved as, whilst the compromised agent and the third party can possibly reason that certain tasks have been performed, the identity of agents attached to tasks cannot be ascertained. In contrast, if the third party is able to reason from available information the identity of an agent associated with a task or a portion of workflow, anonymity is not preserved and the third party may be able to identify an agent target if they wish to increase control over a workflow.

For our example we assume that an agent is compromised: $Agent_2$, say. Now if $\mathbf{P^5_{purchase}}$ is the full purchase workflow model with compromised $Agent_2$ and $\mathbf{P^6_{purchase}}$ is the full purchase workflow model with compromised $Agent_2$ and $Agent_1$ process removed, anonymity is preserved if:

$$\mathbf{P^5_{purchase}} \approx \mathbf{P^6_{purchase}}$$

9.4.2.2 Results

Now we can display a complete successful trace comparison as follows:

$\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}}$		$\mathbf{P}^{\mathbf{6}}_{\mathbf{purchase}}$
$\left\{W_1 \overset{C_{11}}{\Leftrightarrow} Task_1, Agent_1\right\} \wr ID_{Task_1}, pub_{Task_1}$		$\left\{W_1 \overset{C_{12}}{\Leftrightarrow} Task_1, Agent_3\right\} \wr ID_{Task_1}, pub_{Task_1}$
$\left\{W_2 \overset{C_{21}}{\Leftrightarrow} Task_2, Att\,(Agent_2)\right\} \wr ID_{Task_2}, pub_{Task_2}, M_2$		$\left\{W_2 \overset{C_{22}}{\Leftrightarrow} Task_2, Att\,(Agent_2)\right\} \wr ID_{Task_2}, pub_{Task_2}, M_2$
$\left\{W_3 \overset{C_{31}}{\Leftrightarrow} Task_3, Agent_4\right\} \wr ID_{Task_3}, pub_{Task_3}$		$\left\{W_3 \overset{C_{32}}{\Leftrightarrow} Task_3, Agent_4\right\} \wr ID_{Task_3}, pub_{Task_3}$
$\left\{W_4 \overset{C_{41}}{\Leftrightarrow} Task_4, Agent_5\right\} \wr ID_{Task_4}, pub_{Task_4}$		$\left\{W_4 \overset{C_{42}}{\Leftrightarrow} Task_4, Agent_5\right\} \wr ID_{Task_4}, pub_{Task_4}$
$\left\{W_5 \overset{C_{51}}{\Leftrightarrow} Task_5, Att\,(Agent_2)\right\} \wr ID_{Task_5}, pub_{Task_5}, M_5$		$\left\{W_5 \overset{C_{52}}{\Leftrightarrow} Task_5, Att\,(Agent_2)\right\} \wr ID_{Task_5}, pub_{Task_5}, M_5$
$\left\{W_6 \overset{C_{61}}{\Leftrightarrow} Task_6, Agent_1\right\} \wr ID_{Task_6}, pub_{Task_6}$		$\left\{W_6 \overset{C_{62}}{\Leftrightarrow} Task_6, Agent_3\right\} \wr ID_{Task_6}, pub_{Task_6}$

Whilst the above trace comparison is successful trace comparisons can be found for models $\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}}$ and $\mathbf{P}^{\mathbf{6}}_{\mathbf{purchase}}$ that are not successful such as:

$\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}}$		$\mathbf{P}^{\mathbf{6}}_{\mathbf{purchase}}$
$\left\{W_1 \overset{C_{11}}{\Leftrightarrow} Task_1, Agent_1\right\} \wr ID_{Task_1}, pub_{Task_1}$		$\left\{W_1 \overset{C_{12}}{\Leftrightarrow} Task_1, Agent_3\right\} \wr ID_{Task_1}, pub_{Task_1}$
$\left\{W_2 \overset{C_{21}}{\Leftrightarrow} Task_2, Agent_3\right\} \wr ID_{Task_2}, pub_{Task_2}$		No comparable trace — access to $Task_2$ fails
$\left\{W_3 \overset{C_{31}}{\Leftrightarrow} Task_3, Agent_4\right\} \wr ID_{Task_3}, pub_{Task_3}$		
$\left\{W_4 \overset{C_{41}}{\Leftrightarrow} Task_4, Agent_5\right\} \wr ID_{Task_4}, pub_{Task_4}$		
$\left\{W_5 \overset{C_{51}}{\Leftrightarrow} Task_5, Agent_3\right\} \wr ID_{Task_5}, pub_{Task_5}$		
$\left\{W_6 \overset{C_{61}}{\Leftrightarrow} Task_6, Agent_1\right\} \wr ID_{Task_6}, pub_{Task_6}$		

This failure occurs because:

If $Agent_3$ accesses $Task_1$ within $\mathbf{P}^{\mathbf{6}}_{\mathbf{purchase}}$, either $Agent_2$ or $Agent_5$ must access $Task_2$. However, if $Agent_5$ accesses $Task_2$, it cannot access $Task_3$ or $Task_4$ and there are insufficient agents to access both of tasks $Task_3$ and $Task_4$. If $Agent_2$ accesses $Task_2$, static equivalence fails as $Agent_2$ is compromised and an attacker obtains message M_2 but if $Agent_3$ accesses $Task_2$ under $\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}}$, there is no release of M_2 within the trace of $\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}}$ so observational equivalence fails.

If the model includes an additional agent that also has access to task $Task_1$ and that agent is redundant elsewhere throughout the model, the failure of observational equivalence no longer occurs as agent $Agent_3$ is available to access $Task_2$ and all subsequent agent/task assignments can occur in the normal way.

Given this failure we can conclude that:

$$\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}} \not\approx \mathbf{P}^{\mathbf{6}}_{\mathbf{purchase}}$$

Now let us consider performing the same experiment only we reinstate agent $Agent_1$ and remove $Agent_3$ instead.

Now, $\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}}$ is the same as before but this time we compare it with $\mathbf{P}^{\mathbf{7}}_{\mathbf{purchase}}$ being $\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}}$ less $Agent_3$. Again we can find equivalent traces such as:

$\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}}$		$\mathbf{P}^{\mathbf{7}}_{\mathbf{purchase}}$
$\left\{W_1 \overset{C_{11}}{\Leftrightarrow} Task_1, Agent_3\right\} \wr ID_{Task_1}, pub_{Task_1}$		$\left\{W_1 \overset{C_{12}}{\Leftrightarrow} Task_1, Agent_1\right\} \wr ID_{Task_1}, pub_{Task_1}$
$\left\{W_2 \overset{C_{21}}{\Leftrightarrow} Task_2, Att\,(Agent_2)\right\} \wr ID_{Task_2}, pub_{Task_2}, M_2$		$\left\{W_2 \overset{C_{22}}{\Leftrightarrow} Task_2, Att\,(Agent_2)\right\} \wr ID_{Task_2}, pub_{Task_2}, M_2$
$\left\{W_3 \overset{C_{31}}{\Leftrightarrow} Task_3, Agent_4\right\} \wr ID_{Task_3}, pub_{Task_3}$		$\left\{W_3 \overset{C_{32}}{\Leftrightarrow} Task_3, Agent_4\right\} \wr ID_{Task_3}, pub_{Task_3}$
$\left\{W_4 \overset{C_{41}}{\Leftrightarrow} Task_4, Agent_5\right\} \wr ID_{Task_4}, pub_{Task_4}$		$\left\{W_4 \overset{C_{42}}{\Leftrightarrow} Task_4, Agent_5\right\} \wr ID_{Task_4}, pub_{Task_4}$
$\left\{W_5 \overset{C_{51}}{\Leftrightarrow} Task_5, Att\,(Agent_2)\right\} \wr ID_{Task_5}, pub_{Task_5}, M_5$		$\left\{W_5 \overset{C_{52}}{\Leftrightarrow} Task_5, Att\,(Agent_2)\right\} \wr ID_{Task_5}, pub_{Task_5}, M_5$
$\left\{W_6 \overset{C_{61}}{\Leftrightarrow} Task_6, Agent_3\right\} \wr ID_{Task_6}, pub_{Task_6}$		$\left\{W_6 \overset{C_{62}}{\Leftrightarrow} Task_6, Agent_1\right\} \wr ID_{Task_6}, pub_{Task_6}$

But now we obtain the following failed traces:

$\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}}$		$\mathbf{P}^{\mathbf{7}}_{\mathbf{purchase}}$
$\left\{W_1 \overset{C_{11}}{\Leftrightarrow} Task_1, Att\,(Agent_2)\right\} \wr ID_{Task_1}, pub_{Task_1}, M_1$		$\left\{W_1 \overset{C_{12}}{\Leftrightarrow} Task_1, Att\,(Agent_2)\right\} \wr ID_{Task_1}, pub_{Task_1}, M_1$
$\left\{W_2 \overset{C_{21}}{\Leftrightarrow} Task_2, Agent_3\right\} \wr ID_{Task_2}, pub_{Task_2}$		No comparable trace — access to $Task_2$ fails
$\left\{W_3 \overset{C_{31}}{\Leftrightarrow} Task_3, Agent_4\right\} \wr ID_{Task_3}, pub_{Task_3}$		
$\left\{W_4 \overset{C_{41}}{\Leftrightarrow} Task_4, Agent_5\right\} \wr ID_{Task_4}, pub_{Task_4}$		
$\left\{W_5 \overset{C_{51}}{\Leftrightarrow} Task_5, Agent_3\right\} \wr ID_{Task_5}, pub_{Task_5}$		
$\left\{W_6 \overset{C_{61}}{\Leftrightarrow} Task_6, Att\,(Agent_2)\right\} \wr ID_{Task_6}, pub_{Task_6}, M_6$		

This failure occurs because if $Agent_2$ accesses $Task_1$, either $Agent_3$ or $Agent_5$ must access $Task_2$. However, $Agent_3$ is not present in $\mathbf{P}^{\mathbf{7}}_{\mathbf{purchase}}$ and if $Agent_5$ accesses $Task_2$, $Agent_5$ can no longer access $Task_3$ or $Task_4$ and the workflow cannot complete. Given this failure we can conclude that:

$$\mathbf{P}^{\mathbf{5}}_{\mathbf{purchase}} \not\approx \mathbf{P}^{\mathbf{7}}_{\mathbf{purchase}}$$

Alternatively:

$\mathbf{P^5_{purchase}}$		$\mathbf{P^7_{purchase}}$
$\left\{W_1 \overset{C_{11}}{\Leftrightarrow} Task_1, Agent_1\right\} \wr ID_{Task_1}, pub_{Task_1}$		$\left\{W_1 \overset{C_{12}}{\Leftrightarrow} Task_1, Agent_1\right\} \wr ID_{Task_1}, pub_{Task_1}$
$\left\{W_2 \overset{C_{21}}{\Leftrightarrow} Task_2, Agent_3\right\} \wr ID_{Task_2}, pub_{Task_2}$		No comparable trace — access to $Task_2$ fails
$\left\{W_3 \overset{C_{31}}{\Leftrightarrow} Task_3, Agent_4\right\} \wr ID_{Task_3}, pub_{Task_3}$		
$\left\{W_4 \overset{C_{41}}{\Leftrightarrow} Task_4, Agent_5\right\} \wr ID_{Task_4}, pub_{Task_4}$		
$\left\{W_5 \overset{C_{51}}{\Leftrightarrow} Task_5, Att\left(Agent_2\right)\right\} \wr ID_{Task_5}, pub_{Task_5}, M_1$		
$\left\{W_6 \overset{C_{61}}{\Leftrightarrow} Task_6, Agent_1\right\} \wr ID_{Task_6}, pub_{Task_6}$		

Now comparison failure occurs because either $Agent_2$ or $Agent_5$ must access $Task_2$ in $\mathbf{P^7_{purchase}}$ but although a successful trace can be obtained with $Agent_2$ this fails static equivalence when compared to $Agent_3$ accessing $Task_2$ and we have already seen that the workflow fails if $Agent_5$ accesses $Task_2$.

Currently, our tests do not differentiate further than to say that both $Agent_1$ and $Agent_3$ can be linked with $Task_1$ and $Task_2$. However, if we look at the trace comparison above for $\mathbf{P^5_{purchase}}$ and $\mathbf{P^7_{purchase}}$ with $Agent_2$ assigned to $Task_1$, we observe that $Agent_3$ must be the only agent associated with access to $Task_2$ and completion of the workflow, in addition to our compromised agent $Agent_2$. So $Agent_3$ can be identified by its association with $Task_2$. In respect of the trace comparison between $\mathbf{P^5_{purchase}}$ and $\mathbf{P^6_{purchase}}$ then trace failure occurs because we have to use $Agent_2$ with either $Task_1$ or $Task_2$ in order to get a successfully completed workflow trace in $\mathbf{P^6_{purchase}}$. This is not comparable to a trace utilising $Agent_1$ and $Agent_3$ with tasks $Task_1$ or $Task_2$ in $\mathbf{P^5_{purchase}}$. We can conclude therefore that $Agent_1$ must be associated with $Task_1$ or $Task_2$ and completion of the workflow if $Agent_2$ has not accessed either task.

The primary purpose of the anonymity test is to confirm anonymity for a given access control model and any equivalence failure may or may not reflect an exploitable real world example of an anonymity breach. The amount of information obtainable from failure of this anonymity test for different models with different numbers of agents and tasks and different access control constraints will vary over the models. Consequently, we have to be careful in our assessment of the test results as it is easy to overstate real world conclusions based upon failed model traces. When we analyse failed traces we have to bear in mind that the compromised agent or agents and the external observer do not necessarily have a modeller's eye view of the access control procedure. Consequently, in the case of $\mathbf{P^5_{purchase}}$ and $\mathbf{P^6_{purchase}}$ for example, the modeller knows that $Agent_1$ can access $Task_1$ and not $Task_2$ but our compromised agent $Agent_2$ and our outside observer do not necessarily know this a priori.

9.4.2.3 Conclusion

We demonstrate with this experiment that, given a single compromised agent, $Agent_2$ in this case, we cannot conclude that anonymity is preserved for our simple purchases model in respect of agent $Agent_1$, nor can we conclude that anonymity is preserved for our purchases model in respect of $Agent_3$. In the particular experimental cases discussed above it is possible to reason as to how the breach of anonymity occurred and to interpret this breach in the context of a real-world implementation of the model. However, it might not always be the case that breaches of anonymity can necessarily be linked to a real-world context. Our anonymity test proves or otherwise anonymity of agent processes within workflow models but it does not necessarily prove that an exploitable anonymity breach has occurred on failure of the test.

9.5 Summary

In this chapter we apply the security tests that we introduce in chapter 8 to the simple purchases workflow introduced in chapter 4 and encoded in chapter 7. Our security tests confirm that our purchases model is satisfiable given the available set of tasks and agents per section 9.2. We identify a collusion metric of four for our model as per the experiments of section 9.3, so at least four different agents are required to gain complete control over workflow resources. Whilst we cannot make specific judgements concerning the level of security suggested by the collusion metric we can say that, in the context of a small organisation, a metric of four reflects a reasonable application of separation of duties to the purchase workflow. We would generally consider this to be a high level of security in the circumstances. Finally, we perform anonymity experiments in section 9.4 and we identify some examples of anonymity of agents in the context of our purchases workflow. Additionally, we identify some examples of possible breaches of anonymity for agents and we discuss how such potential breaches might be interpreted given that our test is a test for anonymity and not a test for breaches of anonymity.

Chapter 10

Discussion and conclusions

10.1 Discussion

Resource management and access control have become increasingly important and complex components of an organisation's activities. In part this is due to the regulatory framework associated with accountability and good corporate governance, protection of the individual, security and taxation that applies to companies, charitable institutions and governmental bodies. Also, this reflects the fact that there is a multitude of threat vectors arising from widespread information access and increasingly sophisticated attack mechanisms. When combined with powerful motivators for humans to behave dishonestly, such as reduced job security, ideologies and greed, then the need for robust systems of access control within organisations is clear.

The strategic management of access control to resources is provided by business rules. These rules have long been accepted as best able to confront the problems of resource management by major organisations such as the accountancy bodies in the UK [8] and elsewhere and although there have been (and will continue to be) debates over the efficacy of these rules they are the best approach to resource management that is available to us at present.

The primary motivation for our thesis is that business rules, particularly segregation of duties, are not handled satisfactorily by access control systems. Given this premise then the consequences of the failure to apply business rules properly are also not addressed when seeking to verify the adequacy or otherwise of access control policies within organisations.

We identify workflow-based access control as a mechanism whereby segregation of duties can be properly implemented, particularly when combined with a means to memorise access control state. Researchers

have addressed some of the issues regarding workflow-based access control systems such as policy language development. Bertino et al [20, 19], Botha [25, 26, 27] and Wainer et al [98, 99, 100] have all addressed this aspect of workflow-based access control and in the case of Wainer et al the definition and application of static and dynamic segregation of duties are incorporated within the policy language. However, none of these has sought to address the questions concerning the successful application of these policy languages and the security consequences that entail from their failure. There is no attempt within these publications to systematically identify and analyse the consequences of the failure to adequately apply segregation of duties and other business rules or to identify the potential for inconsistencies within access control policies. We argue that the formulation of access control policies requires not only a framework, policy language and prototyping but also tools that enable access control policies to be tested in respect of a selection of potential security concerns, such as collusion and the leaking of information to untrusted third parties, that arise from failure of these policies.

Crampton has successfully extended the research into the modelling and analysis of workflow-based access control systems. He has concentrated on model testing for reachability properties, such as satisfiability and has not extended the testing to security aspects arising from the failure of access control policies [31, 32]. Other researchers have investigated access control modelling and security testing such as Zhang and Ryan [105] and Qunoo and Ryan [78]. Whilst this research has yielded strong model-based testing of security issues within RBAC, these approaches have not addressed access control from a workflow perspective and have not generated tests of security issues arising from the inadequate application of business rules.

Our thesis proposal, Business Process Access Control (BPAC), addresses the modelling of workflow-based systems of access control and focuses upon the analysis and testing of security issues arising from the inadequate application of business rules, particularly segregation of duties, whilst also ensuring that the more traditional tests for access control systems, such as reachability-based tests, can also be applied within the modelling environment.

10.2 Security testing of workflow-based access control policies

We identify a number of security tests that can be applied to workflow-based access control systems. These include basic reachability tests such as satisfiability and completeness, a simple quantifiable test for the susceptibility of workflows to complete control by some number of agents, which we call the collusion metric and an analysis of anonymity of agents and agent/task interactions within workflows. The latter two tests, the collusion metric and the anonymity test are, to our knowledge, unique to

our thesis with regard to the published work on access control. However, these tests are based upon established analytical principles within process algebras such as refinement in CSP [80] and observational equivalence in pi calculus and its derivatives [66, 67, 3].

10.2.1 Reachability tests — satisfiability and completeness

These tests are, at their simplest, verification techniques that ensure that a workflow, when coupled with an access control policy and a set of agents, can operate correctly. In this context, it could be argued that our approach adds little to the work of Crampton and others in respect of workflow-based access control analysis. However, we extend our tests to enable us to reveal certain security properties as, for example, in our analysis of a simple conference management system in section 8.3.2 wherein we are able to highlight an agent access flaw by reachability methods.

10.2.2 The collusion metric

The collusion metric represents an extension of the reachability tests summarised above. It is a simple numerical guide as to the robustness of a workflow, achieved through the implementation of segregation of duties, against complete control over the workflow being obtained by a set of agents and/or an external third party. We observe that control over resources can be obtained by indirect means, such as via communication between agents, and we extend the collusion metric to reflect this possibility.

The collusion metric provides a reasonable initial estimate of workflow robustness but it has limitations. Firstly, the estimation of the collusion metric requires multiple satisfiability tests over various subsets of agents. It is, therefore, likely to be a long-winded exercise when workflow models contain complex workflows and large numbers of participating agents. This problem can be mitigated to an extent by efficiencies in the agent selection process. Secondly, complex workflows with non-deterministic choice have multiple collusion metrics depending upon the satisfiable paths through the workflow. Thirdly, the collusion metric assigns no weight to differing agent roles. Consequently, a director of an organisation is assigned equal importance to an employee within the context of workflow control. In practice, attempting to value agent contributions differently is fraught with difficulty. It can be argued that senior agents are more responsible and reliable and therefore less risky than junior agents. Alternatively, senior agents are a higher security risk than junior agents because they have greater responsibility and access to a wider range of organisational resources.

10.2.3 Anonymity

We introduce the concept of anonymity in respect of workflow-based access control systems. The basic idea is that anonymity is preserved if an agent (or an untrusted third party via a compromised agent) is unable to ascertain which agent or agents has accessed tasks within a workflow. For a workflow such as a purchase order to operate effectively an agent need only be concerned that previous tasks have been completed prior to the completion of her own task. For instance, a purchaser can proceed with a purchase if she has evidence that the transaction has been approved but she does not necessarily need to know who approved the transaction. If the agent is able to deduce the identity of agents accessing prior tasks, that agent (especially if she has been compromised by an external third party) has narrowed the search for potential collusion candidates in respect of the workflow as a whole. Testing for anonymity cannot be achieved simply through the use of reachability methods and requires the power of observational equivalence within our BPAC modelling environment.

10.3 The applied pi calculus as the basis for the BPAC environment

When considering the basis for the BPAC modelling environment we focused upon workflow-friendly possibilities such as Petri nets and process calculi. We eliminated Petri nets as an option as, although they naturally model workflow transitions and can provide a suitable graphical means of establishing satisfiability of workflows, they are limited in a number of respects. In particular, Petri nets cannot satisfactorily handle the interaction of complex data components that comprise constraints-based access-control. Being closed systems they do not handle real-world interactions well, e.g. with a pool of agents, especially when there is uncertainty and non-determinism in respect of these interactions. Further problems arise when attempting to model complex or multiple interacting workflows, hierarchies and reuseable processes where Petri net methods are unwieldy.

Given the above, our attention turned to process calculi and it quickly became apparent that a process calculus approach would be more appropriate to our requirements. We decided upon the applied pi calculus of Abadi and Fournet [3] because it features the theoretical and expressive power of Milner's original pi calculus [66, 67] coupled with the additional ability to handle functions with an equational theory and to differentiate names from variables. We adopted the pi calculus handling of workflows of Puhlmann and Weske [77] and through the use of function-based authentication modelling, as per Abadi et al [3, 4], we were able to develop a secure interactive protocol that forms the basic building block for

the implementation of our BPAC modelling environment.

One of the limitations of our use of the applied pi calculus for our BPAC modelling environment is that the detailed code is somewhat impenetrable and difficult to both read and write. Whilst we have attempted to simplify matters through the use of syntactic sugar wherever appropriate nevertheless the encoded workflow models can become complex and unwieldy very quickly. Also, some aspects of programming models within applied pi calculus demonstrate considerable subtlety, which is not necessarily obvious to the casual observer. However, it was always envisaged that the modelling of access control would behave as an adjunct to the problem of business process management (BPM) as a whole. As a consequence of the isomorphism between workflows for access control and workflows for BPM, a visual approach incorporating access control modelling within BPM would likely occur in practice.

10.4 Support processes for the BPAC modelling environment

An important aspect of the development of the BPAC modelling environment was the requirement for some systematic approach to memorising access control state within an access control model. In order that we can properly model dynamic segregation of duties we considered it necessary to develop some means whereby answers to such questions as "who has accessed the previous task?" or "is an agent currently accessing another workflow?" are made available within the modelling environment. To this end we devised applied pi calculus processes called stores, which could be used to retain access state information such as "agent A_1 has accessed task T_1 in workflow W_1". Such information can be represented in predicate form as:

$$hasAccessed(ID_{A_1}, ID_{T_1}, ID_{W_1})$$

and we model this in the applied pi calculus by a one-way data function that has a similar form to the above:

$$hasAccessed(X, Y, Z)$$

where X, Y and Z are any terms within the model. Given this mechanism we were then able to develop a constraint test for processes that could construct a data function from input information and pattern match it against the store contents. In order to formulate this approach to constraint testing we had to devise set handling processes within the applied pi calculus so that we could create ongoing stores that

could be tested against, written to and have elements deleted. Strictly speaking, our approach creates and uses ordered lists as opposed to sets but the lists behave sufficiently like sets for our purpose.

Having developed the stores we then devised a compact syntax for handling store information: writing, deleting and testing against the store. Whilst this store approach to access state management works for manual processing of BPAC models it might not translate well to automated theorem proving software, such as ProVerif. Currently, ProVerif has no mechanism for memorising state and the ongoing processing of the store and support processes is potentially very resource hungry. More efficient approaches to the store problem may well be possible that circumvent this particular problem.

10.5 Building blocks for access control modelling — the standard core protocol

Using the applied pi calculus we developed standard processes for the three main participants in our implementation of workflow-based access control namely the workflow manager, the agent and the task. The interaction between these participants is called the standard core protocol and complex workflows can be built from multiples of the standard core protocol connected by Puhlmann and Weske trigger functions to form the workflow pattern [77].

10.6 Observational equivalence and the abstraction test for security testing

In order that we could properly implement the security tests identified above we decided to utilise the power of observational equivalence/labelled bisimilarity within the applied pi calculus. It became apparent, however, that observational equivalence was too powerful and process intensive for the simpler reachability-based tests and so we devised a lightweight unidirectional variant of observational equivalence called the abstraction test, which is more suited to reachability type problems. The principle requirement of the abstraction test is the generation of a simplified abstract workflow model in respect of message outputs from tasks. We then perform trace experiments of the full workflow model in an attempt to identify the existence of a trace that behaves as the abstract model. This test is very flexible as it is possible to devise all manner of target abstract workflows. However, the test can involve considerable subtlety in respect of the generation of the abstract models and there is the problem of computational explosion with complex workflows and numbers of agents.

Utilising the abstraction test we were able to develop tests for satisfiability and computation of the collusion metric. Also we devised a reachability test based upon the abstraction test that enabled us to identify the security flaw in the simple conference management problem mentioned above.

We applied the full observational equivalence to our tests for anonymity as the abstraction test was not sufficiently powerful for this security test. The basic principle of the test is to compare two variations of the same workflow model that differ by the presence or absence of a single agent. If there is full observational equivalence between the two variations, anonymity is preserved as the agent cannot be exclusively associated with one or more tasks within the workflow. We extended the test in our examples to recognise the contribution of a compromised agent to anonymity within the workflow model and we demonstrated both successful and unsuccessful anonymity proofs with our examples. It was noted that our anonymity test is a proof for anonymity and that failure of proof did not of itself indicate that anonymity could be compromised. It is important to stress, therefore, that care be taken when interpreting failed anonymity tests as there is no guarantee that the failure might correspond to a real-world example of anonymity compromise. Also, as in the case of the the abstraction test, there is a potential issue with computational explosion with complex workflows and multiple agents and tasks.

10.7 Summary

The key proposal of this thesis is the BPAC modelling environment as a tool for analysing workflow-based access control models. The summary of contributions is presented below:

- We use the applied pi calculus as the basis for our modelling environment.
- We extended the syntax of the applied pi calculus for the purpose of our modelling environment.
- We developed processes in the applied pi calculus for handling sets.
- We developed store processes from the set handling processes so that we can properly handle access control state within our environment.
- We propose a lightweight, one-way variation of observational equivalence, which we call the abstraction test, for reachability-based testing of workflow models.
- We identify business rules, particularly segregation of duties, as an aspect of access control that in our opinion has not been adequately addressed in the research literature.
- We identify workflow-based access control, when coupled with memory for access control state, as a suitable mechanism for properly implementing segregation of duties.

- We identify specific security issues arising from the failure to properly implement segregation of duties, in particular collusion and anonymity.
- We present tests for these security issues. The collusion metric is a measure of the susceptibility of a workflow to total control by agents and external third parties. Anonymity is a test of the inability of external third parties to identify agents and their interactions with tasks in workflows.
- We apply the abstraction test and observational equivalence to tests for the security issues identified above.
- We provide examples of encoding of workflow-based access control scenarios in our modelling environment and we demonstrate the application of the security tests to these models.

10.8 Future work

We were unable to transfer the BPAC modelling environment to an automated verifier tool, such as ProVerif, because of its limitations with the handling of state and the lack of completeness of the observational equivalence implementation. Transfer to an automated verifier, therefore, requires adaptation of existing tools or the creation of a new computer-based tool for this purpose.

There are numerous inefficiencies within the BPAC modelling environment that could be addressed in order to produce a more useable tool. The coding of the standard core protocol components could undoubtedly be simplified and the processes involved in security testing need to be optimised if a practical implementation is to be achieved.

Whilst the potential links between the BPAC modelling environment and BPM is discussed in this thesis no attempt has been made to date to formally connect the modelling environment to examples of BPM systems. At the very least a visual analysis tool is required for a practical implementation of the modelling environment.

We have devised and demonstrated a selection of security tests, primarily in respect of reachability analysis and the implementation of segregation of duties. No doubt there is scope for additional testing of security properties that have not been addressed within this thesis.

The collusion metric is a very basic first attempt at providing a measure for collusion. However, as we discussed above, this measure is somewhat limited and a more meaningful measure could possibly be devised that better reflects the security properties of workflows.

We have not sought to introduce external contributions to constraint handling such as timing or other factors that may influence access control and the incorporation of these factors would add to the practical functionality of the modelling environment.

Appendix A

The standard core protocol in detail

A.1 Introduction

In chapter 6 we outline the broad structure of the applied pi calculus implementation of BPAC. We indicate that the modelling environment is based upon encrypted coding of the key components of BPAC: the workflow manager, agents and tasks that interact with each other in a specific way called the standard core protocol. In this appendix we present the detailed coding of the core components of the modelling system. In the first section we present the unencrypted protocol so as to highlight the salient features of the access control modelling environment and then in the second section we present the full encrypted version of the protocol.

A.2 The basic access control protocol unencrypted

We consider the basic core protocol, which we refer to as C, as representing the interaction between the workflow manager, W, comprising a single $taskCall\,\langle\langle ID_T, K_T, ID_W, K_W\,[]\rangle\rangle$ context, single agent, A and task T processes as indicated in the visualisation in figure 6.1 above. All complex workflow-based interactions can be constructed from the basic protocol. The basic protocol is visualised in its unencrypted form in figure A.1. In summary, the protocol proceeds as follows:

- The workflow manager process W initiates a session of task process T by sending a fresh identity for the task, ID_T together with a workflow session identity ID_W to the task process T.
- The task process acknowledges receipt of ID_T by returning same to the workflow process.

- The workflow manager process advertises the task session by broadcasting ID_T.
- An agent process A receives ID_T and returns it with it's agent identity ID_A to the workflow process so as to signify an interest in accessing task T.
- The workflow manager tests ID_A against history stores. This process is the model enactment of the workflow-based access control policy. Depending upon the outcome of the tests then the workflow manager process continues or halts.
- The workflow manager sends ID_A, ID_T and a freshly generated nonce n_T to the task process T. This signifies to the task process that agent process A has been cleared to communicate with task process T.
- The workflow manager process sends ID_T and the nonce n_T to the agent process A. This signifies to the agent process that it has been cleared to communicate with task process T.
- The agent process A sends ID_A, ID_T and the nonce n_T to the task process T to initiate authorised communication between A and T.
- The task process T sends secret M_T together with ID_T to the agent process A. We use M_T to represent the resources associated with task T so that A having M_T effectively means that A has control over T.
- Agent process A sends its acquired secret, together with a fresh nonce to itself for further communication.
- The agent process A sends data function $end(M_T, ID_T)$ to the task process T to indicate that processing of task resources has been completed.
- The task process T writes to the stores data function

  $hasAccessed(ID_A, ID_T, \ldots)$

  (the data function should take ID_A and ID_T as arguments at the very least) to record that agent A has accessed the instance of task T identified by ID_T.
- Task process T sends the data function $endTask(ID_T)$ to the workflow manager process W to indicate that the task session has been completed.

Whilst this protocol provides suitable interaction between the three parties, namely the workflow manager, tasks and agents, so as to represent the management of access control with static and dynamic constraints,

we also need to recreate a secure space so that this protocol can operate as if it is within the confines of an organisation from the perspective of an outside observer. We can do this either by ensuring that all communication takes place over private channels or we can adopt basic cryptography over public channels. We choose to use the latter approach.

A.3 The case for encryption of the access control protocol

Whilst a level of trust between organisations and their employees or agents must exist if employees are to make a meaningful contribution towards an organisation's activities, nevertheless no employee or agent, however senior, can be totally trusted as the potential cost to an organisation of their defection is potentially too great to countenance.

Separation of duties is implemented within organisations to ensure that agents have limited resource access scope within an organisation and the cost of defection of a single agent is therefore constrained. Ordinarily, agents should not be able to see the details of communications between other agents, workflow managers and tasks. We want to be able to model what happens when an agent turns over to the 'dark side' and in particular, we want to consider this in respect of the limited access scope of the turned agent coupled with whatever extra information is made available to her via intra-agent communication, tasks performed etc. Also, we want to perform tests on these workflows, such as tests for anonymity, that consider the impact of a compromised agent on the information released to external third parties (a Dolev-Yao attacker). Consequently, we model the core protocol using encrypted communication over a public channel. Such an approach to our modelling environment ensures that an operational scope can be implemented over agents and their interactions with tasks and resources as mediated by the systems of access control. Our investigations can then reveal answers to the following queries:

- given the constraints can workflows be performed properly?
- Can an agent secure complete control over a workflow or workflows and their associated resources?
- If a single agent cannot secure complete control over a workflow and its resources, what is the minimum number of agents, given the constraints, that can collectively secure complete control over a workflow and its resources. Is this minimum number acceptable to an organisation?
- Can a compromised agent identify and provide information about the activities of other agents to an external third party that could assist with obtaining complete control over a workflow and associated resources?

In the next section we present the basic protocol with the addition of encryption.

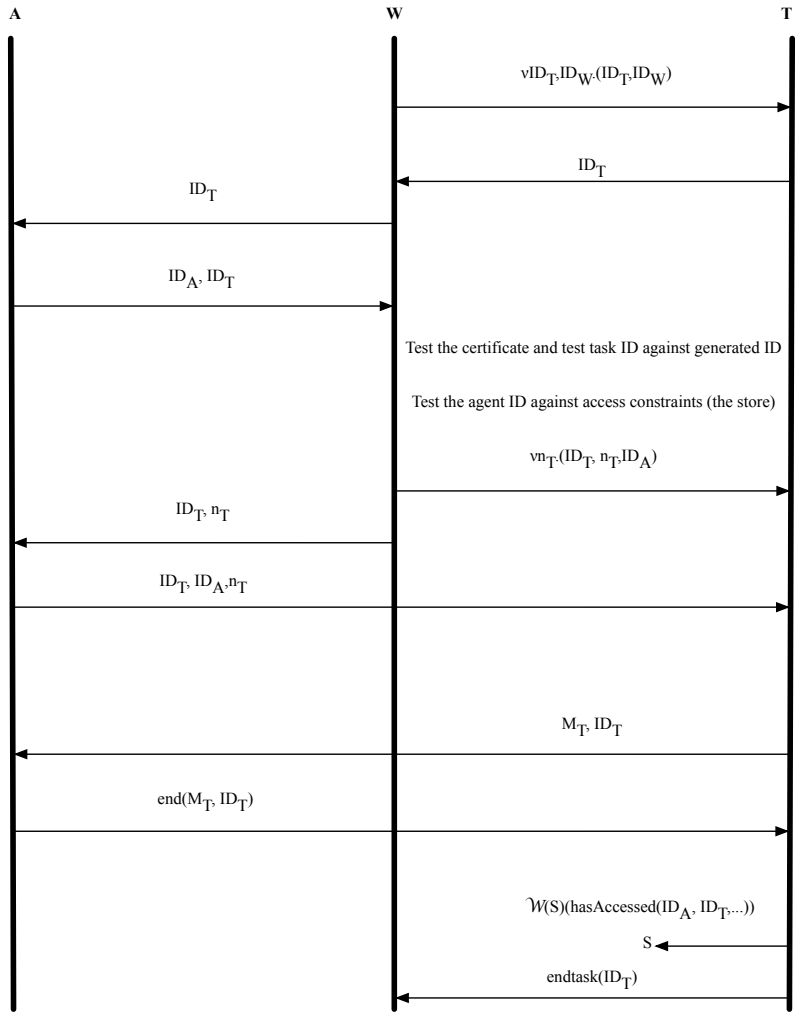


Figure A.1: the basic protocol unencrypted

A.4 The basic protocol with encryption

As in section A.1 above we consider the basic core protocol C as representing the interaction between the workflow manager W, single agent A and task T processes as indicated in the visualisation in figure 6.1. All complex workflow-based interactions can be constructed from the basic protocol. The basic protocol is visualised in figure A.2 and the detailed coding of the protocol is presented in section A.5. The protocol makes extensive use of public key encryption and decryption as communication between W, A and T is carried out over the public channel and we require messages to remain secret. The encryption process is given in the protocol diagram by $enc\,(M, pub)$, where M is an unencrypted message, pub is the public key of the recipient. Decryption is given by $dec\,(N, pte)$, where N is the encrypted message and pte is the private key of the recipient. Additionally, we use signature encryption and decryption denoted by $encs\,(M, pte)$ and $decs\,(N, pub)$ to provide verification of the identity of the message generator to the recipient.

The protocol as visualised can be summarised as follows:

- The workflow manager W initiates a new task session with a fresh task identity, ID_T, sent signed and encrypted to task T.
- On receipt of an encrypted and signed acknowledgement from the new task instance the workflow manager then advertises the task identity to agents as a message ID_T, source verifiable through the accompanying signature and encrypted using an organisation-wide public key, pub_{Org}. The organisation-wide encryption is a low grade security component: there is not a great security risk to the information being sent in plaintext over the public channel. The encryption simply provides a perimeter to the organisation's activities.
- An agent picks up the task advertisement and sends its identity ID_A, matched with the task identity ID_T, signed and encrypted to the workflow manager.
- On receipt of a response from an agent the workflow manager tests the agent identity ID_A against sets of constraints held in the stores. These constraints might include the agent/role assignment and other passive access control rules and dynamic constraints such as agent access to previous tasks.
- The workflow manager sends the agent identity ID_A coupled with the task session identity ID_T and a freshly generated nonce n_T to the task process signed and encrypted. The nonce provides protection against replay attacks.

- The workflow manager passes the same nonce and the task session identity to the agent, signed and encrypted.
- The agent now sends the agent identity ID_A, the task identity ID_T and the nonce n_T to the task process, signed and encrypted.
- The task process passes its resource message, M_T, coupled with the task identity ID_T to the agent, signed and encrypted.
- The agent process sends its acquired resource message M_A, together with its own generated nonce n_{nonce}, encrypted by its own public key over the public channel t as a potential communication for leaking information to other processes, particularly agents. The nonce ensures that this communication is unique over multiple instances of the core process C.
- The agent process passes an $end()$ constraint term to the task process containing such arguments as the agent identity ID_A and the task identity ID_T.
- The task process writes a $hasAccessed()$ constraint term to a store containing such arguments as the agent identity ID_A and the task identity ID_T.
- The task sends an $endTask()$ constraint term to the workflow manager containing the task identity ID_T, signed and encrypted, to signal completion of the communication.

Now we turn our attention to the detailed encoding of the access control protocol.

A.5 Detailed encoding of the core processes for the basic protocol

In the following subsections we present the detailed coding of the core processes that provide the building blocks for access control modelling.

A.5.1 Task

We consider a core task process as comprising the following context code with a pair of holes that can be populated with store communications or the internal process τ as appropriate. We use the syntactic sugar $task\langle\langle K_T, M_T, [\,], [\,]\rangle\rangle$ to represent this context:

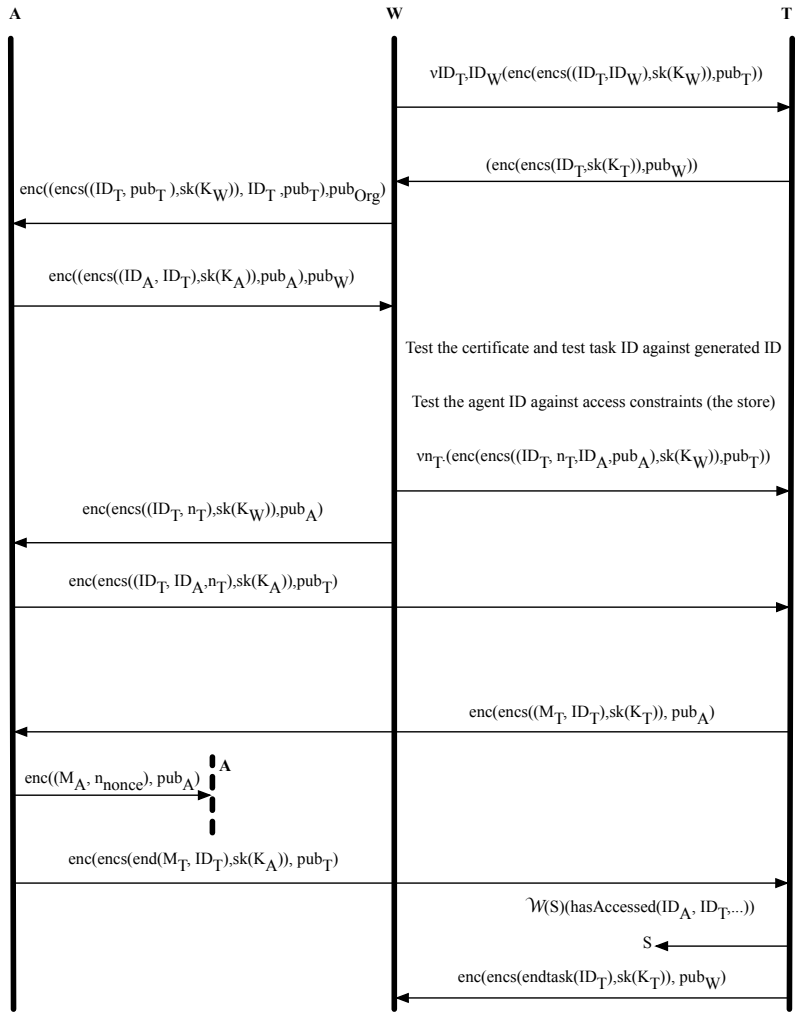


Figure A.2: Workflow-based access control — the basic protocol

$$
\begin{aligned}
& t\left(twenc_T\right).\textbf{let } tdec_T = dec\left(twenc_T, sk\left(K_T\right)\right) \textbf{ in} \\
& \textbf{let } id_T = fst\left(decs\left(tdec_T, pub_W\right)\right) \textbf{ in} \\
& \textbf{let } id_W = snd\left(decs\left(tdec_T, pub_W\right)\right) \textbf{ in} \\
& \textbf{let } tenc_T = encs\left(id_T, sk\left(K_T\right)\right) \textbf{ in} \\
& \textbf{let } wtenc_T = enc\left(tenc_T, pub_W\right) \textbf{ in} \\
& \overline{t}\left\langle wtenc_T\right\rangle.t\left(twenc_{T,n,A,pub_A}\right) \\
& .\textbf{if } fst\left(decs\left(dec\left(twenc_{T,n,A,pub_A}, sk\left(K_T\right)\right), pub_W\right)\right) = id_T \textbf{ then} \\
& \textbf{let } n = snd\left(decs\left(dec\left(twenc_{T,n,A,pub_A}, sk\left(K_T\right)\right), pub_W\right)\right) \textbf{ in} \\
& \textbf{let } id_A = thd\left(decs\left(dec\left(twenc_{T,n,A,pub_A}, sk\left(K_T\right)\right), pub_W\right)\right) \textbf{ in} \\
& \textbf{let } pub_A = fth\left(decs\left(dec\left(twenc_{T,n,A,pub_A}, sk\left(K_T\right)\right), pub_W\right)\right) \textbf{ in} \\
& t\left(taenc_{T,A,n}\right).\textbf{if } fst\left(decs\left(dec\left(taenc_{T,A,n}, sk\left(K_T\right)\right), pub_A\right)\right) = id_T \textbf{ then} \\
& \textbf{if } snd\left(decs\left(dec\left(taenc_{T,A,n}, sk\left(K_T\right)\right), pub_A\right)\right) = id_A \textbf{ then} \\
& \textbf{if } thd\left(decs\left(dec\left(taenc_{T,A,n}, sk\left(K_T\right)\right), pub_A\right)\right) = n \textbf{ then} \\
& [] \\
& .\left(\textbf{let } tenc_{M_T,T} = encs\left(\left(M_T, id_T\right), sk\left(K_T\right)\right) \textbf{ in} \right. \\
& \textbf{let } atenc_{M_T,T} = enc\left(tenc_{M_T,T}, pub_A\right) \textbf{ in} \\
& \overline{t}\left\langle atenc_{M_T,T}\right\rangle.t\left(taenc_{end,M_A,T}\right) \\
& .\textbf{if } decs\left(dec\left(taenc_{end,M_A,T}, sk\left(K_T\right)\right), pub_A\right) = end\left(M_T, id_T\right) \textbf{ then} \\
& [] \\
& .\mathcal{W}\left(S\right)\left(hasAccessed\left(id_A, id_T, id_W\right)\right) \\
& .\textbf{let } tenc_{end,M_T,T} = encs\left(endtask\left(id_T\right), sk\left(K_T\right)\right) \textbf{ in} \\
& \textbf{let } wtenc_{end,M_T,T} = enc\left(tenc_{end,M_T,T}, pub_W\right) \textbf{ in} \\
& \left.\overline{t}\left\langle wtenc_{end,M_T,T}\right\rangle\right)
\end{aligned}
$$

Based on the above a task T_a without additional user-generated code would be defined as follows:

$$T_a \quad \triangleq \quad \nu K_{T_a}.task\left\langle\left\langle K_{T_a}, M_{T_a}, [\tau], [\tau]\right\rangle\right\rangle$$

with task resource label M_{T_a} and a task T_b that deletes a term $reviewer\,(id_A, p)$ from the store S and writes a new term $submitted\,(id_A, p)$ to the store S is written as follows:

$$\begin{aligned} T_b \quad \triangleq \quad & \nu K_{T_b}.task\,\langle\langle K_{T_b}, M_{T_b}, \\ & [\mathcal{D}\,(S)\,(reviewer\,(id_A, p))]\,, [\mathcal{W}\,(S)\,(submitted\,(id_A, p))]\rangle\rangle \end{aligned}$$

with label M_{T_b} representing the resources assigned to task T_b.

A.5.2 Agent

We define $agent\,\langle\langle ID_A, K_A\rangle\rangle$ as syntactic sugar for the following code:

$$
\begin{aligned}
&\nu n_{nonce}.\,(t\,(owenc_{T,pub_T})\\
&.\textbf{let}\ wenc_{T,pub_T} = fst\,(dec\,(owenc_{T,pub_T}, sk\,(K_{Org})))\ \textbf{in}\\
&\textbf{let}\ id_T = snd\,(dec\,(owenc_{T,pub_T}, sk\,(K_{Org})))\ \textbf{in}\\
&\textbf{let}\ pub_T = thd\,(dec\,(owenc_{T,pub_T}, sk\,(K_{Org})))\ \textbf{in}\\
&\textbf{let}\ id2_T = fst\,(decs\,(wenc_{T,pub_T}, pub_W))\ \textbf{in}\\
&\textbf{let}\ pub2_T = snd\,(decs\,(wenc_{T,pub_T}, pub_W))\ \textbf{in}\\
&\textbf{if}\ id2_T = id_T\ \textbf{then}\\
&\textbf{if}\ pub2_T = pub_T\ \textbf{then}\\
&\textbf{let}\ aenc_{A,T} = encs\,((id_A, id_T), sk\,(K_A))\ \textbf{in}\\
&\textbf{let}\ waenc_{A,T,pub_A} = enc\,((aenc_{A,T}, pub_A), pub_W)\ \textbf{in}\\
&\overline{t}\,\langle waenc_{A,T,pub_A}\rangle\\
&.t\,(awenc_{T,n})\,.\textbf{if}\ fst\,(decs\,(dec\,(awenc_{T,n}, sk\,(K_A)), pub_W)) = id_T\ \textbf{then}\\
&\textbf{let}\ n = snd\,(decs\,(dec\,(awenc_{T,n}, sk\,(K_A)), pub_W))\ \textbf{in}\\
&\textbf{let}\ aenc_{T,A,n} = encs\,((id_T, id_A, n), sk\,(K_A))\ \textbf{in}\\
&\textbf{let}\ taenc_{T,A,n} = enc\,(aenc_{T,A,n}, pub_T)\ \textbf{in}\\
&\overline{t}\,\langle taenc_{T,A,n}\rangle\\
&.t\,(atenc_{M_T,T})\,.\textbf{let}\ M_A = fst\,(decs\,(dec\,(atenc_{M_T,T}, sk\,(K_A)), pub_T))\ \textbf{in}\\
&\textbf{let}\ aenc_{M_A,n_{nonce}} = enc\,((M_A, n_{nonce}), pub_A)\ \textbf{in}\\
&\overline{t}\,\langle aenc_{M_A,n_{nonce}}\rangle\\
&\textbf{let}\ aenc_{end,M_A,T} = encs\,(end\,(M_A, id_T), sk\,(K_A))\ \textbf{in}\\
&\textbf{let}\ taenc_{end,M_A,T} = enc\,(aenc_{end,M_A,T}, pub_T)\ \textbf{in}\\
&.\overline{t}\,\langle taenc_{end,M_A,T}\rangle\big)
\end{aligned}
$$

A.5.3 Workflow manager

We use $taskCall\,\langle\langle ID_T, K_T, ID_W, K_W, []\rangle\rangle$ to provide the building blocks for the definition of access control workflows within workflow manager W. Process context $taskCall\,\langle\langle ID_T, K_T, ID_W, K_W, []\rangle\rangle$ is defined as follows:

$$
\begin{aligned}
&(\textbf{let } wenc_T = encs\left((ID_T, ID_W), sk\left(K_W\right)\right) \textbf{ in}\\
&\textbf{let } twenc_T = enc\left(wenc_T, pub_T\right) \textbf{ in}\\
&\overline{t}\left\langle twenc_T\right\rangle.t\left(wtenc_T\right)\\
&.\textbf{if } decs\left(dec\left(wtenc_T, sk\left(K_W\right)\right), pub_T\right) = ID_T \textbf{ then}\\
&\textbf{let } wenc_{T,pub_T} = encs\left((ID_T, pub_T), sk\left(K_W\right)\right) \textbf{ in}\\
&\textbf{let } owenc_{T,pub_T} = enc\left((wenc_{T,pub_T}, ID_T, pub_T), pub_{Org}\right) \textbf{ in}\\
&\overline{t}\left\langle owenc_{T,pub_T}\right\rangle.t\left(waenc_{A,T,pub_A}\right)\\
&.\textbf{let } pub_A = snd\left(dec\left(waenc_{A,T,pub_A}, sk\left(K_W\right)\right)\right) \textbf{ in}\\
&\textbf{let } id_A = fst\left(decs\left(fst\left(dec\left(waenc_{A,T,pub_A}, sk\left(K_W\right)\right)\right), pub_A\right)\right) \textbf{ in}\\
&\textbf{if } snd\left(decs\left(fst\left(dec\left(waenc_{A,T,pub_A}, sk\left(K_W\right)\right)\right), pub_A\right)\right) = ID_T \textbf{ then}\\
&\mathcal{T}\left(Q\right)\left(\left(id_A, pub_A\right)\right):\\
&[]\\
&.\nu n_T.\left(\textbf{let } wenc_{T,n_T,A,pub_A} = encs\left((ID_T, n_T, id_A, pub_A), sk\left(K_W\right)\right) \textbf{ in}\right.\\
&\textbf{let } twenc_{T,n_T,A,pub_A} = enc\left(wenc_{T,n_T,A,pub_A}, pub_T\right) \textbf{ in}\\
&\overline{t}\left\langle twenc_{T,n_T,A,pub_A}\right\rangle\\
&.\textbf{let } wenc_{T,n_T} = encs\left((ID_T, n_T), sk\left(K_W\right)\right) \textbf{ in}\\
&\textbf{let } awenc_{T,n_T} = enc\left(wenc_{T,n_T}, pub_A\right) \textbf{ in}\\
&\left.\overline{t}\left\langle awenc_{T,n_T}\right\rangle\right).t\left(wtenc_{end,M_T,T}\right)\\
&.\textbf{if } decs\left(dec\left(wtenc_{end,M_T,T}, sk\left(K_W\right)\right), pub_T\right) = endtask\left(ID_T\right) \textbf{ then } \tau)
\end{aligned}
$$

An example of a task call using this syntax is:

$$
\begin{aligned}
TaskCall_2 \triangleq\ & \nu ID_{T_2}.taskCall\left\langle\left(ID_{T_2}, K_{T_2}, ID_W, K_W,\right.\right.\\
& [\left(\mathcal{T}\left(R\right)\left(role\left(id_A, clerk\right)\right):\right.\\
& \left.\left.\mathcal{T}\left(S\right)\left(hasAccessed\left(id_A, ID_{T_1}, ID_W\right)\right) : \mathbf{0} :\right)]\right\rangle
\end{aligned}
$$

wherein an instance of the task process identified by the public key pub_{T_2}, is assigned a fresh identity ID_{T_2}, and the task is advertised on the public channel. Once the task identity has been picked up

by an agent process then the task manager checks the credentials of the agent against the stores using the $\mathcal{T}(R)$ and $\mathcal{T}(S)$ test processes. On a satisfactory outcome the workflow manager then facilitates communication between the successful agent and the task.

A.5.4 The standard core protocol C

In section A.5 above we present the complete encoding with encryption for the three main processes that comprise our workflow model, namely agents A, tasks T and the workflow manager W. All three processes interact with each other in a predefined manner for the purpose of our modelling environment and we call this interaction the standard core protocol C.

Definition 21

Given a task process T as set out in section A.5.1, an agent process A as set out in section A.5.2 and a task-call sub-process taskCall $\langle\langle ID_T, K_T, ID_W, K_W, [] \rangle\rangle$ within a workflow process W as outlined in section A.5.3, all defined to follow the initialisation process $\mathcal{M}$ as per section 6.9, then the standard core protocol C is defined as the communication interaction between T, A and W as represented by the diagram 6.1.

A.6 Restrictions on model coding within the BPAC environment

A.6.1 Populating context holes

In formulating the detailed code for tasks (section A.5.1) and the workflow manager (section A.5.3) we have used the standard applied pi calculus concept of contexts and context holes. These indicate that the processes can be altered so that they incorporate the components that we use to manage access control, such as the interactions between processes and the stores. Although context holes can ordinarily be populated by any properly formulated applied pi calculus process, for the purpose of our BPAC environment we restrict the potential contents of these holes to the following: $\mathcal{W}(store)$, $\mathcal{T}(store)$, $\mathcal{D}(store)$ and τ. That is the context holes can only be populated by write, test and delete store functions and the internal process and no other applied pi calculus code. The reason for this restriction is that it ensures that the uniqueness property for the BPAC modelling environment, a key requirement for our security analysis methods, is not compromised.

A.6.2 Adding code to workflow models

Given that the BPAC modelling environment is based upon the applied pi calculus then it is to be expected that workflow models be extended with additional code to enable such features as, for example, simulation of inter-agent communication. Whilst any code that is properly formulated in accordance with the syntactic requirements of the applied pi calculus can, in theory, be added to a workflow model care must be taken to ensure that the uniqueness property is not violated. If it is, tests on the workflow model must be adapted accordingly to taken into account any repetition of transactions. In practice, with simple additions to workflow models, such as communications between agents, the uniqueness property can be maintained by the use of fresh nonces that are added to communicated messages.

A.7 Summary

In this appendix we have presented the detailed coding of the core access control protocol components. In section A.2 communication between core processes is discussed without encryption so as to highlight the salient features of the access control protocol. In section A.4 the fully encrypted protocol communications are discussed and this is followed in section A.5 with a presentation of the detailed encoding in the applied pi calculus of the core components of the access control modelling environment.

Appendix B

Detailed proofs

B.1 Detailed proof of the uniqueness property

B.1.1 The uniqueness property

Let $\mathbf{P}$ be any secure well-formed workflow model as per section 6.12 comprising a number of linked cases of standard core processes, $C_1, \ldots, C_m$ as defined in section A.5.4, initialised by process $\mathcal{M}$ as defined in section 6.9.4, such that $\mathbf{P}$ can undergo labelled transitions of the form $\mathbf{P} \xrightarrow{\alpha^1} \ldots \xrightarrow{\alpha^n} \mathbf{P'}$ then:

1. $\alpha^i \neq \alpha^j$ for $i, j = 1$ to n and $i \neq j$
2. For any two instances of $\mathbf{P}$, $\mathbf{P^1}$ and $\mathbf{P^2}$, say, initialised by process $\mathcal{M}$, such that $\mathbf{P^1} \xrightarrow{\alpha_1^1} \ldots \xrightarrow{\alpha_1^n} \mathbf{P'^1}$ and $\mathbf{P^2} \xrightarrow{\alpha_2^1} \ldots \xrightarrow{\alpha_2^n} \mathbf{P'^2}$ then $\alpha_1^k \neq \alpha_2^l$ for $k, l = 1$ to n.

B.1.2 Stage 1

Firstly, we consider a single run of the standard core protocol and we compare all transitions for that single run.

Let a single run of the standard core protocol within the well-formed workflow model $\mathbf{P}$ be represented by the following trace:

$\mathbf{P} \xrightarrow{\eta_1} \ldots \xrightarrow{\eta_n} \mathbf{P'}$

Where $\eta_1 \ldots \eta_n$ represent the n public transitions of the core protocol as previously defined and we ignore all private transitions.

Then $\eta_i \neq \eta_j$ where $i = 1$ to n, $j = 1$ to n and $i \neq j$.

B.1.3 Proof

We compare all public transitions, $\eta_1 \ldots \eta_n$, against each other within the standard core protocol of chapter 6 and appendix A. We prove that for each transition within the core protocol there is no equal transition within the rest of the core protocol as follows:

1. $\eta_1 = \nu ID_T, ID_W . (enc(encs((ID_T, ID_W), sk(K_W)), pub_T))$

   (a) Case $\eta_2 = enc(encs(ID_T, sk(K_T)), pub_W)$:

   Now $pub_T \triangleq pk(K_T)$ and $pub_W \triangleq pk(K_W)$ and

   $pub_T \neq pub_W$ as $pk(x)$ is collision-free

   $\therefore \nu ID_T, ID_W . (enc(encs((ID_T, ID_W), sk(K_W)), pub_T)) \neq$

   $enc(encs(ID_T, sk(K_T)), pub_W)$ as $enc(x, y)$ is collision-free

   (b) Case $\eta_3 = enc((encs((ID_T, pub_T), sk(K_W)), ID_T, pub_T), pub_{Org})$:

   Now $pub_T \triangleq pk(K_T)$ and $pub_{Org} \triangleq pk(K_{Org})$ and

   $pub_T \neq pub_{Org}$ as $pk(x)$ is collision-free

   $\therefore \nu ID_T, ID_W . (enc(encs((ID_T, ID_W), sk(K_W)), pub_T)) \neq$

   $enc((encs((ID_T, pub_T), sk(K_W)), ID_T, pub_T), pub_{Org})$

   as $enc(x, y)$ is collision-free

   (c) Case $\eta_4 = enc((encs((ID_A, ID_T), sk(K_A)), pub_A), pub_W)$:

   Now $pub_T \triangleq pk(K_T)$ and $pub_W \triangleq pk(K_W)$ and

   $pub_T \neq pub_W$ as $pk(x)$ is collision-free and

   $\therefore \nu ID_T, ID_W . (enc(encs((ID_T, ID_W), sk(K_W)), pub_T)) \neq$

   $enc((encs((ID_A, ID_T), sk(K_A)), pub_A), pub_W)$

   as $enc(x, y)$ is collision-free

   (d) Case $\eta_5 = \nu n_T . (enc(encs((ID_T, n_T, ID_A, pub_A), sk(K_W)), pub_T))$:

   now $(ID_T, ID_W) \neq (ID_T, n_T, ID_A, pub_A)$ by typing over tuples then

   $encs((ID_T, ID_W), sk(K_W)) \neq encs((ID_T, n_T, ID_A, pub_A), sk(K_W))$

   as $encs(x, y)$ is collision-free and

   $\therefore \nu ID_T, ID_W . (enc(encs((ID_T, ID_W), sk(K_W)), pub_T)) \neq$

   $\nu n_T . (enc(encs((ID_T, n_T, ID_A, pub_A), sk(K_W)), pub_T))$

   as $enc(x, y)$ is collision-free

(e) Case $\eta_6 = enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right)$:

Now $pub_T \triangleq pk\left(K_T\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_T \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore \nu ID_T, ID_W . \left(enc\left(encs\left(\left(ID_T, ID_W\right), sk\left(K_W\right)\right), pub_T\right)\right) \neq$

$enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(f) Case $\eta_7 = enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$:

$sk\left(K_W\right) \neq sk\left(K_A\right)$ as $sk\left(x\right)$ is collision-free and

$encs\left(\left(ID_T, ID_W\right), sk\left(K_W\right)\right) \neq$

$encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right)$ as

$encs\left(x, y\right)$ is collision-free

$\therefore \nu ID_T, ID_W . \left(enc\left(encs\left(\left(ID_T, ID_W\right), sk\left(K_W\right)\right), pub_T\right)\right) \neq$

$enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$ as

$enc\left(x, y\right)$ is collision-free

(g) Case $\eta_8 = enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$:

Now $pub_T \triangleq pk\left(K_T\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_T \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore \nu ID_T, ID_W . \left(enc\left(encs\left(\left(ID_T, ID_W\right), sk\left(K_W\right)\right), pub_T\right)\right) \neq$

$enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(h) Case $\eta_9 = enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$:

Now $pub_T \triangleq pk\left(K_T\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_T \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore \nu ID_T, ID_W . \left(enc\left(encs\left(\left(ID_T, ID_W\right), sk\left(K_W\right)\right), pub_T\right)\right) \neq$

$enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(i) Case $\eta_{10} = enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$:

$sk\left(K_W\right) \neq sk\left(K_A\right)$ as $sk\left(x\right)$ is collision-free and

$encs\left(\left(ID_T, ID_W\right), sk\left(K_W\right)\right) \neq$

$encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right)$ as

$encs\left(x, y\right)$ is collision-free

$\therefore \nu ID_T, ID_W.(enc(encs((ID_T, ID_W), sk(K_W)), pub_T)) \neq$

$enc(encs(end(M_T, ID_T), sk(K_A)), pub_T)$ as

$enc(x, y)$ is collision-free

(j) Case $\eta_{11} = enc(encs(endtask(ID_T), sk(K_T)), pub_W)$:

Now $pub_T \triangleq pk(K_T)$ and $pub_W \triangleq pk(K_W)$ and

$pub_T \neq pub_W$ as $pk(x)$ is collision-free and

$\therefore \nu ID_T, ID_W.(enc(encs((ID_T, ID_W), sk(K_W)), pub_T)) \neq$

$enc(encs(endtask(ID_T), sk(K_T)), pub_W)$ as

$enc(x, y)$ is collision-free

2. $\eta_2 = enc(encs(ID_T, sk(K_T)), pub_W)$

(a) Case $\eta_3 = enc((encs((ID_T, pub_T), sk(K_W)), ID_T, pub_T), pub_{Org})$:

Now $pub_W \triangleq pk(K_W)$ and $pub_{Org} \triangleq pk(K_{Org})$ and

$pub_W \neq pub_{Org}$ as $pk(x)$ is collision-free and

$\therefore enc(encs(ID_T, sk(K_T)), pub_W) \neq$

$enc((encs((ID_T, pub_T), sk(K_W)), ID_T, pub_T), pub_{Org})$

as $enc(x, y)$ is collision-free

(b) Case $\eta_4 = enc((encs((ID_A, ID_T), sk(K_A)), pub_A), pub_W)$:

Now $encs(ID_T, sk(K_T)) \neq encs((ID_A, ID_T), sk(K_A)), pub_A$

by typing over tuples and

$\therefore enc(encs(ID_T, sk(K_T)), pub_W) \neq$

$enc((encs((ID_A, ID_T), sk(K_A)), pub_A), pub_W)$ as

$enc(x, y)$ is collision-free

(c) Case $\eta_5 = \nu n_T.(enc(encs((ID_T, n_T, ID_A, pub_A), sk(K_W)), pub_T))$:

Now $pub_W \triangleq pk(K_W)$ and $pub_T \triangleq pk(K_T)$ and

$pub_W \neq pub_T$ as $pk(x)$ is collision-free and

$\therefore enc(encs(ID_T, sk(K_T)), pub_W) \neq$

$\nu n_T.(enc(encs((ID_T, n_T, ID_A, pub_A), sk(K_W)), pub_T))$

as $enc(x, y)$ is collision-free

(d) Case $\eta_6 = enc(encs((ID_T, n_T), sk(K_W)), pub_A)$:

Now $pub_W \triangleq pk(K_W)$ and $pub_A \triangleq pk(K_A)$ and

$pub_W \neq pub_A$ as $pk(x)$ is collision-free and

$\therefore enc(encs(ID_T, sk(K_T)), pub_W) \neq$

$enc(encs((ID_T, n_T), sk(K_W)), pub_A)$ as

$enc(x, y)$ is collision-free

(e) Case $\eta_7 = enc(encs((ID_T, ID_A, n_T), sk(K_A)), pub_T)$:

Now $pub_W \triangleq pk(K_W)$ and $pub_T \triangleq pk(K_T)$ and

$pub_W \neq pub_T$ as $pk(x)$ is collision-free and

$\therefore enc(encs(ID_T, sk(K_T)), pub_W) \neq$

$enc(encs((ID_T, ID_A, n_T), sk(K_A)), pub_T)$ as

$enc(x, y)$ is collision-free

(f) Case $\eta_8 = enc(encs((M_T, ID_T), sk(K_T)), pub_A)$:

Now $pub_W \triangleq pk(K_W)$ and $pub_A \triangleq pk(K_A)$ and

$pub_W \neq pub_A$ as $pk(x)$ is collision-free and

$\therefore enc(encs(ID_T, sk(K_T)), pub_W) \neq$

$enc(encs((M_T, ID_T), sk(K_T)), pub_A)$ as

$enc(x, y)$ is collision-free

(g) Case $\eta_9 = enc((M_A, n_{nonce}), pub_A)$:

Now $pub_W \triangleq pk(K_W)$ and $pub_A \triangleq pk(K_A)$ and

$pub_W \neq pub_A$ as $pk(x)$ is collision-free and

$\therefore enc(encs(ID_T, sk(K_T)), pub_W) \neq$

$enc((M_A, n_{nonce}), pub_A)$ as

$enc(x, y)$ is collision-free

(h) Case $\eta_{10} = enc(encs(end(M_T, ID_T), sk(K_A)), pub_T)$:

Now $pub_W \triangleq pk(K_W)$ and $pub_T \triangleq pk(K_T)$ and

$pub_W \neq pub_T$ as $pk(x)$ is collision-free and

$\therefore enc(encs(ID_T, sk(K_T)), pub_W) \neq$

$enc(encs(end(M_T, ID_T), sk(K_A)), pub_T)$ as

$enc(x, y)$ is collision-free

(i) Case $\eta_{11} = enc(encs(endtask(ID_T), sk(K_T)), pub_W)$:

Now $ID_T \neq endtask(ID_T)$ as $endtask(x)$

is a one-way function and

$encs\left(ID_T, sk\left(K_T\right)\right) \neq$

$encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right)$ as

$encs\left(x, y\right)$ is collision-free

$\therefore enc\left(encs\left(ID_T, sk\left(K_T\right)\right), pub_W\right) \neq$

$enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$ as

$enc\left(x, y\right)$ is collision-free

3. $\eta_3 = enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

   (a) Case $\eta_4 = enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right)$:

   Now $pub_{Org} \triangleq pk\left(K_{Org}\right)$ and $pub_W \triangleq pk\left(K_W\right)$ and

   $pub_{Org} \neq pub_W$ as $pk\left(x\right)$ is collision-free and

   $\therefore enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

   $\neq enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right)$

   as $enc\left(x, y\right)$ is collision-free

   (b) Case $\eta_5 = \nu n_T.\left(enc\left(encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right)$:

   Now $pub_{Org} \triangleq pk\left(K_{Org}\right)$ and $pub_T \triangleq pk\left(K_T\right)$ and

   $pub_{Org} \neq pub_T$ as $pk\left(x\right)$ is collision-free and

   $\therefore enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

   $\neq \nu n_T.\left(enc\left(encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right)$

   as $enc\left(x, y\right)$ is collision-free

   (c) Case $\eta_6 = enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right)$:

   Now $pub_{Org} \triangleq pk\left(K_{Org}\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

   $pub_{Org} \neq pub_A$ as $pk\left(x\right)$ is collision-free and

   $\therefore enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

   $\neq enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right)$

   as $enc\left(x, y\right)$ is collision-free

   (d) Case $\eta_7 = enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$:

   Now $pub_{Org} \triangleq pk\left(K_{Org}\right)$ and $pub_T \triangleq pk\left(K_T\right)$ and

   $pub_{Org} \neq pub_T$ as $pk\left(x\right)$ is collision-free and

   $\therefore enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

   $\neq enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$

   as $enc\left(x, y\right)$ is collision-free

(e) Case $\eta_8 = enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$:

Now $pub_{Org} \triangleq pk\left(K_{Org}\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_{Org} \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

$\neq enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$

as $enc\left(x, y\right)$ is collision-free

(f) Case $\eta_9 = enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$:

Now $pub_{Org} \triangleq pk\left(K_{Org}\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_{Org} \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

$enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(g) Case $\eta_{10} = enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$:

Now $pub_{Org} \triangleq pk\left(K_{Org}\right)$ and $pub_T \triangleq pk\left(K_T\right)$ and

$pub_{Org} \neq pub_T$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

$\neq enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$

as $enc\left(x, y\right)$ is collision-free

(h) Case $\eta_{11} = enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$:

Now $pub_{Org} \triangleq pk\left(K_{Org}\right)$ and $pub_W \triangleq pk\left(K_W\right)$ and

$pub_{Org} \neq pub_W$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

$\neq enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$

as $enc\left(x, y\right)$ is collision-free

4. $\eta_4 = enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right)$

(a) Case $\eta_5 = \nu n_T.\left(enc\left(encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right)$:

Now $pub_W \triangleq pk\left(K_W\right)$ and $pub_T \triangleq pk\left(K_T\right)$ and

$pub_W \neq pub_T$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right) \neq$

$\nu n_T.\left(enc\left(encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right)$ as

$enc\left(x, y\right)$ is collision-free

(b) Case $\eta_6 = enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right)$:

Now $pub_W \triangleq pk\left(K_W\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_W \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right) \neq$

$enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(c) Case $\eta_7 = enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$:

Now $pub_W \triangleq pk\left(K_W\right)$ and $pub_T \triangleq pk\left(K_T\right)$ and

$pub_W \neq pub_T$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right) \neq$

$enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$ as

$enc\left(x, y\right)$ is collision-free

(d) Case $\eta_8 = enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$:

Now $pub_W \triangleq pk\left(K_W\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_W \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right) \neq$

$enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(e) Case $\eta_9 = enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$:

Now $pub_W \triangleq pk\left(K_W\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_W \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right) \neq$

$enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(f) Case $\eta_{10} = enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$:

Now $pub_W \triangleq pk\left(K_W\right)$ and $pub_T \triangleq pk\left(K_T\right)$ and

$pub_W \neq pub_T$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right) \neq$

$enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$ as

$enc\left(x, y\right)$ is collision-free

(g) Case $\eta_{11} = enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$:

Now $encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A \neq$

$encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right)$

by typing over tuples

$\therefore enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right) \neq$

$enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$ as

$enc\left(x, y\right)$ is collision-free

5. $\eta_5 = \nu n_T.\left(enc\left(encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right)$

(a) Case $\eta_6 = enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right)$:

Now $pub_T \triangleq pk\left(K_T\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_T \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore \nu n_T.\left(enc\left(encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right) \neq$

$enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(b) Case $\eta_7 = enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$:

$sk\left(K_W\right) \neq sk\left(K_A\right)$ as $sk\left(x\right)$ is collision-free and

$encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right) \neq$

$encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right)$ as

$encs\left(x, y\right)$ is collision-free

$\therefore \nu n_T.\left(enc\left(encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right) \neq$

$enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$ as

$enc\left(x, y\right)$ is collision-free

(c) Case $\eta_8 = enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$:

Now $pub_T \triangleq pk\left(K_T\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_T \neq pub_A$ as $pk\left(x\right)$ is collision-free and

$\therefore \nu n_T.\left(enc\left(encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right) \neq$

$enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$ as

$enc\left(x, y\right)$ is collision-free

(d) Case $\eta_9 = enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$:

Now $pub_T \triangleq pk\left(K_T\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_T \neq pub_A$ as $pk(x)$ is collision-free and

$\therefore \nu n_T.(enc(encs((ID_T, n_T, ID_A, pub_A), sk(K_W)), pub_T)) \neq$

$enc((M_A, n_{nonce}), pub_A)$ as

$enc(x, y)$ is collision-free

(e) Case $\eta_{10} = enc(encs(end(M_T, ID_T), sk(K_A)), pub_T)$:

$sk(K_W) \neq sk(K_A)$ as $sk(x)$ is collision-free and

$\therefore enc((encs((ID_A, ID_T), sk(K_A)), pub_A), pub_W) \neq$

$encs(end(M_T, ID_T), sk(K_A))$ as

$encs(x, y)$ is collision-free

$\therefore \nu n_T.(enc(encs((ID_T, n_T, ID_A, pub_A), sk(K_W)), pub_T)) \neq$

$enc(encs(end(M_T, ID_T), sk(K_A)), pub_T)$ as

$enc(x, y)$ is collision-free

(f) Case $\eta_{11} = enc(encs(endtask(ID_T), sk(K_T)), pub_W)$:

Now $pub_T \triangleq pk(K_T)$ and $pub_W \triangleq pk(K_W)$ and

$pub_T \neq pub_A$ as $pk(x)$ is collision-free and

$\therefore \nu n_T.(enc(encs((ID_T, n_T, ID_A, pub_A), sk(K_W)), pub_T)) \neq$

$enc(encs(endtask(ID_T), sk(K_T)), pub_W)$ as

$enc(x, y)$ is collision-free

6. $\eta_6 = enc(encs((ID_T, n_T), sk(K_W)), pub_A)$

(a) Case $\eta_7 = enc(encs((ID_T, ID_A, n_T), sk(K_A)), pub_T)$:

Now $pub_A \triangleq pk(K_A)$ and $pub_T \triangleq pk(K_T)$ and

$pub_A \neq pub_T$ as $pk(x)$ is collision-free and

$\therefore enc(encs((ID_T, n_T), sk(K_W)), pub_A) \neq$

$enc(encs((ID_T, ID_A, n_T), sk(K_A)), pub_T)$ as

$enc(x, y)$ is collision-free

(b) Case $\eta_8 = enc(encs((M_T, ID_T), sk(K_T)), pub_A)$:

$sk(K_W) \neq sk(K_T)$ as $sk(x)$ is collision-free and

$encs((ID_T, n_T), sk(K_W)) \neq$

$encs((M_T, ID_T), sk(K_T))$ as

$encs(x, y)$ is collision-free

$\therefore enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right) \neq$

$enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(c) Case $\eta_9 = enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$:

Now $encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right) \neq \left(M_A, n_{nonce}\right)$

as collision-free and

$\therefore enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right) \neq$

$enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(d) Case $\eta_{10} = enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$:

Now $pub_A \triangleq pk\left(K_A\right)$ and $pub_T \triangleq pk\left(K_T\right)$ and

$pub_A \neq pub_T$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right) \neq$

$enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$ as

$enc\left(x, y\right)$ is collision-free

(e) Case $\eta_{11} = enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$:

Now $pub_A \triangleq pk\left(K_A\right)$ and $pub_W \triangleq pk\left(K_W\right)$ and

$pub_A \neq pub_W$ as $pk\left(x\right)$ is collision-free and

$\therefore enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right) \neq$

$enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$ as

$enc\left(x, y\right)$ is collision-free

7. $\eta_7 = enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$

(a) Case $\eta_8 = enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$:

Now $pub_T \triangleq pk\left(K_T\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_T \neq pub_A$ as $pk\left(x\right)$ is collision-free

$\therefore enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right) \neq$

$enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$ as

$enc\left(x, y\right)$ is collision-free

(b) Case $\eta_9 = enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$:

Now $pub_T \triangleq pk\left(K_T\right)$ and $pub_A \triangleq pk\left(K_A\right)$ and

$pub_T \neq pub_A$ as $pk(x)$ is collision-free and

$\therefore enc(encs((ID_T, ID_A, n_T), sk(K_A)), pub_T) \neq$

$enc((M_A, n_{nonce}), pub_A)$ as

$enc(x, y)$ is collision-free

(c) Case $\eta_{10} = enc(encs(end(M_T, ID_T), sk(K_A)), pub_T)$:

Now $(ID_T, ID_A, n_T) \neq end(M_T, ID_T)$ as

$end(x, y)$ is a one-way function and

$encs((ID_T, ID_A, n_T), sk(K_A)) \neq$

$encs(end(M_T, ID_T), sk(K_A))$ as

$encs(x, y)$ is collision-free

$\therefore enc(encs((ID_T, ID_A, n_T), sk(K_A)), pub_T) \neq$

$enc(encs(end(M_T, ID_T), sk(K_A)), pub_T)$ as

$enc(x, y)$ is collision-free

(d) Case $\eta_{11} = enc(encs(endtask(ID_T), sk(K_T)), pub_W)$:

Now $pub_T \triangleq pk(K_T)$ and $pub_W \triangleq pk(K_W)$ and

$pub_T \neq pub_W$ as $pk(x)$ is collision-free

$\therefore enc(encs((ID_T, ID_A, n_T), sk(K_A)), pub_T) \neq$

$enc(encs(endtask(ID_T), sk(K_T)), pub_W)$ as

$enc(x, y)$ is collision-free

8. $\eta_8 = enc(encs((M_T, ID_T), sk(K_T)), pub_A)$

(a) Case $\eta_9 = enc((M_A, n_{nonce}), pub_A)$:

Now $encs((ID_T, ID_A, n_T), sk(K_A)) \neq (M_A, n_{nonce})$

as collision-free and

$\therefore enc(encs((M_T, ID_T), sk(K_T)), pub_A) \neq$

$enc((M_A, n_{nonce}), pub_A)$ as

$enc(x, y)$ is collision-free

(b) Case $\eta_{10} = enc(encs(end(M_T, ID_T), sk(K_A)), pub_T)$:

Now $pub_A \triangleq pk(K_A)$ and $pub_T \triangleq pk(K_T)$ and

$pub_A \neq pub_T$ as $pk(x)$ is collision-free

$\therefore enc(encs((M_T, ID_T), sk(K_T)), pub_A) \neq$

$enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$ as

$enc\left(x, y\right)$ is collision-free

(c) Case $\eta_{11} = enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$:

Now $pub_A \triangleq pk\left(K_A\right)$ and $pub_W \triangleq pk\left(K_W\right)$ and

$pub_A \neq pub_W$ as $pk\left(x\right)$ is collision-free

$\therefore enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right) \neq$

$enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$ as

$enc\left(x, y\right)$ is collision-free

9. $\eta_9 = enc\left(\left(M_A, n_{nonce}\right), pub_A\right)$

(a) Case $\eta_{10} = enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$:

Now $pub_A \triangleq pk\left(K_A\right)$ and $pub_T \triangleq pk\left(K_T\right)$ and

$pub_A \neq pub_T$ as $pk\left(x\right)$ is collision-free

$\therefore enc\left(\left(M_A, n_{nonce}\right), pub_A\right) \neq$

$enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$ as

$enc\left(x, y\right)$ is collision-free

(b) Case $\eta_{11} = enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$:

Now $pub_A \triangleq pk\left(K_A\right)$ and $pub_W \triangleq pk\left(K_W\right)$ and

$pub_A \neq pub_W$ as $pk\left(x\right)$ is collision-free

$\therefore enc\left(\left(M_A, n_{nonce}\right), pub_A\right) \neq$

$enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$ as

$enc\left(x, y\right)$ is collision-free

10. $\eta_{10} = enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$

(a) Case $\eta_{11} = enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$:

Now $pub_T \triangleq pk\left(K_T\right)$ and $pub_W \triangleq pk\left(K_W\right)$ and

$pub_T \neq pub_W$ as $pk\left(x\right)$ is collision-free

$\therefore enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right) \neq$

$enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$ as

$enc\left(x, y\right)$ is collision-free

B.1.4 Conclusion

It can be concluded that there are no circumstances within the standard core protocol in which public transitions are equal therefore we conclude that all public transitions within the standard core protocol are unique.

B.1.5 Stage 2

Now we know from stage 1 of the proof that uniqueness is preserved for a single instance of the core protocol so that now we need to prove that uniqueness is preserved over multiple instances of the core protocol. For stage 2 of this proof we consider two generalised instances of the core protocol, C_1 and C_2 within the well-formed workflow model $\mathbf{P}$. We consider each protocol transition, η_i^1 and η_i^2 with $i = 1 \ldots n$ in turn and compare the transition for some task T_a represented by the task identity ID_{T_a} within C_1 against the equivalent transition for some task T_b represented by the task identity ID_{T_b} in C_2. In all of the following cases $pub_T \triangleq pk(K_T)$, $pub_W \triangleq pk(K_W)$ and $pub_A \triangleq pk(K_A)$ as usual.

B.1.6 Proof

We prove that the similar transitions within the core protocol for each of ID_{T_a} and ID_{T_b} are not equal as follows:

1. Case $\eta_1 = \nu ID_T, ID_W.(enc(encs((ID_T, ID_W), sk(K_W)), pub_T))$:

   Now $ID_{T_a} \neq ID_{T_b}$ for any a or b if ID_{T_a} and ID_{T_b} are freshly declared as is required for the workflow model.

   $\therefore \nu ID_{T_a}, ID_W.(enc(encs((ID_{T_a}, ID_W), sk(K_W)), pub_T)) \neq$

   $\nu ID_{T_b}, ID_W.(enc(encs((ID_{T_b}, ID_W), sk(K_W)), pub_T))$

   as $encs(x, y)$ and $enc(x, y)$ are collision-free for any ID_W, K_W and K_T.

2. Case $\eta_2 = enc(encs(ID_T, sk(K_T)), pub_W)$:

   Now $ID_{T_a} \neq ID_{T_b}$ for any a or b given ID_{T_a} and ID_{T_b} were freshly declared in 1. above.

   $\therefore enc(encs(ID_{T_a}, sk(K_T)), pub_W) \neq$

   $enc(encs(ID_{T_b}, sk(K_T)), pub_W)$

   as $encs(x, y)$ and $enc(x, y)$ are collision-free for any K_W and K_T.

3. Case $\eta_3 = enc\left(\left(encs\left(\left(ID_T, pub_T\right), sk\left(K_W\right)\right), ID_T, pub_T\right), pub_{Org}\right)$

   Now $ID_{T_a} \neq ID_{T_b}$ for any a or b given ID_{T_a} and ID_{T_b} were freshly declared in 1. above.

   $\therefore encs\left(\left(ID_{T_a}, pub_T\right), sk\left(K_W\right)\right), ID_{T_a}, pub_T \neq$

   $encs\left(\left(ID_{T_b}, pub_T\right), sk\left(K_W\right)\right), ID_{T_b}, pub_T$

   as $encs\left(x, y\right)$ is collision-free for any K_W and K_T.

   and $\therefore enc\left(\left(encs\left(\left(ID_{T_a}, pub_T\right), sk\left(K_W\right)\right), ID_{T_a}, pub_T\right), pub_{Org}\right)$

   $\neq enc\left(\left(encs\left(\left(ID_{T_b}, pub_T\right), sk\left(K_W\right)\right), ID_{T_b}, pub_T\right), pub_{Org}\right)$

   as $enc\left(x, y\right)$ is collision-free for any K_{Org} .

4. Case $\eta_4 = enc\left(\left(encs\left(\left(ID_A, ID_T\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right)$

   Now $ID_{T_a} \neq ID_{T_b}$ for any a or b given ID_{T_a} and ID_{T_b} were freshly declared in 1. above.

   $\therefore enc\left(\left(encs\left(\left(ID_A, ID_{T_a}\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right) \neq$

   $enc\left(\left(encs\left(\left(ID_A, ID_{T_b}\right), sk\left(K_A\right)\right), pub_A\right), pub_W\right)$

   as $encs\left(x, y\right)$ and $enc\left(x, y\right)$ are collision-free for any ID_A, K_A and K_W.

5. Case $\eta_5 = \nu n_T.\left(enc\left(encs\left(\left(ID_T, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right)$:

   Now $ID_{T_a} \neq ID_{T_b}$ for any a or b given ID_{T_a} and ID_{T_b} were freshly declared in 1. above.

   $\therefore \nu n_T.\left(enc\left(encs\left(\left(ID_{T_a}, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right) \neq$

   $\nu n_T.\left(enc\left(encs\left(\left(ID_{T_b}, n_T, ID_A, pub_A\right), sk\left(K_W\right)\right), pub_T\right)\right)$

   as $encs\left(x, y\right)$ and $enc\left(x, y\right)$ are collision-free for any ID_A, K_A, K_T and K_W.

6. Case $\eta_6 = enc\left(encs\left(\left(ID_T, n_T\right), sk\left(K_W\right)\right), pub_A\right)$:

   Now $ID_{T_a} \neq ID_{T_b}$ for any a or b given ID_{T_a} and ID_{T_b} were freshly declared in 1. above.

   $\therefore enc\left(encs\left(\left(ID_{T_a}, n_T\right), sk\left(K_W\right)\right), pub_A\right) \neq$

   $enc\left(encs\left(\left(ID_{T_b}, n_T\right), sk\left(K_W\right)\right), pub_A\right)$

   as $encs\left(x, y\right)$ and $enc\left(x, y\right)$ are collision-free for any K_A and K_W.

7. Case $\eta_7 = enc\left(encs\left(\left(ID_T, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$:

   Now $ID_{T_a} \neq ID_{T_b}$ for any a or b given ID_{T_a} and ID_{T_b} were freshly declared in 1. above.

   $\therefore enc\left(encs\left(\left(ID_{T_a}, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right) \neq$

   $enc\left(encs\left(\left(ID_{T_b}, ID_A, n_T\right), sk\left(K_A\right)\right), pub_T\right)$

   as $encs\left(x, y\right)$ and $enc\left(x, y\right)$ are collision-free for any ID_A, K_A and K_T.

8. Case $\eta_8 = enc\left(encs\left(\left(M_T, ID_T\right), sk\left(K_T\right)\right), pub_A\right)$:

   Now $ID_{T_a} \neq ID_{T_b}$ for any a or b given ID_{T_a} and ID_{T_b} were freshly declared in 1. above.

   $\therefore enc\left(encs\left(\left(M_T, ID_{T_a}\right), sk\left(K_T\right)\right), pub_A\right) \neq$

   $enc\left(encs\left(\left(M_T, ID_{T_b}\right), sk\left(K_T\right)\right), pub_A\right)$

   as $encs\left(x, y\right)$ and $enc\left(x, y\right)$ are collision-free for any M_T, K_A and K_T.

9. Case $\eta_9 = enc\left(encs\left(end\left(M_T, ID_T\right), sk\left(K_A\right)\right), pub_T\right)$:

   Now $ID_{T_a} \neq ID_{T_b}$ for any a or b given ID_{T_a} and ID_{T_b} were freshly declared in 1. above.

   $\therefore enc\left(encs\left(end\left(M_T, ID_{T_a}\right), sk\left(K_A\right)\right), pub_T\right) \neq$

   $enc\left(encs\left(end\left(M_T, ID_{T_b}\right), sk\left(K_A\right)\right), pub_T\right)$

   as $end\left(x, y\right)$, $encs\left(x, y\right)$ and $enc\left(x, y\right)$ are collision-free for any M_T, K_A and K_T.

10. Case $\eta_{10} = enc\left(encs\left(endtask\left(ID_T\right), sk\left(K_T\right)\right), pub_W\right)$:

    Now $ID_{T_a} \neq ID_{T_b}$ for any a or b given ID_{T_a} and ID_{T_b} were freshly declared in 1. above.

    $\therefore enc\left(encs\left(endtask\left(ID_{T_a}\right), sk\left(K_{T_a}\right)\right), pub_W\right) \neq$

    $enc\left(encs\left(endtask\left(ID_{T_b}\right), sk\left(K_{T_b}\right)\right), pub_W\right)$

    as $endtask\left(x, y\right)$, $encs\left(x, y\right)$ and $enc\left(x, y\right)$ are collision-free for any K_W and K_T.

11. We prove that $\eta_i^1 \neq \eta_j^2$ where $i \neq j$ in a similar manner to stage 1 of the proof, i.e. we consider each η_i^1 in turn from η_1^1 to η_{10}^1 and we compare against η_{i+1}^2 to η_{10}^2. All proof cases for stage 1 are independent of ID_T therefore all proof steps are the same for $\eta_i^1 \neq \eta_j^2$.

B.1.7 Conclusion

It can be concluded that no public transitions are equal for any two cases of the standard core protocol within well-formed workflow model $\mathbf{P}$ and all public transitions within the well-formed workflow model are unique.

B.1.8 Stage 3

Finally, we consider any two instances $\mathbf{P_1}$ and $\mathbf{P_2}$ of the well-formed workflow model $\mathbf{P}$ and we prove that public transitions are unique for these two instances $\mathbf{P_1}$ and $\mathbf{P_2}$.

B.1.9 Proof

1. By renaming we know that points 1 to 11 above apply for any ID_{T_a} and ID_{T_b} and we generalise to say that no two public transitions within multiple instances of the core protocol are equal.

B.1.10 Conclusion

It can be concluded that there are no circumstances within multiple instances of the core protocol for any task identities ID_{T_a} and ID_{T_b} in which public transitions are equal therefore we conclude that all public transitions within multiple instances of the core protocol are unique.

Appendix C

Labelled reductions for anonymity experiments

C.1 Experiment 8

The detailed labelled reduction steps for experiment 8 summarised by

No.	$\mathbf{P^{15}_{term}}$	$\mathbf{P^{16}_{term}}$
1	$W_{term} \overset{C_1}{\Leftrightarrow} T_1, A_1$	$W_{term} \overset{C_1}{\Leftrightarrow} T_1, A_3$ FAIL

are as displayed in figure C.1.

No.	$\mathbf{P^{15}_{term}}$		$\mathbf{P^{16}_{term}}$	
1		$\nu y_{41}.\overline{t}\langle y_{41}\rangle \xrightarrow{\quad}$		$\nu y_{42}.\overline{t}\langle y_{42}\rangle \xrightarrow{\quad}$
2	$\longrightarrow^*$	$\nu z_{11}.\overline{t}\langle z_{11}\rangle t(z_{11}) \xrightarrow{\quad}\xrightarrow{\quad}$	$\longrightarrow^*$	$\nu z_{12}.\overline{t}\langle z_{12}\rangle t(z_{12}) \xrightarrow{\quad}\xrightarrow{\quad}$
3	$\longrightarrow^*$	$\nu u_{11}.\overline{t}\langle u_{11}\rangle t(u_{11}) \xrightarrow{\quad}\xrightarrow{\quad}$	$\longrightarrow^*$	$\nu u_{12}.\overline{t}\langle u_{12}\rangle t(u_{12}) \xrightarrow{\quad}\xrightarrow{\quad}$
4	$\longrightarrow^*$	$\nu z_{21}.\overline{t}\langle z_{21}\rangle t(z_{21}) \xrightarrow{\quad}\xrightarrow{\quad}$	$\longrightarrow^*$	$\nu z_{22}.\overline{t}\langle z_{22}\rangle t(z_{22}) \xrightarrow{\quad}\xrightarrow{\quad}$
5	$\longrightarrow^*$	$\nu y_{11}.\overline{t}\langle y_{11}\rangle t(y_{11}) \xrightarrow{\quad}\xrightarrow{\quad}$	$\longrightarrow^*$	$\nu y_{12}.\overline{t}\langle y_{12}\rangle t(y_{12}) \xrightarrow{\quad}\xrightarrow{\quad}$
6	$\longrightarrow^*$	$\varnothing$	$\longrightarrow^*$	$\varnothing$
7	OR	$\nu z_{51}.\overline{t}\langle z_{51}\rangle t(z_{51}) \xrightarrow{\quad}\xrightarrow{\quad}$		FAIL
		$\ldots$		

Figure C.1: Reduction steps for processes $\mathbf{P^{15}_{term}}$and $\mathbf{P^{16}_{term}}$

C.2 Experiment 9

The detailed labelled reduction steps for experiment 9 summarised by

$\mathbf{P^{19}_{term}}$		$\mathbf{P^{20}_{term}}$
$\left\{W_1 \overset{C_{11}}{\Leftrightarrow} T_1, A_1\right\} \wr ID_{T_1}, pub_{T_1}$		$\left\{W_1 \overset{C_{12}}{\Leftrightarrow} T_1, A_4\right\} \wr ID_{T_1}, pub_{T_1}$
$\left\{W_2 \overset{C_{21}}{\Leftrightarrow} T_2, Att\,(A_2)\right\} \wr ID_{T_2}, pub_{T_2}, M_{T_2}$		$\left\{W_2 \overset{C_{22}}{\Leftrightarrow} T_2, Att\,(A_2)\right\} \wr ID_{T_2}, pub_{T_2}, M_{T_2}$

Represents the detailed test process based upon labelled bisimilarity as presented below.

We step through process $\mathbf{P^{19}_{term}}$ reductions, matching the reductions to $\mathbf{P^{20}_{term}}$ and we confirm static equivalence at each reduction step then we step through process $\mathbf{P^{20}_{term}}$, matching reductions to $\mathbf{P^{19}_{term}}$ and confirming static equivalence at each reduction step. We display the reductions for process $\mathbf{P^{19}_{term}}$ and tasks T_1 and T_2 in graphic form in figure C.2 and for process $\mathbf{P^{20}_{term}}$ in figure C.3. These graphics show successive states and labelled transitions that are numerically referenced to the tables of reductions for $\mathbf{P^{19}_{term}}$ and $\mathbf{P^{20}_{term}}$ and task T_1 in figure C.1 and $\mathbf{P^{19}_{term}}$ and $\mathbf{P^{20}_{term}}$ and task T_2 in figure C.2. Also, we display the static equivalences for each reduction stage of $\mathbf{P^{19}_{term}}$ and $\mathbf{P^{20}_{term}}$ in figures C.4 and C.5 and these are numerically cross-referenced to the representations of transitions and reduction steps.

Static equivalence at each stage of the reduction is as displayed for task T_1 in figure C.4 and for task T_2 in figures C.5 and C.6.

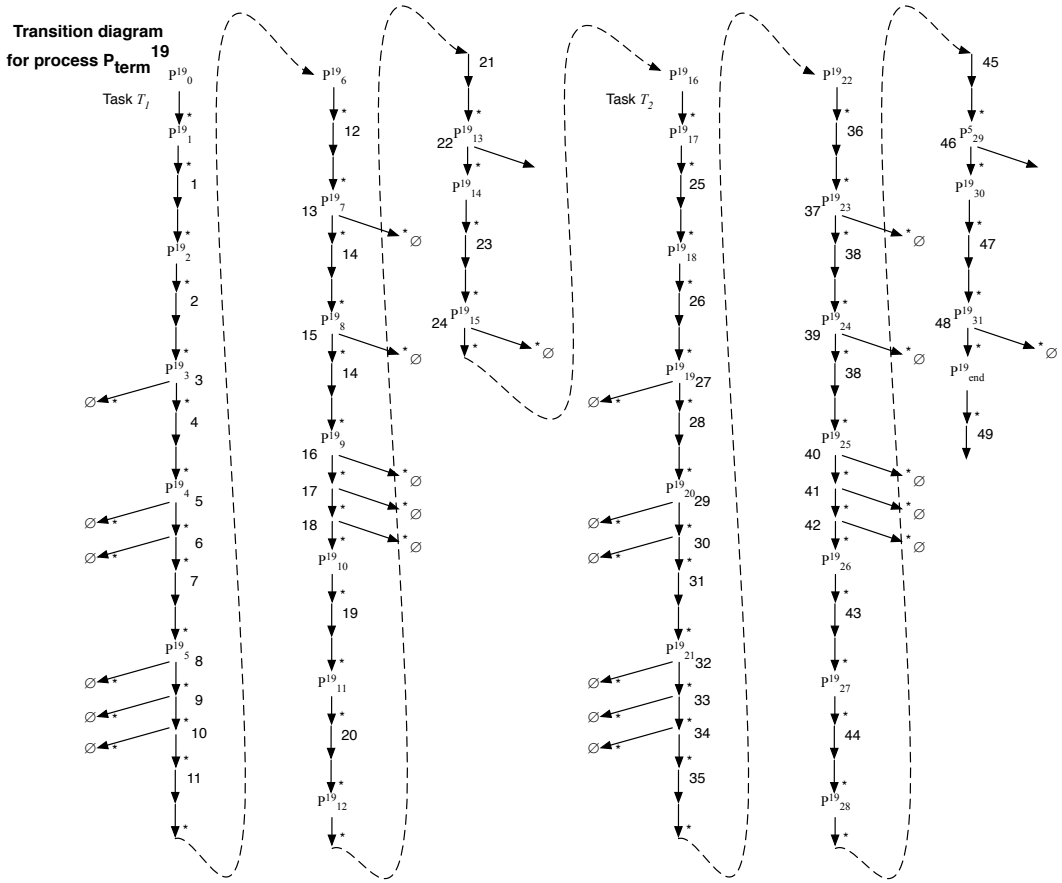


Figure C.2: Transition diagram for process $\mathbf{P}^{\mathbf{19}}_{\mathbf{term}}$

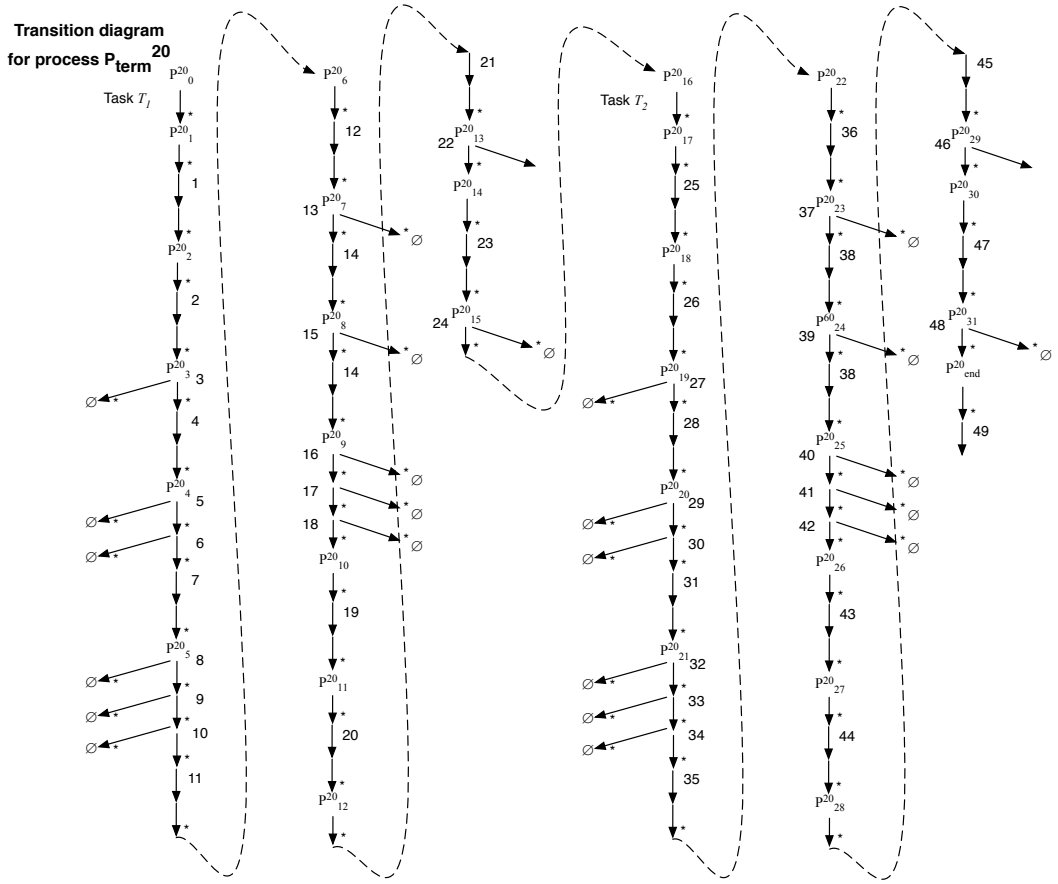


Figure C.3: Transition diagram for process $\mathbf{P}^{\mathbf{20}}_{\mathbf{term}}$

$\mathbf{P^{19}_{term}}$				$\mathbf{P^{20}_{term}}$	
Task T_1					
No.	State	Transition		State	Transition
0	P_0^{19}	$\longrightarrow^*$		P_0^{20}	$\longrightarrow^*$
1	P_1^{19}	$\longrightarrow^*$ $\xrightarrow{\nu z_{15}.\bar{t}\langle z_{15}\rangle}\xrightarrow{t(z_{15})}\longrightarrow^*$		P_1^{20}	$\longrightarrow^*$ $\xrightarrow{\nu z_{16}.\bar{t}\langle z_{16}\rangle}\xrightarrow{t(z_{16})}\longrightarrow^*$
2	P_2^{19}	$\longrightarrow^*$ $\xrightarrow{\nu u_{15}.\bar{t}\langle u_{15}\rangle}\xrightarrow{t(u_{15})}\longrightarrow^*$		P_2^{20}	$\longrightarrow^*$ $\xrightarrow{\nu u_{16}.\bar{t}\langle u_{16}\rangle}\xrightarrow{t(u_{16})}\longrightarrow^*$
3	P_3^{19}	$\longrightarrow^*$ $\varnothing$		P_3^{20}	$\longrightarrow^*$ $\varnothing$
4	OR	$\longrightarrow^*$ $\xrightarrow{\nu z_{25}.\bar{t}\langle z_{25}\rangle}\xrightarrow{t(z_{25})}\longrightarrow^*$		OR	$\longrightarrow^*$ $\xrightarrow{\nu z_{26}.\bar{t}\langle z_{26}\rangle}\xrightarrow{t(z_{26})}\longrightarrow^*$
5	P_4^{19}	$\longrightarrow^*$ $\varnothing$		P_4^{20}	$\longrightarrow^*$ $\varnothing$
6	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
7	OR	$\longrightarrow^*$ $\xrightarrow{\nu y_{15}.\bar{t}\langle y_{15}\rangle}\xrightarrow{t(y_{15})}\longrightarrow^*$		OR	$\longrightarrow^*$ $\xrightarrow{\nu y_{16}.\bar{t}\langle y_{16}\rangle}\xrightarrow{t(y_{16})}\longrightarrow^*$
8	P_5^{19}	$\longrightarrow^*$ $\varnothing$		P_5^{20}	$\longrightarrow^*$ $\varnothing$
9	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
10	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
11	OR	$\longrightarrow^*$ $\xrightarrow{\nu z_{55}.\bar{t}\langle z_{55}\rangle}\xrightarrow{t(z_{55})}\longrightarrow^*$		OR	$\longrightarrow^*$ $\xrightarrow{\nu z_{56}.\bar{t}\langle z_{56}\rangle}\xrightarrow{t(z_{56})}\longrightarrow^*$
12	P_6^{19}	$\longrightarrow^*$ $\xrightarrow{\nu z_{65}.\bar{t}\langle z_{65}\rangle}\xrightarrow{t(z_{65})}\longrightarrow^*$		P_6^{20}	$\longrightarrow^*$ $\xrightarrow{\nu z_{66}.\bar{t}\langle z_{66}\rangle}\xrightarrow{t(z_{66})}\longrightarrow^*$
13	P_7^{19}	$\longrightarrow^*$ $\varnothing$		P_7^{20}	$\longrightarrow^*$ $\varnothing$
14	OR	$\longrightarrow^*$ $\xrightarrow{\nu y_{25}.\bar{t}\langle y_{25}\rangle}\longrightarrow^*$		OR	$\longrightarrow^*$ $\xrightarrow{\nu y_{26}.\bar{t}\langle y_{26}\rangle}\longrightarrow^*$
15	P_8^{19}	$\longrightarrow^*$ $\varnothing$		P_8^{20}	$\longrightarrow^*$ $\varnothing$
14	OR	$\xrightarrow{t(y_{25})}\longrightarrow^*$		OR	$\xrightarrow{t(y_{26})}\longrightarrow^*$
16	P_9^{19}	$\longrightarrow^*$ $\varnothing$		P_9^{20}	$\longrightarrow^*$ $\varnothing$
17	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
18	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
	OR	$\longrightarrow^*$		OR	$\longrightarrow^*$
19	P_{10}^{19}	$\longrightarrow^*$ $\xrightarrow{\nu u_{25}.\bar{t}\langle u_{25}\rangle}\xrightarrow{t(u_{25})}\longrightarrow^*$		P_{10}^{20}	$\longrightarrow^*$ $\xrightarrow{\nu u_{26}.\bar{t}\langle u_{26}\rangle}\xrightarrow{t(u_{26})}\longrightarrow^*$
20	P_{11}^{19}	$\longrightarrow^*$ $\xrightarrow{\nu y_{35}.\bar{t}\langle y_{35}\rangle}\longrightarrow^*$		P_{11}^{20}	$\longrightarrow^*$ $\xrightarrow{\nu y_{36}.\bar{t}\langle y_{36}\rangle}\longrightarrow^*$
21	P_{12}^{19}	$\longrightarrow^*$ $\xrightarrow{\nu y_{45}.\bar{t}\langle y_{45}\rangle}\xrightarrow{t(y_{45})}\longrightarrow^*$		P_{12}^{20}	$\longrightarrow^*$ $\xrightarrow{\nu y_{46}.\bar{t}\langle y_{46}\rangle}\xrightarrow{t(y_{46})}\longrightarrow^*$
22	P_{13}^{19}	$\longrightarrow^*$ $\varnothing$		P_{13}^{20}	$\longrightarrow^*$ $\varnothing$
	OR	$\longrightarrow^*$		OR	$\longrightarrow^*$
23	P_{14}^{19}	$\longrightarrow^*$ $\xrightarrow{\nu u_{35}.\bar{t}\langle u_{35}\rangle}\xrightarrow{t(u_{35})}\longrightarrow^*$		P_{14}^{20}	$\longrightarrow^*$ $\xrightarrow{\nu u_{36}.\bar{t}\langle u_{36}\rangle}\xrightarrow{t(u_{36})}\longrightarrow^*$
24	P_{15}^{19}	$\longrightarrow^*$ $\varnothing$		P_{15}^{20}	$\longrightarrow^*$ $\varnothing$
	OR	$\longrightarrow^*$		OR	$\longrightarrow^*$

Table C.1: Reduction steps for processes $\mathbf{P^{19}_{term}}$ and $\mathbf{P^{20}_{term}}$ — task T_1

Task T_2					
No.	State	Transition		State	Transition
	P_{16}^{19}	$\longrightarrow^*$		P_{16}^{20}	$\longrightarrow^*$
25	P_{17}^{19}	$\longrightarrow^* \xrightarrow{\nu z_{75}.\bar{t}\langle z_{75}\rangle} \xrightarrow{t(z_{75})} \longrightarrow^*$		P_{17}^{20}	$\longrightarrow^* \xrightarrow{\nu z_{76}.\bar{t}\langle z_{76}\rangle} \xrightarrow{t(z_{76})} \longrightarrow^*$
26	P_{18}^{19}	$\longrightarrow^* \xrightarrow{\nu u_{45}.\bar{t}\langle u_{45}\rangle} \xrightarrow{t(u_{45})} \longrightarrow^*$		P_{18}^{20}	$\longrightarrow^* \xrightarrow{\nu u_{46}.\bar{t}\langle u_{46}\rangle} \xrightarrow{t(u_{46})} \longrightarrow^*$
27	P_{19}^{19}	$\longrightarrow^*$ $\varnothing$		P_{19}^{20}	$\longrightarrow^*$ $\varnothing$
28	OR	$\longrightarrow^* \xrightarrow{\nu z_{85}.\bar{t}\langle z_{85}\rangle} \xrightarrow{t(z_{85})}$		OR	$\longrightarrow^* \xrightarrow{\nu z_{86}.\bar{t}\langle z_{86}\rangle} \xrightarrow{t(z_{86})}$
29	P_{20}^{19}	$\longrightarrow^*$ $\varnothing$		P_{20}^{20}	$\longrightarrow^*$ $\varnothing$
30	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
31	OR	$\longrightarrow^* \xrightarrow{\nu y_{55}.\bar{t}\langle y_{55}\rangle} \xrightarrow{t(y_{55})} \longrightarrow^*$		OR	$\longrightarrow^* \xrightarrow{\nu y_{56}.\bar{t}\langle y_{56}\rangle} \xrightarrow{t(y_{56})} \longrightarrow^*$
32	P_{21}^{19}	$\longrightarrow^*$ $\varnothing$		P_{21}^{20}	$\longrightarrow^*$ $\varnothing$
33	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
34	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
35	OR	$\longrightarrow^* \xrightarrow{\nu z_{115}.\bar{t}\langle z_{115}\rangle} \xrightarrow{t(z_{115})} \longrightarrow^*$		OR	$\longrightarrow^* \xrightarrow{\nu z_{116}.\bar{t}\langle z_{116}\rangle} \xrightarrow{t(z_{116})} \longrightarrow^*$
36	P_{22}^{19}	$\longrightarrow^* \xrightarrow{\nu z_{125}.\bar{t}\langle z_{125}\rangle} \xrightarrow{t(z_{125})} \longrightarrow^*$		P_{22}^{20}	$\longrightarrow^* \xrightarrow{\nu z_{126}.\bar{t}\langle z_{126}\rangle} \xrightarrow{t(z_{126})} \longrightarrow^*$
37	P_{23}^{19}	$\longrightarrow^*$ $\varnothing$		P_{23}^{20}	$\longrightarrow^*$ $\varnothing$
38	OR	$\xrightarrow{\nu y_{65}.\bar{t}\langle y_{65}\rangle} \longrightarrow^*$		OR	$\xrightarrow{\nu y_{66}.\bar{t}\langle y_{66}\rangle} \longrightarrow^*$
39	P_{24}^{19}	$\longrightarrow^*$ $\varnothing$		P_{24}^{20}	$\longrightarrow^*$ $\varnothing$
38	OR	$\xrightarrow{t(y_{65})} \longrightarrow^*$		OR	$\xrightarrow{t(y_{66})} \longrightarrow^*$
40	P_{25}^{19}	$\longrightarrow^*$ $\varnothing$		P_{25}^{20}	$\longrightarrow^*$ $\varnothing$
41	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
42	OR	$\longrightarrow^*$ $\varnothing$		OR	$\longrightarrow^*$ $\varnothing$
	OR	$\longrightarrow^*$		OR	$\longrightarrow^*$
43	P_{26}^{19}	$\longrightarrow^* \xrightarrow{\nu u_{55}.\bar{t}\langle u_{55}\rangle} \xrightarrow{t(u_{55})} \longrightarrow^*$		P_{26}^{20}	$\longrightarrow^* \xrightarrow{\nu u_{56}.\bar{t}\langle u_{56}\rangle} \xrightarrow{t(u_{56})} \longrightarrow^*$
44	P_{27}^{19}	$\longrightarrow^* \xrightarrow{\nu y_{75}.\bar{t}\langle y_{75}\rangle} \longrightarrow^*$		P_{27}^{20}	$\longrightarrow^* \xrightarrow{\nu y_{76}.\bar{t}\langle y_{76}\rangle} \longrightarrow^*$
45	P_{28}^{19}	$\longrightarrow^* \xrightarrow{\nu y_{85}.\bar{t}\langle y_{85}\rangle} \xrightarrow{t(y_{85})} \longrightarrow^*$		P_{28}^{20}	$\longrightarrow^* \xrightarrow{\nu y_{86}.\bar{t}\langle y_{86}\rangle} \xrightarrow{t(y_{86})} \longrightarrow^*$
46	P_{29}^{19}	$\longrightarrow^*$ $\varnothing$		P_{29}^{20}	$\longrightarrow^*$ $\varnothing$
	OR	$\longrightarrow^*$		OR	$\longrightarrow^*$
47	P_{30}^{19}	$\longrightarrow^* \xrightarrow{\nu u_{65}.\bar{t}\langle u_{65}\rangle} \xrightarrow{t(u_{65})} \longrightarrow^*$		P_{30}^{20}	$\longrightarrow^* \xrightarrow{\nu u_{66}.\bar{t}\langle u_{66}\rangle} \xrightarrow{t(u_{66})} \longrightarrow^*$
48	P_{31}^{19}	$\longrightarrow^*$ $\varnothing$		P_{31}^{20}	$\longrightarrow^*$ $\varnothing$
	OR	$\longrightarrow^*$		OR	$\longrightarrow^*$
49	P_{end}^{19}	$\longrightarrow^* \xrightarrow{\bar{t}\langle M_{end}\rangle}$		P_{end}^{20}	$\longrightarrow^* \xrightarrow{\bar{t}\langle M_{end}\rangle}$

Table C.2: Reduction steps for processes $\mathbf{P_{term}^{19}}$ and $\mathbf{P_{term}^{20}}$ — task T_2

	$\mathbf{P^{19}_{term}}$		$\mathbf{P^{20}_{term}}$
Task T_1			
No.			
1	$\nu twenc_{T_1}.\{^{twenc_{T_1}}/_{z_{15}}\}$	$\approx_S$	$\nu twenc_{T_1}.\{^{twenc_{T_1}}/_{z_{16}}\}$
2	$\nu wtenc_{T_1}.\{^{wtenc_{T_1}}/_{u_{15}}\}$	$\approx_S$	$\nu wtenc_{T_1}.\{^{wtenc_{T_1}}/_{u_{16}}\}$
4	$\{^{fst(decs(wenc_{T_1,pub_{T_1}},pub_W))}/_{id_T}\} \mid$ $\{^{snd(decs(wenc_{T_1,pub_{T_1}},pub_W))}/_{pub_T}\} \mid$ $\{^{fst(dec(owenc_{T_1,pub_{T_1}},sk(K_{Org})))}/_{wenc_{T_1,pub_{T_1}}}\} \mid$ $\{^{snd(dec(owenc_{T_1,pub_{T_1}},sk(K_{Org})))}/_{id2_T}\} \mid$ $\{^{thd(dec(owenc_{T_1,pub_{T_1}},sk(K_{Org})))}/_{pub2_T}\} \mid$ $\{^{owenc_{T_1,pub_{T_1}}}/_{z_{25}}\}$	$\approx_S$	$\{^{fst(decs(wenc_{T_1,pub_{T_1}},pub_W))}/_{id_{T_1}}\} \mid$ $\{^{snd(decs(wenc_{T_1,pub_{T_1}},pub_W))}/_{pub_{T_1}}\} \mid$ $\{^{fst(dec(owenc_{T_1,pub_{T_1}},sk(K_{Org})))}/_{wenc_{T_1,pub_{T_1}}}\} \mid$ $\{^{snd(dec(owenc_{T_1,pub_{T_1}},sk(K_{Org})))}/_{id2_{T_1}}\} \mid$ $\{^{thd(dec(owenc_{T_1,pub_{T_1}},sk(K_{Org})))}/_{pub2_{T_1}}\} \mid$ $\{^{owenc_{T_1,pub_{T_1}}}/_{z_{26}}\}$
7	$\nu waenc_{A_1,T_1,pub_{A_1}}.$ $\{^{waenc_{A_1,T_1,pub_{A_1}}}/_{y_{15}}\}$	$\approx_S$	$\nu waenc_{A_4,T_1,pub_{A_4}}.$ $\{^{waenc_{A_4,T_1,pub_{A_4}}}/_{y_{16}}\}$
11	$\nu twenc_{T_1,n_{T_1},A_1,pub_{A_1}}.$ $\{^{twenc_{T_1,n_{T_1},A_1,pub_{A_1}}}/_{z_{55}}\}$	$\approx_S$	$\nu twenc_{T_1,n_{T_1},A_4,pub_{A_4}}.$ $\{^{twenc_{T_1,n_{T_1},A_4,pub_{A_4}}}/_{z_{56}}\}$
12	$\nu awenc_{T_1,n_{T_1}}.$ $\{^{awenc_{T_1,n_{T_1}}}/_{z_{65}}\}$	$\approx_S$	$\nu awenc_{T_1,n_{T_1}}.$ $\{^{awenc_{T_1,n_{T_1}}}/_{z_{66}}\}$
14	$\nu taenc_{T_1,A_1,n_{T_1}}.$ $\{^{taenc_{T_1,A_1,n_{T_1}}}/_{y_{25}}\}$	$\approx_S$	$\nu taenc_{T_1,A_4,n_{T_1}}.$ $\{^{taenc_{T_1,A_4,n_{T_1}}}/_{y_{26}}\}$
14	$\nu taenc_{T_1,A_1,n_{T_1}}.$ $\{^{taenc_{T_1,A_1,n_{T_1}}}/_{y_{25}}\}$	$\approx_S$	$\nu taenc_{T_1,A_4,n_{T_1}}.$ $\{^{taenc_{T_1,A_4,n_{T_1}}}/_{y_{26}}\}$
19	$\nu atenc_{M_{T_1},T_1}.$ $\{^{atenc_{M_{T_1},T_1}}/_{u_{25}}\}$	$\approx_S$	$\nu atenc_{M_{T_1},T_1}.$ $\{^{atenc_{M_{T_1},T_1}}/_{u_{26}}\}$
20	$\nu aenc_{M_{T_1},n_{nonce}}.$ $\{^{aenc_{M_{T_1},n_{nonce}}}/_{y_{35}}\}$	$\approx_S$	$\nu aenc_{M_{T_1},n_{nonce}}.$ $\{^{aenc_{M_{T_1},n_{nonce}}}/_{y_{36}}\}$
21	$\nu taenc_{end,M_{T_1},T_1}.$ $\{^{taenc_{end,M_{T_1},T_1}}/_{y_{45}}\}$	$\approx_S$	$\nu taenc_{end,M_{T_1},T_1}.$ $\{^{taenc_{end,M_{T_1},T_1}}/_{y_{46}}\}$
23	$\nu wtenc_{end,M_{T_1},T_1}.$ $\{^{wtenc_{end,M_{T_1},T_1}}/_{u_{35}}\}$	$\approx_S$	$\nu wtenc_{end,M_{T_1},T_1}.$ $\{^{wtenc_{end,M_{T_1},T_1}}/_{u_{36}}\}$

Figure C.4: Static equivalence for processes $\mathbf{P^{19}_{term}}$ and $\mathbf{P^{20}_{term}}$ — task T_1

	$\mathbf{P}^{\mathbf{19}}_{\mathbf{term}}$		$\mathbf{P}^{\mathbf{20}}_{\mathbf{term}}$
Task T_2			
No.			
24	$\nu twenc_{T_2}.\{^{twenc_{T_2}}/_{z_{75}}\}$	$\approx_S$	$\nu twenc_{T_2}.\{^{twenc_{T_2}}/_{z_{76}}\}$
25	$\nu wtenc_{T_2}.\{^{wtenc_{T_2}}/_{u_{45}}\}$	$\approx_S$	$\nu wtenc_{T_2}.\{^{wtenc_{T_2}}/_{u_{46}}\}$
27	$\{^{fst(decs(wenc_{T_2,pub_{T_2}},pub_W))}/_{id_{T_2}}\} \mid$ $\{^{snd(decs(wenc_{T_2,pub_{T_2}},pub_W))}/_{pub_{T_2}}\} \mid$ $\{^{fst(dec(owenc_{T_2,pub_{T_2}},sk(K_{Org})))}/_{wenc_{T_2,pub_{T_2}}}\} \mid$ $\{^{snd(dec(owenc_{T_2,pub_{T_2}},sk(K_{Org})))}/_{id2_{T_2}}\} \mid$ $\{^{thd(dec(owenc_{T_2,pub_{T_2}},sk(K_{Org})))}/_{pub2_{T_2}}\} \mid$ $\{^{owenc_{T_2,pub_{T_2}}}/_{z_{85}}\}$	$\approx_S$	$\{^{fst(decs(wenc_{T_2,pub_{T_2}},pub_W))}/_{id_{T_2}}\} \mid$ $\{^{snd(decs(wenc_{T_2,pub_{T_2}},pub_W))}/_{pub_{T_2}}\} \mid$ $\{^{fst(dec(owenc_{T_2,pub_{T_2}},sk(K_{Org})))}/_{wenc_{T_2,pub_{T_2}}}\} \mid$ $\{^{snd(dec(owenc_{T_2,pub_{T_2}},sk(K_{Org})))}/_{id2_{T_2}}\} \mid$ $\{^{thd(dec(owenc_{T_2,pub_{T_2}},sk(K_{Org})))}/_{pub2_{T_2}}\} \mid$ $\{^{owenc_{T_2,pub_{T_2}}}/_{z_{86}}\}$
31	$\{^{encs((ID_{A_2},id_{T_2}),sk(K_{T_2}))}/_{aenc_{A_2,T_2}}\} \mid$ $\{^{enc((aenc_{A_2,T_2},pub_{A_2}),pub_W)}/_{waenc_{A_2,T_2,pub_{A_2}}}\} \mid$ $\{^{waenc_{A_2,T_2,pub_{A_2}}}/_{y_{55}}\}$	$\approx_S$	$\{^{encs((ID_{A_2},id_{T_2}),sk(K_{T_2}))}/_{aenc_{A_2,T_2}}\} \mid$ $\{^{enc((aenc_{A_2,T_2},pub_{A_2}),pub_W)}/_{waenc_{A_2,T_2,pub_{A_2}}}\} \mid$ $\{^{waenc_{A_2,T_2,pub_{A_2}}}/_{y_{56}}\}$
35	$\nu twenc_{T_2,n_{T_2},A_2,pub_{A_2}}.$ $\{^{twenc_{T_2,n_{T_2},A_2,pub_{A_2}}}/_{z_{115}}\}$	$\approx_S$	$\nu twenc_{T_2,n_{T_2},A_2,pub_{A_2}}.$ $\{^{twenc_{T_2,n_{T_2},A_2,pub_{A_2}}}/_{z_{116}}\}$
36	$\{^{fst(decs(wenc_{T_2,n_{T_2}},pub_W))}/_{id3_{T_2}}\} \mid$ $\{^{snd(decs(wenc_{T_2,n_{T_2}},pub_W))}/_{n_{T_2}}\} \mid$ $\{^{dec(awenc_{T_2,n_{T_2}},sk(K_{A_2}))}/_{wenc_{T_2,n_{T_2}}}\} \mid$ $\{^{awenc_{T_2,n_{T_2}}}/_{z_{125}}\}$	$\approx_S$	$\{^{fst(decs(wenc_{T_2,n_{T_2}},pub_W))}/_{id3_{T_2}}\} \mid$ $\{^{snd(decs(wenc_{T_2,n_{T_2}},pub_W))}/_{n_{T_2}}\} \mid$ $\{^{dec(awenc_{T_2,n_{T_2}},sk(K_{A_2}))}/_{wenc_{T_2,n_{T_2}}}\} \mid$ $\{^{awenc_{T_2,n_{T_2}}}/_{z_{126}}\}$
38	$\nu n.$ $\{^{encs((id_{T_2},ID_{A_2},n),sk(K_{A_2}))}/_{aenc_{T_2,A_2,n}}\} \mid$ $\{^{enc(aenc_{T_2,A_2,n},pub_{T_2})}/_{taenc_{T_2,A_2,n_{T_2}}}\} \mid$ $\{^{taenc_{T_2,A_2,n_{T_2}}}/_{y_{65}}\}$	$\approx_S$	$\nu n.$ $\{^{encs((id_{T_2},ID_{A_2},n),sk(K_{A_2}))}/_{aenc_{T_2,A_2,n}}\} \mid$ $\{^{enc(aenc_{T_2,A_2,n},pub_{T_2})}/_{taenc_{T_2,A_2,n_{T_2}}}\} \mid$ $\{^{taenc_{T_2,A_2,n_{T_2}}}/_{y_{66}}\}$
38	$\{^{encs((id_{T_2},ID_{A_2},n),sk(K_{A_2}))}/_{aenc_{T_2,A_2,n}}\} \mid$ $\{^{enc(aenc_{T_2,A_2,n},pub_{T_2})}/_{taenc_{T_2,A_2,n_{T_2}}}\} \mid$ $\{^{taenc_{T_2,A_2,n_{T_2}}}/_{y_{65}}\}$	$\approx_S$	$\{^{encs((id_{T_2},ID_{A_2},n),sk(K_{A_2}))}/_{aenc_{T_2,A_2,n}}\} \mid$ $\{^{enc(aenc_{T_2,A_2,n},pub_{T_2})}/_{taenc_{T_2,A_2,n_{T_2}}}\} \mid$ $\{^{taenc_{T_2,A_2,n_{T_2}}}/_{y_{66}}\}$
43	$\{^{fst(decs(tenc_{M_{T_2},T_2},pub_{T_2}))}/_{M_{T_2}}\} \mid$ $\{^{snd(decs(tenc_{M_{T_2},T_2},pub_{T_2}))}/_{id4_{T_2}}\} \mid$ $\{^{dec(atenc_{M_{T_2},T_2},sk(K_{A_2}))}/_{tenc_{M_{T_2},T_2}}\} \mid$ $\{^{atenc_{M_{T_2},T_2}}/_{u_{55}}\}$	$\approx_S$	$\{^{fst(decs(tenc_{M_{T_2},T_2},pub_{T_2}))}/_{M_{T_2}}\} \mid$ $\{^{snd(decs(tenc_{M_{T_2},T_2},pub_{T_2}))}/_{id4_{T_2}}\} \mid$ $\{^{dec(atenc_{M_{T_2},T_2},sk(K_{A_2}))}/_{tenc_{M_{T_2},T_2}}\} \mid$ $\{^{atenc_{M_{T_2},T_2}}/_{u_{56}}\}$

Figure C.5: Static equivalence for processes $\mathbf{P}^{\mathbf{19}}_{\mathbf{term}}$ and $\mathbf{P}^{\mathbf{20}}_{\mathbf{term}}$ — task T_2 part 1

	$\mathbf{P^{19}_{term}}$		$\mathbf{P^{20}_{term}}$
Task T_2			
No.			
44	$\nu aenc_{M_{T_2},n_{nonce}}.$ $\{aenc_{M_{T_2},n_{nonce}}/y_{75}\}$	$\approx_S$	$\nu aenc_{M_{T_2},n_{nonce}}.$ $\{aenc_{M_{T_2},n_{nonce}}/y_{76}\}$
45	$\{encs(end(M_{T_2},id_{T_2}),sk(K_{A_2}))/aenc_{end,M_{T_2},T_2}\} \mid$ $\{enc(aenc_{end,M_{T_2},T_2,pub_{T_2}})/taenc_{end,M_{T_2},T_2}\} \mid$ $\{taenc_{end,M_{T_2},T_2}/y_{85}\}$	$\approx_S$	$\{encs(end(M_{T_2},id_{T_2}),sk(K_{A_2}))/aenc_{end,M_{T_2},T_2}\} \mid$ $\{enc(aenc_{end,M_{T_2},T_2,pub_{T_2}})/taenc_{end,M_{T_2},T_2}\} \mid$ $\{taenc_{end,M_{T_2},T_2}/y_{86}\}$
47	$\nu wtenc_{end,M_{T_2},T_2}.$ $\{wtenc_{end,M_{T_2},T_2}/u_{65}\}$	$\approx_S$	$\nu wtenc_{end,M_{T_2},T_2}.$ $\{wtenc_{end,M_{T_2},T_2}/u_{66}\}$
49	$\{M_{end}/x_{5end}\}$	$\approx_S$	$\{M_{end}/x_{6end}\}$

Figure C.6: Static equivalence for processes $\mathbf{P^{19}_{term}}$ and $\mathbf{P^{20}_{term}}$ — task T_2 part 2

Bibliography

[1] Martín Abadi and Bruno Blanchet. Analyzing security protocols with secrecy types and logic programs. *J. ACM*, 52(1):102–146, 2005.

[2] Martin Abadi, Michael Burrows, Butler Lampson, and Gordon Plotkin. A calculus for access control in distributed systems. *ACM Trans. Program. Lang. Syst.*, 15(4):706–734, 1993.

[3] Martin Abadi and Cedric Fournet. Mobile values, new names, and secure communication. *SIGPLAN Not.*, 36(3):104–115, 2001.

[4] Martín Abadi, Bruno Blanchet, and Cédric Fournet. Just fast keying in the pi calculus. *ACM Transactions on Information and System Security (TISSEC)*, 10(3):1–59, July 2007.

[5] Alexandre Alves, Assaf Arkin, Sid Askary, Ben Bloch, Francisco Curbera, Yaron Goland, Neelakantan Kartha, Canyang Kevin Liu, Dieter Konig, Vinkesh Mehta, Satish Thatte, Danny van der Rijn, Prasad Yendluri, and Alex Liu. OASIS - Web Services Business Process Execution Language Version 2.0 Committee Draft, May 2006.

[6] Ross J. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems.* Wiley, UK, Jan 2001.

[7] Tony Andrews, Francisco Curbera, Hitesh Dholakia, Yaron Goland, Johannes Klein, Frank Leymann, Kevin Liu, Dieter Roller, Doug Smith, Satish Thatte, Ivana Trickovic, and Sanjiva Weerawarana. *Business Process Execution Language for Web Services version 1.1.* BEA Systems, IBM Corporation, Microsoft Corporation, SAP AG, Siebel Systems, May 2003.

[8] APB. The Auditing Practices Board International Standards on Auditing (UK and Ireland) 315 - obtaining an understanding of the entity and its environment and assessing the risks of material misstatement, Dec 2004.

[9] Myrto Arapinis, Tom Chothia, Eike Ritter, and Mark Ryan. Analysing unlinkability and anonymity using the applied pi calculus. In *Proceedings of the 23rd IEEE Computer Security Foundations symposium (CSF 2010)*, pages 107–121. IEEE Computer Society, 2010.

[10] V. Atluri and W-K. Huang. An authorization model for workflows. In *Computer Security-ESORICS 96. 4th European Symposium on Research in Computer Security Proceedings*, pages 44–64, 1996.

[11] Gianfranco Balbo, Jorg Desel, Kurt Jensen, Wolfgang Reisig, Grzegorz Rozenberg, and Manuel Silva. Introductory Tutorial Petri Nets. In *Petri Nets 2000 - 21st International Conference on Application and Theory of Petri Nets*, Aarhus, Denmark, June 2000. Department of Computer Science, University of Aarhus.

[12] A.K. Bandara, E.C. Lupu, and A. Russo. Using event calculus to formalise policy specification and analysis. In *IEEE 4th International Workshop on Policies for Distributed Systems and Networks, 2003. Proceedings. POLICY 2003.*, pages 26–39. IEEE, Jun 2003.

[13] M. Y. Becker. Cassandra: Flexible trust management and its application to electronic health records. Technical Report 648, University of Cambridge Computer Laboratory, 2005.

[14] M.Y. Becker and P. Sewell. Cassandra: flexible trust management, applied to electronic health records. In *Computer Security Foundations Workshop, 2004. Proceedings. 17th IEEE*, pages 139–154, Jun 2004.

[15] D. Elliott Bell and Leonard J. LaPadula. Secure computer systems: Mathematical foundations. Technical Report MTR2547 vol1, March (reissued in electronic form 1996) 1973.

[16] D. Elliott Bell and Leonard J. LaPadula. Secure computer system: Unified exposition and multics interpretation. Technical Report MTR-2997, Mar 1976.

[17] D. Elliott Bell and Leornard J. LaPadula. Secure computer systems: A mathematical model. Technical Report MTR2547 vol2, May 1973.

[18] Elisa Bertino, Piero Andrea Bonatti, and Elena Ferrari. TRBAC: A temporal role-based access control model. *ACM Trans. Inf. Syst. Secur.*, 4(3):191–233, 2001.

[19] Elisa Bertino, Elena Ferrari, and Vijay Atluri. The specification and enforcement of authorization constraints in workflow management systems. *ACM Trans. Inf. Syst. Secur.*, 2(1):65–104, 1999.

[20] Elisa Bertino, Elena Ferrari, and Vijayalakshmi Atluri. A flexible model supporting the specification and enforcement of role-based authorization in workflow management systems. In *RBAC '97:*

Proceedings of the second ACM workshop on Role-based access control, pages 1–12, New York, NY, USA, 1997. ACM Press.

[21] Bruno Blanchet. An efficient cryptographic protocol verifier based on prolog rules. In *CSFW '01: Proceedings of the 14th IEEE Workshop on Computer Security Foundations*, page 82, Washington, DC, USA, 2001. IEEE Computer Society.

[22] Bruno Blanchet. *ProVerif Automatic Cryptographic Protocol Verifier User Manual*. CNRS, Departement d'Informatique, Ecole Normale Superieure, Paris, Feb 2007.

[23] C. Bodei, P. Degano, F. Nielson, and H. R. Nielson. Static analysis of processes for no read-up and no write-down. *Lecture Notes in Computer Science*, 1578:120–134, 1999.

[24] E.A. Boiten and M.C. Bujorianu. Exploring UML refinement through unification. In J. Jurjens, B. Rumpe, R. France, and E.B. Fernandez, editors, *Critical Systems Development with UML - Proceedings of the UML'03 workshop*, number TUM-I0323, pages 47–62. Technische Universitat Munchen, September 2003.

[25] R. A. Botha and J. H. P. Eloff. Access control in document-centric workflow systems - an agent-based approach. *Computers & Security*, 20(6):525–532, 2001.

[26] Reinhardt A. Botha. *CoSAWoE - A Model for Context-sensitive Access Control in Workflow Environments*. PhD thesis, Faculty of Natural Sciences, Rand Afrikaans University, Nov 2001.

[27] Reinhardt A. Botha and Jan H. P. Eloff. Separation of duties for access control enforcement in workflow environments. *IBM Syst. J.*, 40(3):666–682, 2001.

[28] Chiara Braghin, Daniele Gorla, and Vladimiro Sassone. Role-based access control for a distributed calculus. *Journal of Computer Security*, 14:113–155, 2006.

[29] David D. Clark and David R. Wilson. A comparison of commercial and military computer security policies. *sp*, 00:184, 1987.

[30] William R. Cook, Sourabh Patwardhan, and Jayadev Misra. Workflow patterns in orc. In *Coordination Models and Languages*, volume 4038 of *LNCS*, pages 82–96. Springer Berlin/Heidelburg, 2006.

[31] Jason Crampton. A reference monitor for workflow systems with constrained task execution. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 38–47, New York, NY, USA, 2005. ACM.

[32] Jason Crampton and Hemanth Khambhammettu. Delegation and satisfiability in workflow systems. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 31–40, New York, NY, USA, 2008. ACM.

[33] N. Damianou, A. Bandara, M. Sloman, and E. Lupu. A survey of policy specification approaches. 2002. Department of Computing, Imperial College of Science Technology and Medicine, London.

[34] Nicodemos Damianou, Naranker Dulay, Emil Lupu, and Morris Sloman. The ponder policy specification language. In *Proc. Policy 2001: Workshop on Policies for Distributed Systems and Networks*, pages 18–39. Springer-Verlag, 2001.

[35] Stéphanie Delaune, Steve Kremer, and Mark Ryan. Verifying privacy-type properties of electronic voting protocols. In *Journal of Computer Security*, pages 435–487. IOS Press, 2009.

[36] John Derrick, David Akehurst, and Eerke Boiten. A framework for UML consistency. In L. Kuzniarz, G. Reggio, J. L. Sourrouille, and Z. Huzar, editors, *2002 Workshop on Consistency Problems in UML-based Software Development*, pages 30–45, October 2002.

[37] John DeTreville. Binder, a logic-based security language. Technical Report MSR-TR-2002-21, Microsoft Research, Microsoft Corporation One Microsoft Way Redmond, WA 98052, Mar 2002.

[38] DoD. DoD 5200.28-STD Department of Defense Trusted Computer System Evaluation Criteria, Dec 1985.

[39] Marlon Dumas and Arthur H. M. ter Hofstede. Uml activity diagrams as a workflow specification language. In *Proceedings of the 4th International Conference on The Unified Modeling Language, Modeling Languages, Concepts, and Tools*, volume 2185 of *LNCS*, pages 76–90, London, UK, 2001. Springer-Verlag.

[40] eHI. Junior doctors' confidential details openly displayed. *eHealth Insider Primary Care*, Apr 2007.

[41] Carl Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian Thomas, and Tatu Ylonen. SPKI certificate theory - IETF RFC 2693, Sep 1999.

[42] U. Engberg and M. Nielsen. A calculus of communicating systems with label-passing. Technical report, Department of Computer Science, Aarhus University, May 1986.

[43] R. Eshuis and J. Dehnert. Reactive Petri nets for workflow modeling. *Application and Theory of Petri Nets 2003, LNCS*, 2679:295–314, 2003.

[44] Rik Eshuis and Akhil Kumar. An integer programming based approach for verification and diagnosis of workflows. *Data Knowl. Eng.*, 69(8):816–835, August 2010.

[45] E.U. Directive 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, Oct 1995.

[46] D. F. Ferraiolo, J. F. Barkley, and D. R. Kuhn. A role-based access control model and reference implementation within a corporate intranet. *ACM Transactions on Information and System Security(TISSEC)*, 2(1):34–64, Feb 1999.

[47] D. F. Ferraiolo and D. R. Kuhn. Role based access control. In *Proceedings of 15th National Computer Security Conference*, 1992.

[48] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security(TISSEC)*, 4(3):224–274, Aug 2001.

[49] D. P. Guelev, M. Ryan, and P. Y. Schobbens. Model-checking access control policies. *Seventh Information Security Conference (ISC'04). Lecture Notes in Computer Science*, 3225:219–230, 2004.

[50] J. Y. Halpern and V. Weissman. Using first-order logic to reason about policies. In *Proceedings of the Computer Security Foundations Workshop (CSFW'03)*, 2003.

[51] H.M.Government. Data Protection Act 1998, 1998.

[52] C. A. R. Hoare. Communicating sequential processes. *Commun. ACM*, 21(8):666–677, 1978.

[53] P. Huber, K. Jensen, and R. M. Shapiro. Hierarchies in coloured petri nets. In *Proceedings of the 10th International Conference on Application and Theory of Petri Nets, 1989, Bonn*, pages 192–209, 1989. not read - abstract only.

[54] K. Jensen. An Introduction to the Theoretical Aspects of Coloured Petri Nets. *A Decade of Concurrency, Lecture Notes in Computer Science*, 803:230–272, 1994.

[55] James B. D. Joshi, Basit Shafiq, Arif Ghafoor, and Elisa Bertino. Dependencies and separation of duty constraints in GTRBAC. In *SACMAT '03: Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 51–64, New York, NY, USA, 2003. ACM Press.

[56] D. Kitchin, W.R. Cook, and J. Misra. A language for task orchestration and its semantic properties. In *Proc. of the Intl. Conf. on Concurrency Theory (CONCUR)*, volume 4137 of *Springer LNCS*, pages 477–491. Springer-Verlag, Berlin, 2006.

[57] Steve Kremer and Mark Ryan. Analysis of an electronic voting protocol in the applied pi calculus. In *ESOP*, pages 186–200, 2005.

[58] Lars M. Kristensen, Soren Christensen, and Kurt Jensen. The practitioner's guide to coloured petri nets. *International Journal on Software Tools for Technology Transfer*, 2:98–132, 1998.

[59] Butler W. Lampson. "Protection," Proc. Fifth Princeton Symposium on Information Sciences and Systems, Princeton University, March 1971, pp. 437-443 reprinted in. *SIGOPS Oper. Syst. Rev.*, 8(1):18–24, Jan 1974.

[60] N. Li and J. Mitchell. RT: A role-based trust-management framework. In *Third DARPA Information Survivability Conference and Exposition (DISCEX III)*, 2003.

[61] Ninghui Li, Ziad Bizri, and Mahesh V. Tripunitara. On mutually-exclusive roles and separation of duty. In *CCS '04: Proceedings of the 11th ACM conference on Computer and communications security*, pages 42–51, New York, NY, USA, 2004. ACM Press.

[62] Ninghui Li and John C. Mitchell. Datalog with Constraints: A Foundation for Trust Management Languages. In *PADL 03: Proceedings of the 5th International Symposium on Practical Aspects of Declarative Languages*, pages 58–73. Springer-Verlag, 2003.

[63] Ninghui Li, John C. Mitchell, and William H. Winsborough. Design of a role-based trust management framework. In *Proc. IEEE Symposium on Security and Privacy, Oakland*, May 2002.

[64] Ninghui Li, William H. Winsborough, and John C. Mitchell. Distributed credential chain discovery in trust management: extended abstract. In *ACM Conference on Computer and Communications Security*, pages 156–165, 2001.

[65] Robin Milner. Functions as processes. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*, pages 167–180, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[66] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, i. *Inf. Comput.*, 100(1):1–40, 1992.

[67] Robin Milner, Joachim Parrow, and David Walker. A calculus of mobile processes, ii. *Inf. Comput.*, 100(1):41–77, 1992.

[68] Michael J. Nash and Keith R. Poland. Some conundrums concerning separation of duty. *IEEE Symposium on Security and Privacy*, pages 201–207, May 1990.

[69] NIST. American National Standard for Information Technology - Role Based Access Control ANSI INCITS 359-2004, Feb 2004.

[70] Matunda Nyanchama and Sylvia L. Osborn. Access rights administration in role-based security systems. In *Proceedings of the IFIP WG11.3 Working Conference on Database Security VII*, pages 37–56, Amsterdam, The Netherlands, The Netherlands, 1994. North-Holland Publishing Co.

[71] John Oates. DoH's latest d'oh! *The Register*, April 2007.

[72] Eyal Oren and Armin Haller. Formal frameworks for workflow modelling. Technical Report DERI Technical Report 2005-04-07, DERI - Digital Enterprise Research Institute, DERI Galway, University Road, Galway, IRELAND, Apr 2005.

[73] Lewis Page. US service personnel at risk of ID theft. *The Register*, July 2007.

[74] Carl Adam Petri. *Kommunikation mit Automaten.* PhD thesis, Institut für Instrumentelle Mathematik, Bonn, 1962. not read.

[75] Michael R. Phillips. Employees Continue to Be Susceptible to Social Engineering Attempts That Could Be Used by Hackers. Technical Report 2007-20-107, US Treasury Inspector General for Tax Administration, Washington, D.C. 20220, Jul 2007.

[76] Corrado Priami, Aviv Regev, Ehud Shapiro, and William Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80(1):25–31, October 2001.

[77] Frank Puhlmann and Mathias Weske. Using the pi-calculus for formalizing workflow patterns. In *Third International Conference in Business Process Management(BPM 2005)*, volume 3649 of *Springer LNCS*, pages 153–168. Springer Verlag, Berlin Heidelburg, Sep 2005.

[78] Hasan Qunoo and Mark Ryan. Modelling dynamic access control policies for web-based collaborative systems. In *Proceedings of the 24th annual IFIP WG 11.3 working conference on Data and applications security and privacy*, DBSec'10, pages 295–302, Berlin, Heidelberg, 2010. Springer-Verlag.

[79] N. Russell, A.H.M. ter Hofstede, W.M.P. van der Aalst, and N. Mulyar. Workflow Control-Flow Patterns: A Revised View. *BPM Center Report*, (BPM-06-22), 2006.

[80] P.Y.A. Ryan, S.A. Schneider, M.H. Goldsmith, G Lowe, and A.W. Roscoe. *The Modelling and Analysis of Security Protocols: the CSP Approach.* Pearson Education Limited, UK, 2010.

[81] SAIC. SAIC Addresses Possible Data Compromise. *SAIC News Release*, Jul 2007.

[82] Ravi Sandhu, David Ferraiolo, and Richard Kuhn. The NIST model for role-based access control: Towards a unified standard. In *Proceedings of 5th ACM Workshop on Role-Based Access Control*, pages 47–64, Jul 2000.

[83] 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.

[84] Richard Simon and Mary Ellen Zurko. Separation of Duty in Role-based Environments. In *CSFW '97: Proceedings of the 10th IEEE workshop on Computer Security Foundations*, page 183, Washington, DC, USA, 1997. IEEE Computer Society.

[85] Howard Smith and Peter Fingar. Workflow is just a Pi process. *BPTrends*, Jan 2004.

[86] R. K. Thomas and R. S. Sandhu. Towards a task-based paradigm for flexible and adaptable access control in distributed applications. In *NSPW '92-93: Proceedings on the 1992-1993 workshop on New security paradigms*, pages 138–142, New York, NY, USA, 1993. ACM Press.

[87] U.S.Government. Sarbanes-Oxley Act of 2002, Jan 2002.

[88] W. M. P. van der Aalst. Making Work Flow: On the Application of Petri nets to Business Process Management. In J. Esparza and C. Lakos, editors, *Application and Theory of Petri Nets*, volume 2360 of *Springer LNCS*, pages 1–22. Springer-Verlag, Berlin, 2002.

[89] Wil van der Aalst. Petri nets refresher, 2005.

[90] Wil M.P. van der Aalst. Why workflow is NOT just a Pi-process. *BPTrends*, Feb 2004.

[91] W.M.P. van der Aalst. *Timed coloured Petri nets and their application to logistics.* PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands, 1992.

[92] W.M.P. van der Aalst. Pi calculus versus petri nets: let us eat humble pie rather than further inflate the pi hype. 2003.

[93] W.M.P van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros. Workflow Patterns. *Distributed and Parallel Databases*, 14(3):5–51, July 2003.

[94] W.M.P. van der Aalst, K.M. van Hee, and G.J. Houben. Modelling workflow management systems with high-level petri nets. In G. De Michelis, C. Ellis, and G. Memmi, editors, *Proceedings of the second Workshop on Computer-Supported Cooperative Work, Petri nets and related formalisms*, pages 31–50, 1994.

[95] K. M. van Hee, L. J. Somers, and M. Voorhoeve. Executable specifications for distributed information systems. In E. D. Falkenberg and P. Lindgreen, editors, *Proceedings of the IFIP TC 8*, pages 139–156. WG 8.1 Working Conference on Information System Concepts: An In-depth Analysis, Elsevier Science Publishers, Amsterdam, 1989. not read.

[96] Various. bpel xml.org. Website, March 2011.

[97] Various. Workflow management coalition (wfmc). Website, March 2011.

[98] Jacques Wainer, Paulo Barthelmess, and Akhil Kumar. W-rbac - a workflow security model incorporating controlled overriding of constraints. *Int. J. Cooperative Inf. Syst.*, 12(4):455–485, 2003.

[99] Jacques Wainer and Akhil Kumar. A fine-grained, controllable, user-to-user delegation method in RBAC. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 59–66, New York, NY, USA, 2005. ACM Press.

[100] Jacques Wainer, Akhil Kumar, and Paulo Barthelmess. Dw-rbac: A formal security model of delegation and revocation in workflow systems. *Inf. Syst.*, 32(3):365–384, 2007.

[101] WFMC. Workflow Management Coalition Terminology and Glossary Issue 3.0. Technical Report WFMC-TC-1011, Workflow Management Coalition, Workflow Management Coalition, 2 Crown Walk, Winchester, Hampshire, UK, Feb 1999.

[102] Stephen A. White. Process Modeling Notations and Workflow Patterns. *BPTrends*, March 2004.

[103] Cristian Radu Zervos. *Colored Petri Nets: Their Properties and Applications.* PhD thesis, University of Michigan, Michigan, USA, 1977. Not read.

[104] N. Zhang. *Generating Verified Access Control Policies through Model-Checking.* PhD thesis, School of Computer Science, The University of Birmingham, 2005.

[105] Xinwen Zhang, Francesco Parisi-Presicce, Ravi Sandhu, and Jaehong Park. Formal model and policy specification of usage control. *ACM Trans. Inf. Syst. Secur.*, 8(4):351–387, 2005.