Access Control on Provenance

By

Xinyu Fan

A thesis submitted to Macquarie University

for the degree of Master of Research

Department of Computing

October 2018

Typeset in LaTeX 2_ε.

Except where acknowledged in the customary manner, the material presented in this thesis is, to the best of my knowledge, original and has not been submitted in whole or part for a degree in any university.

Xinyu Fan

Acknowledgements

First of all, I am very grateful to my supervisors, Prof. Jian Yang and A/Prof. Michael Hitchens, who guided me to explore cyber security. They were always encouraging me to pursue research. I also would like to particularly express my gratitude to my associate supervisor A/Prof. Michael Hitchens. His professional knowledge in the field of access control has been of enormous benefit me in the establishment of my PhD project.

I must also express my gratitude to my parents who have always supported me in my research. Their continuing encouragement gave me the courage to overcome obstacles that I encountered during this period.

Of enormous value throughout the creation of this thesis was the excellent academic environment and professional support provided by administrative staff and the technical support staff of the Computing Department at Macquarie University, including Tracy Rushmer, Jane Yang, Yan Wang and others.

List of Publications

- Xinyu Fan, Vijay Varadharajan, Michael Hitchens *Provenance Based Classification Access Policy System Based on Encrypted Search for Cloud Data Storage*, ISC2015:283-298, 2015.

Abstract

Though provenance has long played a major role in the context of art and archaeology (in terms of lineage or pedigree), more recently it has become more important for data in various sectors such as finance and medicine. It is not just about the origin or creator of data but also what sort of operations have been performed by whom and in what context, especially when it comes to security and privacy. As provenance research on security stays at its initial stage, some open problems and research challenges have been identified for provenance research, specifically in terms of security issues. Access control involving provenance is treated as a primary security issue, and is the main area to which we are trying to contribute. Integrity and non-repudiation should also be ensured for provenance.

This thesis mainly focuses on preserving the security of provenance as well as utilising provenance as conditions to control proper access to data. The contributions of the thesis are illustrated as follows:

We propose three frameworks of access control policies on provenance. *The Partition-based Access Control Policy Language on Provenance* is tailored based on our extended provenance model (OPM^+). The fine-grained policies determine access for provenance, based on our defined provenance partitions instead of whole provenance graphs. Moreover, Algorithms for merging policy results and transferring provenance graphs according to policy results are provided as well. Following this, *The Provenance-based Access Control policies* employs provenance partitions as conditions to evaluate accessibility for data. Our proposed policies distinguish different types of attributes extracted from provenance, where the result

of each policy is a value in the “four-valued" decisions set. Policy algebras for the “four-valued" decision set are tailored accordingly. Further, to provide a comprehensive scope for access control policies involving provenance, *Purpose-based Access Policies on provenance* are proposed. This defines allowed/prohibited access purposes for data based on attributes in provenance. A series of corresponding internal and external policy algebras is provided to merge purpose sets.

We also provide two cryptographic schemes to implement access control policies involving provenance. One scheme implements Provenance-based file Classification Policies which sort files based on given provenance partitions (keywords) from their provenance. The scheme enables to search given provenance partitions in the ciphertext of provenance as well as check authentication of users. The other scheme is derived from attribute-based access control encryption schemes. It allows data owners to encrypt data based on Provenance-based Access Control policies.

Contents

List of Figures

List of Tables

We will bankrupt ourselves in the vain search for absolute security.

Dwight D. Eisenhower

1 Introduction

In the Oxford English Dictionary, provenance is defined as: *(1) the fact of coming from the particular source or quarter; origin, derivation; (2) the history or pedigree of a work of art, manuscript, rare book, etc.; concretely, a record of the ultimate derivation and passage of an item through its various owners.* In computer systems, the provenance of a piece of data logs processes and operations which result in the piece of data and is relevant to the source and origins. Provenance can be expressed as a directed acyclic graph (DAG), illustrating how a data artifact is processed by an execution. In such a DAG of provenance under the Open Provenance Model (OPM)[1], nodes present three main entities including *Artifact*, *Agent* and *Process* and edges represent connections to the main entities.

Provenance is a type of data can be utilised in the Cloud and database. The advance of Cloud and database technology has aroused the concerns of privacy. The current database technology enables to collect and store vast person-specific data. Although privacy violations directly abuse the benefits of customers, most enterprises and institutes are concerned about the issues of data privacy and security. Some large companies, including IBM, Google,

demonstrating trusted privacy applications, aiming to attract more potential users.

Provenance has been utilised for various purposes, including estimating data quality and data reliability via providing data derivation, tracing the audit trail of data to detect errors in data generation, establishing the copyright and ownership of data to determine liability in case of erroneous data *etc.*

However, some open problems and research challenges have been identified for provenance research in term of security issues[11]. Access control on provenance is treated as the primary security issue and is the main area to which we are trying to contribute. Integrity and non-repudiation should also be ensured for provenance as a piece of data and its connecting provenance might be generated by multi-owners. Historical operations logged in provenance are facts which cannot be altered. Hence, the integrity of provenance without unintentional or malicious manipulation should be ensured in provenance-aware systems. Moreover, derivation of authorisation information and context-based authorisation specifications are also open problems for security issues of provenance. But this thesis focuses on proposing fine-grained access control policies and cryptography schemes to implement the policies.

1.1 Motivation

The goal of the thesis is to protect the security of provenance and employing provenance to preserve the security of data.

Assume such a scenario regarding the applications of access control policies. Data and provenance are stored in a database, and the database server and data owners define access control policies to decide whether a request can be accepted or denied. The policies are expected to introduce items recorded in provenance as conditions or restrictions. How to define fine-grained access control policies under a proper provenance model is the main concern of this thesis.

Access control to provenance cannot be extended directly from the access decision of a piece of data to its provenance, as the sensitivity of provenance is possibly different from the data from which it derives. For instance, there is a provenance graph logs regarding how a project was established by recording who did what operations. Even though the project can be published, the provenance regarding the techniques and strategies to construct the project

should be kept as a secret.

In addition, existing traditional access control techniques including role-based access control[12], attribute-based access control [13] cannot be applied to provenance access control straightforwardly, because of provenance is a type of meta-data with a specific data structure. The entities are linked by dependencies to show the relationship between the entities. Hence, to provide fine-grained access control policy frameworks involving provenance, more contributions are explored in this thesis.

Provenance logging historical performances executed on a piece of data reveals the sensitivity of the data. Hence it can be employed as conditions of policies to control access to the data. Traditional access control policies usually utilise current status as conditions, while provenance-based access control makes access decisions based on operations logged in provenance which proposes more fine-grained access control policies.

The whole thesis is developed upon our self-define provenance model OPM$^+$, which stores attributes with the vertex in provenance graphs. OPM$^+$ supports more fine-grained access control policies.

As the existing work [2][14][15][16][17][18] cannot define provenance subgraphs properly, a proper fine-grained access control policy to control access to provenance should be proposed. Details including how to transform provenance graphs by removing unallowed partitions are lacking from existing work.

Moreover, provenance contains labels between the three main entities under the OPM, which needs to be particularly handled for access control, especially when an access control policy determines that certain subgraphs of a provenance can be accessed instead of the whole graph. As a provenance graph continually develops along with the changes of the data, access control policies of provenance are usually generated before all provenance graphs are produced, where specific names of the subjects and objects cannot be predicted. Hence, access control policies need to be tailored at an abstract level. This can also reduce redundancy and contributes to the re-use of policies.

In addition, policy algebras to merge results of individual policies for a request should be handled carefully for access control policies involving provenance. For a request to access a piece of data, it is possible to retrieve many provenance-based access control policies to make access decisions. As a result, how to combine results of policies should be tailored

especially for unique fine-grained policy languages or models.

Access control policies are not restricted to only making decisions to determine whether a piece of data can be accessed or not. Policies can also define which purposes a piece of data can be accessed. In another word, privacy policies to confine eligible access purposes under certain conditions are crucial access control mechanisms.

Finally, simply tailoring access control policies cannot address the security issues on provenance access control thoroughly, and corresponding cryptographic schemes to implement the policies are also required to complete the project completely.

1.2 Contributions

In this thesis, we propose a comprehensive scope of access control policies involving provenance and cryptography schemes implementing the policies. The policies can be applied to databases and Cloud servers. Provenance partitions are employed as conditions or results, which support more fine-grained policies. The operations performed on data can be introduced as restrictions for all types of policies proposed in this thesis. We provide three types of policies which server for different purposes.

1. PACLP: A Partition-Based Access Control Policy Language for Provenance.

A partition-based access control policy language is proposed in Chapter 3, which is more fine-grained comparing with existing work and defines provenance partitions at a summarised level. The policies determine which provenance partitions can be accessed and which partitions cannot be accessed. In this framework, we tailored and defined language elements to support more fine-grained policy language, which is lacking previous work. For instance, PACLP for the first time defines how to transform provenance partitions in order to hide information which is not allowed by the policies, in order to respond an access request. The contributions in details are listed as follows:

- We extend the OPM by proposing the OPM$^+$ which stores attribute for entities in a provenance DAG. Hence, OPM$^+$ support more fine-grained policies. Namely, vertices can be defined and searched over their attaching attributes. For example, a policy can

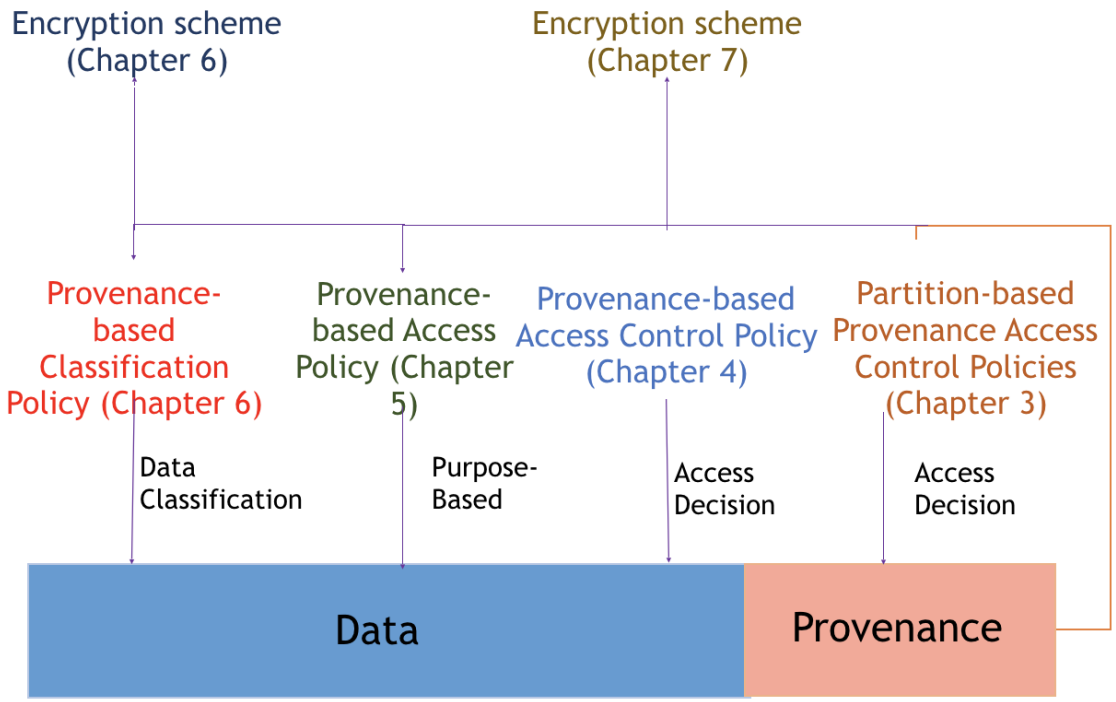

Figure 1.1: the overall conceptual framework for all the contributions

define a collection of vertices which are attached with a timestamp as 10/6/2017. Thus, vertices do not have to be enumerated in policies.

- We define several types of provenance partitions under OPM$^+$, which can be expressed over XPath. In PACLP, provenance partitions are utilised as conditions or results.
- We propose PACLP by extending existing policy languages. PACLP enables the specification of partial provenance graphs as well as transformation scope, transformation mode, and transformation labels, in order to partition a provenance graph in a fine-grained approach and define how to transform vertices in order to return a new subgraph for the access requestors.
- Algorithms to retrieve applicable policies for a request and merge results of individual policies as a final decision are proposed. In addition, we present algorithms to transform a targeted provenance DAG into a new subgraph which can be returned to the requestors.

2. **Provenance-based Access Control Policies**.

Following PACLP, we propose a sound fine-grained provenance-based access control policy model which determines whether data can be accessed based on its provenance. Namely, it employs provenance as conditions. The object for accessing this policy model is data, instead of provenance graphs. This policy model is tailored with a “four-valued" decision set, which means it supports partial matching scenarios, including 0/3 matching, 1/3 matching, 2/3matching, 3/3 matching. The framework consists of a target section, policy evaluation, and policy combination. Target language specifies which requests are matching to the policy. Namely, when a query is received, target section checks if the policy is applicable to the query. Then, the policy evaluates applicable queries and produce a result. All results from applicable policies are combined based on a logical structure.

We propose several types of atomic targets (conditions). Path atomic target only uses elements from provenance. Namely, the decision is determined by whether given operations have been performed on the data. Associated atomic target takes attributes from both provenance and requests, which implies the determining is based on whether the requestors have performed operations on the data. For instance, let an atomic target be: if the requestor previously edited data but did not submit it, the access request can be allowed.

Moreover, following a mechanism of policy tree structure [19], we replace the attribute triple as provenance partitions. To cooperate with the “four-valued" decision set, we re-arrange logic operators to merge results under the “four-valued" decision set.

3. Purpose-based Access Control Policies on Provenance.

In order to establish a comprehensive scope of policies involving provenance, we propose a framework for purpose-based access policy. For the first time, we propose an access policy model to employ provenance partitions as conditions to determine access purposes. Firstly, we define the semantics and syntax of atomic privacy policies which map conditions to a set of permitted or prohibited usages.

Even though purposes were classified based on various sensitivities in previous work. Algebras which distinctively combine purposes in different sensitivity levels were not previously proposed. Therefore, we particularly create functions for algebras of the policies involving purposes. Our policy algebras merge purposes in different groups by different logic operators. In addition, both internal algebras and external algebras of this framework are presented.

4. Provenance-based Classification Access Policy System based on Encrypted Search for Cloud Data Storage.

In Chapter 6, a scheme to implement provenance-based file classification policy is proposed. Under the assumption of the system model of this work, policies define how to classify files based on their attached provenance by searching keywords from provenance. We propose a scheme to implement this policy. The scheme allows the Policy Decision Server to check the encrypted provenance without decrypting the provenance. While at the same time, it provides guarantees to the Policy Decision Server that the provenance is from a genuine source. Such a solution will enable authenticated and confidential provenance information to be used in the access control service without revealing its plain content. To achieve such a solution, we introduce a new notion of Encrypted Provenance Search Scheme (EPSS). EPSS is based on the searchable encryption method proposed by Boneh *et al.*[20].

5. Provenance-based partitioned encryption for fine-grained access control in Cloud Computing.

Finally, a scheme to implement provenance-based access control policies is proposed. Provenance-based Encryption (PBE) extends from ciphertext policy attribute-based access control (CP-ABE)[21], which implements provenance-based access control policies.

We propose an assumption to implement provenance-based access control policies in the Cloud. In the Cloud, a file could be generated by several data owners. Each data owner generates an access control policy for the section of data she contributes. As the access control policies are based on the provenance. PBE is able to generate keys and encrypt sections of a file to implement the policies. Concretely, the scheme proposes a more efficient approach for calculation.

1.3 Roadmap of the Thesis

The thesis is structured as follows:

Chapter 2 proposes the literature review of references relevant to the research program,

including provenance background and models, access control policies, cryptographic techniques.

Chapter 3 presents a framework for a partition-based access control policy language for provenance. The policy language enables the tailoring of access control policies for provenance, in order to preserve the security and privacy of provenance. Particularly, the policy is able to define which provenance partitions can or can not be accessed instead of making access decisions based on a whole provenance graph. Three algorithms to implement the policies and transform provenance DAG based on the results of policies are also presented.

Chapter 4 proposes a fine-grained policy model for provenance-based access control, which evaluates the accessibility of data based on provenance. Because provenance records historical operations performed on provenance provide clues for the sensitivities and vulnerabilities of provenance. Moreover, corresponding data algebras for the policies is proposed as well.

Chapter 5 presents purpose-based privacy policies on provenance which decide permitted or prohibited access purposes based on provenance. Particularly, the purposes are classified as different levels based on the sensitivities. Accordingly, functions to merge purposes sets generated for individual policies are tailored as well, which includes both internal and external policy algebras.

In Chapter 6, we propose a scheme to implement Provenance-based file classification policies. The policies classify files based on the keywords from provenance. The scheme enables a server to search keywords in encrypted provenance without decoding it. In addition, the scheme could also confirm the authentication of data owners.

In Chapter 7, we present a scheme to encrypt files based on access control policies. The scheme operates under the security assumption that each data owner generates provenance-based access control policies for the section of data they contribute. The scheme facilitates a fast computation when the access control structures of sections in a file are similar.

Finally, Chapter 8 concludes the thesis with proposed further work in the future.

2

Literature Review

2.1 Introduction

Provenance is a document recording how a piece of data attains its current status, which can be used to understand the process by which data was obtained and transformed[22]. It can be employed for estimating data quality and reliability, detecting data errors, establishing the copyright, *etc.*

In terms of access control issues related to provenance, there are two main branches. They are provenance access control and provenance-based access control. As metadata, there are also security and privacy concerns associated with provenance itself, including the integrity, confidentiality, and availability of provenance information. Particularly, provenance access control determines accessibility to provenance in order to preserve the security of provenance; provenance-based access control utilises provenance as conditions to estimate the accessibility to the data.

Since provenance is itself data (metadata), it seems that the access control of provenance is already addressed by a vast amount of existing work. But, existing access control techniques cannot be applied straightforwardly due to specific data structures of provenance. Challenges regarding access control issues regarding provenance are surveyed[23][24][25], a discussion is mainly developed revolving bridging the gaps between provenance and normal data.

Because the specific data structures of provenance are the main reason that most existing access control techniques cannot be applied directly, in this chapter, we firstly review several popular provenance models which are followed by provenance access control policies, provenance-based access control policies *etc*. In these proposed work, access control policies are tailored by considering the specific data structure of provenance. Moreover, policy algebras which merge results of individual policies are also explored in this chapter.

In addition, cryptography can also contribute to access control regarding provenance. Cryptography schemes tailored especially for provenance related access control policies are also proposed in this thesis. Hence, we also reviewed some cryptographic schemes which are generated to execute access control. Provenance-based access control can borrow techniques of attribute-based access control by extracting attributes from provenance. Hence, attribute-based access control encryption schemes are surveyed in this chapter. Moreover, we also explored a series of schemes on keyword search techniques. To preserve the confidentiality of provenance, we assume the provenance stored in the Cloud is encrypted. In order to implement Provenance-based Access Control policies which employ attributes in provenance as conditions, a proper scheme which enables the cloud server to search the given attributes defined in access control policies from the ciphertext of provenance is expected.

In this chapter, the reviewing literature is organised as follows:

- Section 2.2 introduces provenance background and provenance models which are addressed for access control purpose.
- Section 2.3 reviews a series of access control policies or policy models relevant to provenance access control, provenance-based access control and policy algebras.
- Section 2.4 surveys cryptographic schemes and techniques, where we explored the topics on encryption schemes for provenance, keyword searching schemes and attribute-based encryption schemes, in order to explore techniques to construct implementations for our proposed policies.

- Section 2.5 points out the connection between the references and our own contributions.

2.2 Provenance Background and Models

In this section, we introduce the background of data provenance, including a definition, application and several provenance models.

Though provenance has long played a major role in the context of art and archaeology (in terms of lineage or pedigree), more recently it has become more important for data in various sectors such as finance and medicine[25][26][27][28]. It is not just about the origin or creator of data but also what sort of operations have been performed by whom and in what context, especially when it comes to security and privacy. As such, representation of data provenance provides information about the ownership as well as actions and modifications which have been performed on the data. In terms of access control, characteristics such as data accuracy, timeliness and the path of transfer of data are important. For instance, with the Sarbanes-Oxley Act[29], the consequences for signing incorrect corporate financial statements became dire. Formal penalties for non-compliance with SOX can include fines, removal from listings on public stock exchanges and invalidation of D&O insurance policies. Therefore it is important to keep track of data which contributes to financial reports and to authenticate the people who worked on them.

A consensus began to emerge at the International Provenance and Annotation Workshops ($IPAW'06$), where over 50 participants who were interested in the issues of data annotations, data derivation, data provenance and process documents. Provenance research increasingly has developed at following IPAWs to understand better the representations for provenance, rationales motivating designs and research challenges of provenance [30][31][32][33][34]. At IPAWs, participators discuss and develop new ideas and explore connections between disciplines and between academic research on provenance and practical applications. Different techniques and provenance models have been proposed in many areas such as workflow systems, visualization, databased, digital libraries, and knowledge representation. An important issue the focus on is how to integrate these techniques and models so that complete provenance can be derived for complex data products.

In recent years, defining provenance models [1][2][14] which are adopted by various systems is a topic of research. Moreau *et al.*[1] provides Open Provenance Model (OPM)

which could be presented as a form of a directed acyclic graph (DAG). It consists of five main entities as nodes and dependencies as edges and records processes actioned on data. The three main entities are “Artifact", “Process" and “Agent" respectively, and each edge represents a causal dependency. Specifically, *Artifact* is a piece of data which may be a physical object or a digital representation stored in computers; *Process* is an action affecting the artifact and creating new artifacts; *Agent* is an entity enabling and controlling the process. These main entities act as nodes in the provenance directed acyclic graph. Dependencies describe the relationships between these entities and connect the nodes in the graph. Figure 1 shows the five main dependencies. They are “used" (Process used Artifact); “wasGeneratedBy" (Artifact was generated by Process); “wasControlledBy" (Process was controlled by Agent); “wasTriggeredBy" (Process2 was triggered by Process 1); “wasDerivedFrom" (Artifact2 was derived from Artifact1).

OPM meets the requirement allowing provenance information to be exchanged between systems, and it is possible for developers to build and share tools operating on OPM. Currently, it has been employed by some provenance-aware systems. Ni *et al.* [2] provides another provenance model that presents a set of provenance records. It defines five kinds of records which are operational records, message records, actor records, preference records, and context records and each record consists of several attributes. However, these attributes are optional in that the record might be null. To carry contextual information such as time, temporal aspects, user ID and so on, Nguyen *et al.* [14] presents an extensional model of OPM by adding attributes data of a transaction to vertex “Action" of each transaction. For instance, this attribute information supports extra access control policies of Dynamic Separation of Duties (DSOD) and there are other proposals [15][16][17][18] focusing on specific application domains such as electronic health data and scientific records.

However, OPM only records three main entities and dependencies between the main entities and these cannot support more fine-grained access control policies. This is because, attributes of *process*, *agent*, *artifact* are crucial conditions for access control. Hence, in this thesis, we extend OPM by proposing OPM$^+$. In OPM$^+$, each of the main entities is optionally associated with an attribute set, where these attributes can be employed as restrictions in our proposed policies.

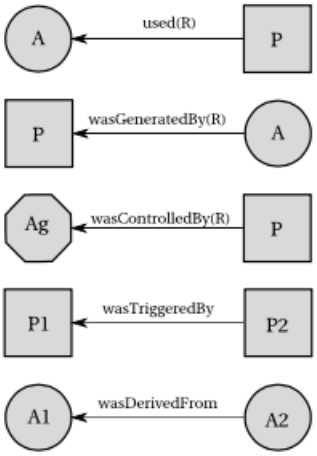

Figure 2.1: Dependencies in Provenance Model [1]

2.3 Access-Control Policy

After showing general definitions and usages of access control, a few access matrix implementation approaches and basic access control models are reviewed in this section, including Basic Access Control Policy Models, Provenance Access Control, Provenance-Based Access Control, and policy algebras.

2.3.1 Basic Access Control Policy Models

We begin the exploration with basic access control policy models including Access control lists (ACLs)[12], capability, authorisation relations, mandatory access control (MAC), discretionary access control (DAC) and Role-Based Access Control (RBAC)[13]. The basic policy models are foundations of the access control theoretical system.

Access Control constrains what a user can do and which objects they are allowed to access [12]. This satisfies access control goals to restrain activities that could lead to a breach of security. In a system, a reference monitor mediates each attempted access by a visitor (or program executed on behalf of the visitor) to objects. The reference monitor accesses an authorisation database for the purpose of deciding if the visitor attempting to do an operation is authorised to perform, where the authorization based on security policies of an organisation are administered and maintained by the security administrators. However, users might also be able to modify some authorisation in the database, for example, to define permissions for their owned files. In addition, the relevant activity might be recorded by auditing monitors,

as access control is not a complete solution to secure a system, it needs to be coupled with auditing.

As a conceptual model, the access matrix specifies that each subject possesses which rights each subject possesses for each object. In this matrix, rows are subjects and columns are for objects. Each cell of access matrix defines the authorised accessibility for the subject in the row and the object in the column. In systems, the numbers of access matrices might be enormous and there are a few approaches to implementing access matrices in systems. ACL is a popular approach, where each object is associated with an ACL. It indicates the access authorised to execute for each subject. ACL corresponds to store matrices by columns and checks which operations are authorised for the object. Similarly, one capability is a twin approach to ACLs: every user or subject is attached with a list indicating, in a system, which accesses this subject is authorised to operate in the object. It stores access matrices by rows and provides an easy review for all authorised accesses of objects. However, paired advantages and disadvantages of ACLs and capability are seen with respect to access review. Hence, authorisation relation provides representations of the access matrix not favouring one aspect of access review over the other, where each row in this table presents one access right of a subject to an object.

Traditional access control models include MAC with a relatively long history and DAC [13]. MAC is based on predetermined compartments associated with each classification of subjects and clearance of objects. In a system, every object and every user are assigned a security level. The security level assigned to an object represents the sensitivity of the content of the object, as unauthorised disclosure of the information could lead to potential damage. The subject's security level which is also called clearance reflects the user's trustworthiness. In the simplest case, for military and civilian government arenas, the set hierarchically consists of Top Secret(TS) > Secret (S) > Confidential (C) > Unclassified (U). The Bell-La Padula model [35] is a classic MAC example that presents a model of multi-level security, where information confidentiality is the main focus. In particular, the two principles are required to prevent information in high-level objects to flow to objects at lower levels, as follows:

- Read down— A subject's clearance must be higher than the security level of the object being read.
- Write up— A subject's clearance must be lower than the security level of the object being written.

Hence, information only flows upwards or within the same security level in such a system. MAC can be applied to protect information integrity as well. For instance, the integrity levels are classified as Crucial (C), Important (I), and Unknown (U). A subject's integrity level represents the degree of trust of information stored in the object as well as the potential damage due to unauthorised modification. A subject's integrity level reflects the subject's trustworthiness of operation such as modifying, inserting or deleting data. Rules similar to those for preserving secrecy are shown as follows:

- Read up— A subject's integrity level must be lower than the integrity level of the object being read.
- Write down— A subject's integrity level must be higher than the integrity level of the object being written.

These principles protect integrity by preventing information in lower level objects to flow to higher level objects. As well as controlling information flow, additional mechanisms are also required to achieve integrity [36].

DAC is usually discussed in contrast to MAC (sometimes termed non-discretionary access control), which is an approach restricting or granting access to objects based policies determined by the objects' owner group and/or subjects. The specific authorisation is checked for each request of a user to access an object. If an authorisation exists stating that the user can access the object in the specific mode, the access will be granted, otherwise, it will be denied. Discretionary policies are flexible for a variety of systems and applications. Hence, they are widely used for various implementations, especially for commercial and industrial applications. DAC policies do not provide real assurance on the flow of information in a system. Bypassing the access restrictions stated through the authorisation is easy. For instance, when the permission to read data is granted to a user, the user might pass it to other users who are not authorised to read it without the knowledge of the owner. Information dissemination is not controlled. On the contrary, in mandatory systems, dissemination of information is controlled by preserving information stored in high-level objects to flow into low-level objects.

However, there are many practical requirements which can not be covered by classical mandatory and discretionary policies. RBAC was introduced, but it was considered as a more general model in comparison with these traditional models. RBAC assigns subjects to roles,

and each role is defined by policies on accessing rights to objects. The user assigned a role is permitted to execute all accesses for that which the role is authorised, where the same role can be taken by more than one subject. In addition, a subject could take on different roles on different occasions. For some RBAC proposals, the user is limited to obtain one role at a time. While other proposals allow a user to execute more than one role at the same time. Various approaches will be adopted in different applications, as there is no fixed standard in this arena.

Sandhu *et al.* [12] listed a few advantages of role-based approach which supports Authorisation Management, Hierarchical Roles, Separation of Duties and Object Classes. Firstly, it breaks authorisation task into two parts by introducing a logical independence in specifying user authorisations. Specifically, one designates roles to users and the other one designates access rights for objects to roles. It dramatically simplifies authorisation management. For example, when the responsibilities of an object changes, the previous role will be replaced by a new one. This avoids withdrawing access rights to corresponding objects if all authorisation is assigned directly to users and objects. Secondly, it supports hierarchical roles, where sub-roles will inherit privileges assigned to the general class role. For instance, teaching staff consists of lecturers and professors: a user who is assigned a role of lecturer will inherit access permissions for the general role of the teaching staff. In this way, RBAC further enhances authorisation management. Thirdly, RBAC supports separation of duties, in that it refers to the fact that no user, on their own, should be assigned privileges which could compromise. For instance, the person authorising the review of a paper should not be the one who submitted it. Separation of duties could be executed either dynamically or statically. Last but not least, the access authorities on each object could determine access automatically based on the classification of the objects. This benefits the authorisation management by making its role-easier.

Zhou *et al.* [37] proposed a new access control system for cloud environments, which integrated attribute-based access control with hierarchical role-based access control. The presented access control system automatically assigns users to roles based on policies applied to the attributes of users and roles. Hence, it relieves the data owner from the online and computational burdens of user-role assignment processes, especially, for large-scale systems with a huge number of users and continuously changing user role policies. In addition, hierarchical roles are applied in order to solve key management problems in a decentralised environment. Attribute-based policies support a fine-grained and flexible access control.

These basic access control models establish a solid theoretical basis for access control. Many existing policy models are developed based on basic access control models.

2.3.2 Provenance Access Control

Privacy and security of provenance are perceived as the main bottleneck to broad applications of provenance[38][39][40]. Privacy helps individuals maintain their autonomy and individuality, and security is the protection from theft and damage to provenance, as well as from disruption or misdirection of provenance. Hence, lacking protection for privacy and security can not convince users to trust the application of provenance. The research domain is still at its initial stage. Provenance security involves ensuring the integrity and availability of provenance records and connectivity. One of the current research challenges of provenance access control is lack of access control languages which fully support the specific requirements including defining provenance graph partitions. Several previous papers have provided provenance access control languages.

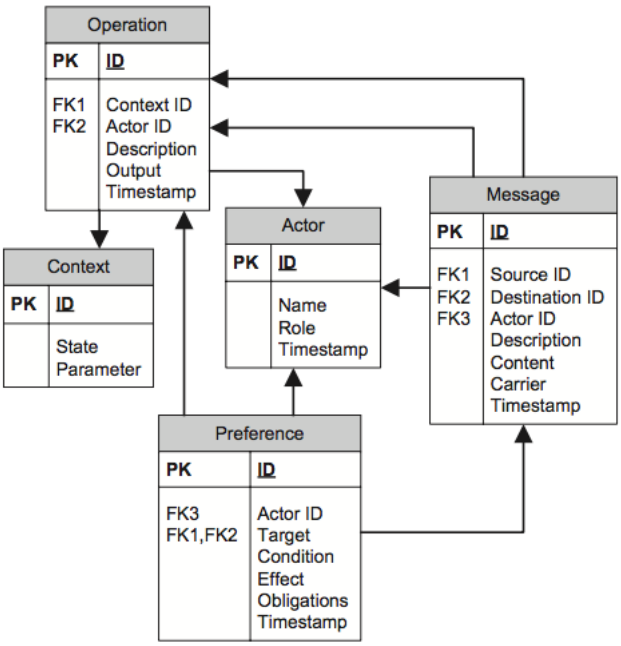

Figure 2.2: Provenance Record Schemata [2]

Ni *et al.* [2] presented an access control language tailored to generate policies including users' self-defined policies. The language is designed under the new provenance model

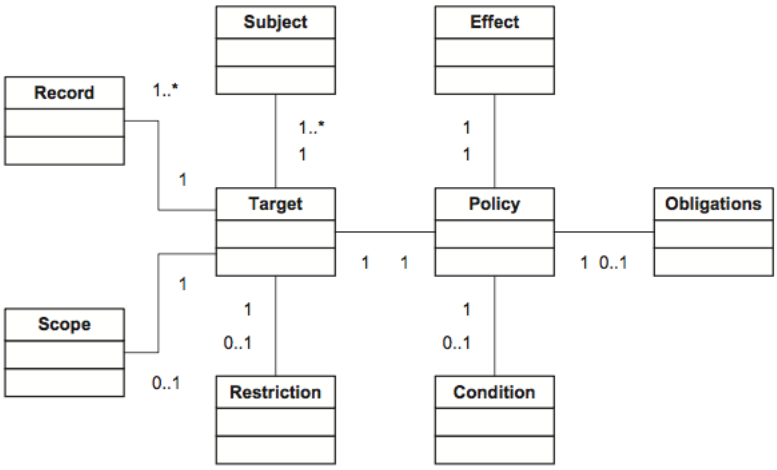

Figure 2.3: Access Control Language [2]

defined in the same paper. Under the basic assumption of the provenance model, a piece of data caused or manipulated by an operation results in output messages. A maximum of one operation manipulates a piece of data for one record. The model is presented as a set of provenance records. There are five types of records for each process. These are operation records, message records, actor records, preference records, and context records respectively. Each record is logged as several attributes. Figure 2.3 shows the elements of provenance records, where PK represents “primary key" and FK represents “foreign key". In this provenance, every record is uniquely identified by an identification attribute. For each record, some attributes are optional. Hence, their values might be null.

As a documented history, provenance is immutable [41]. However, in this provenance model, preference records are allowed to be rationally changed by their actors. Preferences are updated via two approaches: overwriting previous attributes, and associating each preference record with a time-stamp. When different preferences result in conflicted evaluation outcomes, the latest preferences take precedence.

The five records shown in Figure 2.2 make up a provenance graph and there are relations between the records. For one graph, attributes in one record reference attribute in other records. An operation, message, and preference records precisely reference the primary key of actor record; operation record references the PK of context; and so forth.

Their proposed access control language is based on the above provenance model which supports policies from organisations and actors. Each organisational policy consists of *target, condition, effect*, and *obligations* items. This is illustrated in Figure 2.3. Actor preference policies include one more time-stamp item, where the latest preferences take precedence.

The *target* enumerates subjects and objects that a policy is intending to apply. It includes an optional element *restriction* to refine the range of an applicable target as well as a *scope* element with two possible values “transferable" and “non-transferable", which defines whether the records could be extended to their ancestors or not. The *condition* item screens the context requirements for matching access requests, while *effect* states the results of the policy with four potential values. Lastly, *obligation* points out operations that have to be executed before the evaluation or after the access, and *timestamp* is an item in actor's preference policies. In the end, for one request, combining the consequences of all applicable policies outputs the final result.

This access control language is tailored for the provenance model presented in the same paper. It satisfies some specific requirements for provenance access control including supporting fine-grained access control policy for a specific data structure of provenance, aggregating preferences from data actors. However, as this work is still at the initial stage of provenance access control, it leaves several open issues.

Firstly, evaluation of target or condition might raise uncertainties, as predictions might not be evaluated due to lack of privileges. For instance, the restrictions are on the status of systems, but this information can not always be certain to be available for the party evaluating policies. Hence, the corresponding mechanism is expected to be introduced. Secondly, because policies are generated by multi-parties, detecting redundant, conflicting, and repeated policies is a challenge. Moreover, the policies lack the delegation mechanism which is essential for flexible and effective regulation of access control.

Cadenhead *et al.* [42] complements and extends existing access control language to protect both traditional data items and data provenance. Compare with the language proposed by Ni *et al.*, their language is tailored for an RDF data representation. They define the grammar of each of the tags in the language, which allows evaluation for the correctness of policies. Moreover, it allows unambiguous translation of policies into a form that can be used by the proper layer in the proposed system architecture.

In addition, the provenance graph structure not only poses difficulties to access control policies but also to querying language. It can use the set of names in a graph V_G to answer

common queries about provenance such as why-provenance, where-provenance, and how-provenance [43]. They create templates which are parameterised for a specific type of user query to anticipate the varying number of queries a user could ask.

A Denials-take-precedence conflict resolution is proposed in this paper as well. Under three scenarios which are $G_1 \subseteq G_2$ $G_1 \supseteq G_2$ $G_1 \cap G_2$ of graphs overlapping, the conflicts at the same area take deny as final effect. Namely, G_1 is permits access, but it consists of a subgraph G_2 which denies access. Only the diagram $G_1 - G_2$ can be accessed via combining the two results over Denial-take-precedence.

Danger *et al.* [44] proposed TACLP, an access control language to answer queries with transformed graphs which abstract over the missing nodes and fragments. Hence, they introduced the *Transformation* element into the access control language [2]. This allows administrators to specify provenance subgraphs to return users if the whole graph is not accessible. The transformation element consists of the specification of which nodes need to be hidden and which transformation operations should be applied. The two transformation types are *single* and *subgraph*, and the three transformation levels are *hide* (Rem_R), *minimum abstraction* (Rep_R but no soft dependencies will appear in the transformed graphs), and *maximum abstraction* (Rep_R soft dependencies could appear in the transformed graphs).

They presented algorithms to transform the provenance graphs. The general idea behind the algorithms is to collect nodes to be transformed, while the policies are being evaluated. The access control preference is defined as “deny takes precedence" or “permit takes precedence". The order in which to evaluate policies with the four effects is different for the two types of precedence, as the four effects which are “Absolute Permitted", “Deny", “Necessary Permit", and “Permit" are originally defined with different priorities.

While existing work focuses on graph transformation mechanisms, Liang *et al.* [45] propose the access control language for provenance that mainly deals with conflict solutions. In particular, the validity constraints of provenance to specify that certain parts of provenance are restricted to access has been taken into account. Provenance owners can define a set of objects from a nested partition of a provenance graph, and specify policies to access based on roles of subjects.

Moreover, this paper also explored how to extend the proposed approaches for a complete

provenance graph. In this paper, a simple provenance graph can be viewed as an inner layer of the complete one. Particularly, three forms of an object are defined as: an edge $e = (p, ag, wcb)$, a subgraph of its inner simple graph, and a kind of combination of three main entities and relationships. Data owners are able to define any forms of objects to support fine-grained access control. An algorithm Obj is proposed to calculate the smallest object for all the three forms, where users can partition the complete graph into fine-grained objects. Moreover, different roles can also be assigned to objects. This supports role-centric policies for answering access queries.

Kuwabara *et al.* [46] proposed a mechanism to facilitate access control and version management of an RDF database, which embodies RDF statements by supplementing meta-data and employs version management and statement-level access control. To be more specific, the meta-data was introduced under two situations, when a new triple was manually added by a user or derived from an existing ontology triple, which must be engaged under the specification of access permission and the existence of the version. In terms of the version formed under a tree structure, even though the parent version, which is specified as a new one is imported, they are still valid in their child version. Moreover, the meta-data associated with a reified triple is utilised for access control by specifying its property. In addition, the querying rewriting is mentioned to check the access condition.

In a nutshell, there are still issues not addressed by the existing provenance access control policies. Firstly, existing provenance access controls have not been established on a provenance graph model which has attributes of the main entities. This indicates that vertices in a provenance graph have not been confined by relevant attributes, such as timestamps, locations, roles *etc.* Moreover, more fine-grained access control for provenance has not been proposed in regards to provenance graph partitions, partition transformation labels. Lastly, proper algorithms for merging individual policies have not been provided.

Therefore, based on the existing work and open problems, we develop our contributions for provenance access control in this thesis. We establish a framework based on our extended provenance model OPM$^+$, which enables confining vertices in a provenance graph by relevant attributes. In addition, our proposed provenance access control language is fine-grained:

- to define allowed or prohibited provenance partitions for accessing;

- to hide sensitive partitions, thereby creating a new provenance graph.

Moreover, functions and algorithms to perform policies and transform provenance graphs based on policy results are presented in the thesis as well.

2.3.3 Provenance-based Access Control

Different from how Provenance Access Control determines access to provenance, Provenance-based Access Control controls access to data based on its provenance, because provenance contains clues to reveal sensitivities and vulnerabilities of data. This section explores several papers which attempt to use data provenance to make access control decisions[47][48].

Park *et al.* [49] proposes a basic provenance-based access control model $PBAC_B$, which facilitates additional capabilities beyond those available in traditional access control models. This paper also mentions a family of PBAC models, $PBAC_B$ being the basic model in this family. It defines three criteria for the provenance-based access control family, namely:

- the kind of provenance data in the system;
- whether policies are based on acting user dependencies and object dependencies; and
- whether the policies are readily available or need to be retrieved.

The three other models $PBAC_U$, $PBAC_A$ and $PBAC_{PR}$ extend one of these three criteria respectively. However, in this paper, the mechanism to list dependency paths manually and search dependency paths from provenance graphs is not unrealistic, because, in real cloud systems, the items in dependency path lists might be huge numbers.

Ni *et al.* [2] propose an access control language influenced by the XACML language which supports both actor preferences and organisational access control policies. In their system, applicable organisational policies and matching preferences are evaluated together for a given query. However, there are obvious shortcomings for the evaluation, such as the fact that uncertain decisions are inevitable due to the lack of privileges.

Lacroix and Boucelma [50] introduced a provenance-based access control system for the cloud. This integrates a rule-based mechanism and provenance into access-control models.

Particularly, indirect dependencies or locally-cloud context specific ones, are presented as standard expressions on dependency paths. Moreover, this system involves rule propagation and a mediator which is a central execution engine, in order to govern verification and authorisation of the execution process.

Based on PBAC, Nguyen, Park, and Sandhu [51] proposed several variations of a centralised provenance and PBAC-enabled authorisation services architecture, as well as identifying the potential to adopt PBAC into the cloud Infrastructure-as-a-Service. The two approaches to communication between the service components are intra-service communication and inter-service communication. In addition, by demonstrating, evaluating and analysing the architectural implementation in the context of the OpenStack cloud management platform, the authors proved that the design constitutes a strong foundation for enhanced authorisation in cloud platforms.

Decat *et al.* [52], a concurrency control scheme specifically for the evaluation of access control policies was proposed. This work is motivated by the need to enforce access control policies on distributed applications, in which the policies have to be evaluated concurrently and distributedly as well. However, for certain classes of policies such as history-based policies, concurrency can be exploited to gain elevated access. By leveraging the specific structure of a policy evaluation, the concurrency control scheme effectively prevents such incorrect access decisions and is able to scale to a large number of machines while incurring only limited and asymptotically bounded latency overhead. This result could not have been achieved by employing more general techniques for concurrency control. This paper has taken an important step towards applying policy-based access control to realistic applications in practice.

Yan and Fang [53] proposed a constrained policy representation for facilitating IRM optimisation. This policy representation is expressive enough to represent simple integrity policies, Generalised Chinese Wall Policies, and Hierarchical One-Out-Of-k Policies. The core optimisation procedure is safe, unobtrusive and effective. The optimisation procedure has been extended to accommodate a distributed optimisation protocol, in which an untrusted code producer may formulate method interfaces to boost the optimisation effectiveness of a distrusting code consumer. A prototype of the procedure has been implemented and demonstrated to exhibit positive performance characteristics. The authors are exploring alternative optimisation directives that could lead to more effective optimisation than their

current design of method interfaces.

In order to facilitate access control in provenance-aware systems, Sun *et al.* [3] established an abstract provenance model which is named Typed Provenance Model (TPM). Hence, TPM accommodates generic attributed-based policies by employing *provenance-type* as a special attribute. In addition, different levels of abstraction were proposed aiming to accommodate provenance-aware policies flexibly. In detail, TPM defines two provenance types including element types and dependency types, where each type identifies a collection of elements such as nodes and edges in provenance graphs. Dependency type is mentioned as the core concept of this model, which is defined as T:=N(E, C), where N is a composition of the name of a dependency type T, E is an element type as an effect, C is an element type as a cause.

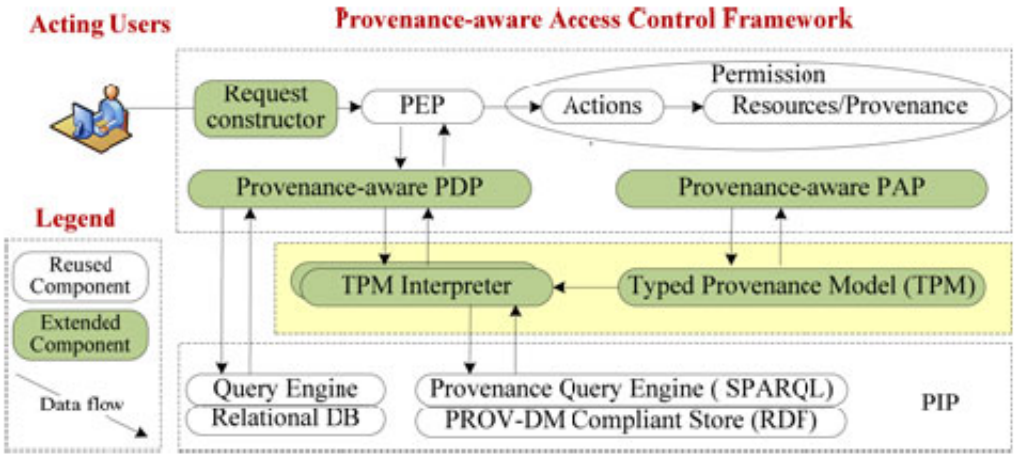

Figure 2.4: A Layered Architecture of Provenance-aware Access Control Framework [3]

A layered architecture of provenance-aware access control framework is also proposed in paper [3]. In the architecture, reused or extended modules from either the existing XACML architecture or provenance-aware systems (PAS) are on the top layer and bottom layer, while a set of TPM interpreters serving specification and enforcement of policies is displayed in the middle layer. TPM interpreter links Provenance-aware PDP and Provenance Query Engine (SPARQL) is able to correctly interpret dependency types. However, when several provenance stores in various physical representations are used in a PAS, it demands multiple provenance query engines and TPM interpreters.

We identify some issues from the existing provenance-based access control work. Firstly, different types of attributes have not been distinguished properly in policies, including vertices types, vertices names, and vertices attributes. In a fine-grained provenance-based access

control policy, meeting different types of attributes should result in correspondingly different results. Secondly, operators to merge results in a multi-valued decision set should be generated. In this thesis, mechanisms to address these identified issues are presented.

2.3.4 Policy Algebra

As more than one policy or rule might affect a request, functions in terms of how to merge results of individual policies as a final decision should be taken into account to construct a complete framework of access control. Hence, some access control policy algebra frameworks are reviewed in this section.

Bonatti *et al.* proposed an algebra [54] of policies with their formal semantics and demonstrate how to how to formulate complex policies using algebra. Firstly, they identified the problem of integrating policies in a modular and incremental fashion. It is proven that the composition algebra with restricted closure operators is less powerful compared with relational algebra because it is not able to manipulate multiple relation schemata. However, certain significant decision issues which are otherwise undecidable are decidable for the composition algebras. Then, a translation of policy expressions into equivalent logic programs is the basis for the implementation of the algebra, which is illustrated in this paper. In addition, the expressions of this algebra are analysed via a comparison with first-order logic. This work addresses the issue of combining authorisation specifications possibly in different languages.

In 2003, Wijesekera and Jajodia presented [55] a propositional algebra for modeling security policy operators such as union, sequential composition, intersection, adding provisions and so on. After recognising the issue that authorisation policies can assign sets of permissions to subjects, two operations are identified in this paper. Specifically, the external type only quantifies on permission sets without reorganising them, while the other internal type alters these sets. Moreover, the policy models can be used to identify whether two policy operations or integrations are equivalent, and then a few algebraic identities are constructed to verify equivalent policies. The other policy model is generated to identify whether compositely policies are complete, consistent, or deterministic. This work improves the outcome in several ways. Firstly, a predicate version is presented, in which properties are generated from relation, function, variable, and constant symbols, but not left as abstract symbols. The area of the basic automata model is another improvement.

In distributed systems, access control policies of collaborating parties are often required to be integrated. Bonita *et al.*[54] firstly introduced the concept of policy composition. In this paper, logic programming and partial evaluation techniques for evaluating algebra expressions are employed to compose access control policies. Following Bonita *et al.*[54], Rao *et al.*[56] provides a framework which uses algebra for fine-grained integration of sophisticated access control policies. The algebra consists of binary and unary operations as well as derived operations, which are able to address the constraints of integrations. Particularly, the algebra combines three-valued access policies. These are Permit policy, Deny policy and Not-applicable polices. The integration of policies might consist of multiple operators, hence this paper defines the concept of FIA expressions. Then, it proves the completeness of this framework that can handle many policy integration scenarios and present approaches to automatically generate integrated policies given FIA policy expression and to transform them into actual XACML policies.

Ni *et al.* [57] presented a D-algebra which is functionally complete and computationally effective to integrate decisions from multiple policies. The D-algebra consists of an analysis of policy languages decision mechanism to contribute relevant applications in the context of access control policies. They also implemented the decision specification language and developed a toolkit to evaluate policy expressions. In addition, this paper investigated issues concerning decision composition for hierarchies.

Hanson *et al.* [58] provided a data-purpose algebra used to model allowable usage of data. The system of Records is associated with System of Records Notices (SORN) which maps attributes of content to permissible usage scope. The data repository specifies input conditions and a set of routine uses *U (n)* for those matching the conditions. Similarly, the source agent also sets SORN which state the policies on the permissible usages of outgoing data. The final scope of usage is the intersection of the results of both sides of SORN and the purpose of attaching to the data content. However, this work is not good enough to support more complicated policies involving historical records and the relationship between data.

The existing techniques of policy algebras cannot be utilised for our proposed access control policies directly. Hence, based on the existing policy algebras, we tailored policy algebras for our proposed access control policies. The algebras merge results of individual policies to generate a final result for an access request.

2.4 Cryptographic Techniques for Implementing Access Control Policies

In this section, we explore cryptographic techniques which could be borrowed to implement access control policies on provenance. To protect privacy and confidentiality of provenance itself, several previous works have been proposed to encrypt provenance. Moreover, provenance should be checked or verified by a third party to meet provenance-based access control. In the meantime, provenance is encrypted to avoid leaking information to a third party. Then, keyword search techniques for ciphertext could be employed for provenance-based access control. In addition, as a method to facilitate provenance-based access control, cryptographic schemes can effectively preserve the confidentiality of data via encryption and grant decryption keys to eligible users. Therefore, in this section, we present relevant literature on these techniques.

2.4.1 Confidentiality of Provenance

Data provenance might be sensitive information, in which case the security of provenance (for example [41], [59] and [60]) has increasingly aroused attention. There are have been several attempts to encrypt provenance information to keep its confidentiality. Li *et al.*[61] proposes a provenance-aware system based on Attribute-based signature (ABS) which supports fine-grained access control policies. The users' privacy is also protected because the attribute private key of users is issued with an anonymous key-issuing protocol from multiple attribute authorities. However, the whole computation is built on the assumption that the could server has a large computational ability. Chow *et al.*[62] proposed a cryptographic design for cloud storage systems supporting dynamic users and provenance data.

In the area of encrypted data search, Boneh *et al.* [63] present a Public Key Encryption with Keyword Search (PKES) scheme. We will be making use of this work in the design of our Provenance-based Classification Access scheme. Essentially, the work proposed by Boneh *et al.*[63] considers the following scenario: when Alice receives emails, she would like to set a gate that helps her to check whether the incoming emails contain certain sensitive keywords such as "urgent". However, the emails are encrypted to protect privacy. As the gateway is not fully trusted, Alice does not want to grant the gateway the ability to decrypt

her emails. The PKES scheme enables the gateway to conduct a test to verify if the encrypted emails contain the keywords while learning nothing else about the content of the emails themselves.

Bates *et al.* [4] introduced a framework to manage provenance metadata in the cloud. In addition, in order to enforce organisational security policies, it introduced an approach for using provenance as the basis for an attribute-based access control system. Although it employs provenance to enforce access control, the mechanism effectively transfers the access control paradigm from a stateless decision made strictly on the current state of data to a stateful evaluation according to data's origin and lineage. In the protocol which is illustrated in Figure 2.2, the Cloud Provenance Authority (PA) generates a signature by the random number n_c sent from the Clients (C) which will be sent with a provenance chain and a random number, then when the client updates the provenance and object and sign them with random number n_{pa} This is the first protocol been aware of to be focused on securing and managing provenance in distributed cloud environments. Further, the implementation using the Cumulus cloud storage environment on the University of Oregon's ACISS science cloud illustrates that this is a practical system to support thousands of operation per second on a modestly specified deployment.

1. $C \rightarrow PA: \quad n_c, Guid_c, oid$
2. $PA \rightarrow C: \quad n_{pa}, PC_{oid,t-1}, Sign[K_{pa}^{-}, PC_{oid,t-1}||n_c]$
3. $C \rightarrow PA: \quad Sign[K_c^{-}, PC_{oid,t}||n_{pa}], PC_{oid,t}, Object$
4. $PA \rightarrow C: \quad Sign[K_{pa}^{-}, PC_{oid,t}]$

Figure 2.5: Commitment Protocol for Writing Operations[4]

2.4.2 Public-key Encryption with Keyword-Search Schemes (PEKS)

To implement provenance-based access control, a server could be viewed as a third party which can not be fully trusted. Hence, provenance is usually encrypted to preserve confidentiality. Keyword Search schemes for ciphertext propose solutions to facilitate provenance-based access control, which enables the ciphertext of provenance to be verified without decryption.

Assume Bob sends emails to Alice encrypted under Alice's public keys [64]. An email

gateway would like to verify whether the mail contains the keyword “urgent". Thus, Alice sends a short secret key T_w to the mail server. It enables the server to locate all the messages containing the keyword “urgent", but learn nothing else. The secret key T_w was generated by Alice under her private key.

To be more specific, PEKS schemes propose a mechanism which enables a keyword search in the ciphertext, in order to preserve the privacy and confidentiality of data against a third party. Usually, the encipherers owning the private keys generate the “trapdoors" of data for the keywords checking purpose. The schemes provide an approach which allows the third party to check keywords without decrypting the ciphertext.

PEKS has been applied to various applications, including retrieving sensitive ciphertext in cloud, searchable and encrypted audit logs[65], intelligent email routing[66] [64] and *etc.* Fang *et al.* [5] presented an example application for keyword search schemes in the figure below, where Bob sends messages attaching keywords $\tilde{C}$ to Alice, where $\tilde{C} = (C_{PKE}||C_{PEKS})$ = (PKE(pk_A, m) || PEKS (pk_A, w)). The Email server verifies if the ciphertext matches trapdoor T'_w generated by Alice, namely w$\stackrel{?}{=} w'$.

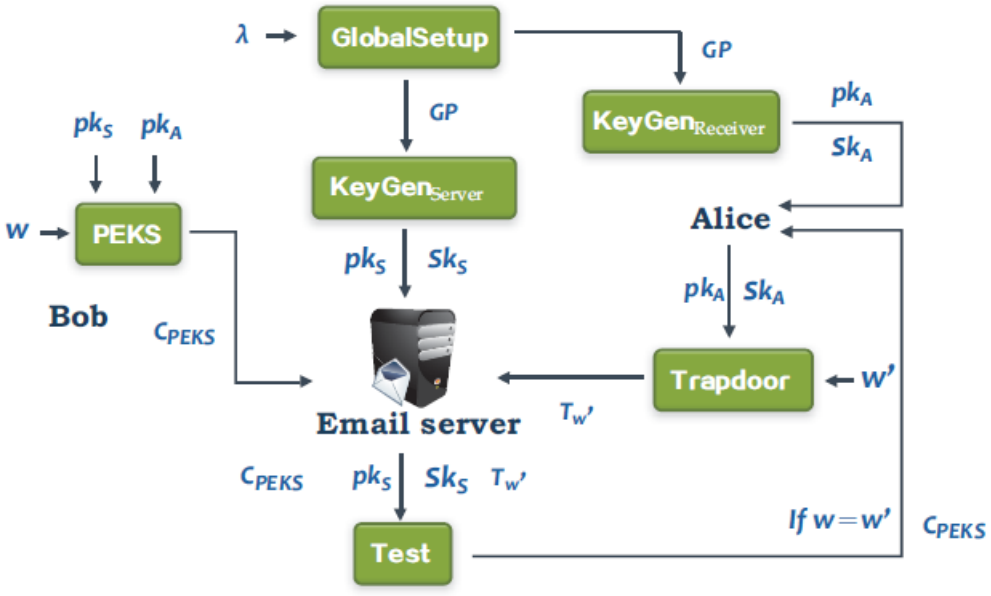

Figure 2.6: Keyword Search Scheme Application in Intelligent Email Routing[5]

1) *Traditional PEKS:* Bones *et al.* [7] proposed a original mechanism named *Public Key Encryption with Keyword Search* which is derived from Identity-based Encryption (IBE). It is utilised in the following scenario. When Bob sends emails that are encrypted under Alice’s

public key to Alice. An email gateway wants to verify if there are nominated keywords in the emails such as “urgent", in order to process them accordingly. Hence, Alice sends a key of “urgent" to the gateway. The gateway can test whether the keywords are in the encrypted emails but learn nothing else.

Abdalla *et al.* [66] identified and filled in some gaps in terms of consistency which indicates that decryption reverses encryption. By defining computational and statistical consistency, they prove the computational consistency of a scheme [7] and provide a new scheme which meets statistical consistency. Three extensions of the basic notions are suggested these are anonymous HIBE, public-key encryption with temporary keyword search, and identity-based encryption with a keyword search.

Waters *et al.*[65] designed a scheme to protect the contents of audit logs as well as making them searchable. They used identity-based encryption to protect symmetric keys which encrypt audit log entries. A third party utilises capabilities created by a privileged agent to search the audit log of records if it matches certain keywords. The scheme has been implemented for MySQL database queries. It turns out the scheme to improve security and convenience over symmetric key based schemes.

Khader [67] proposed the K-Resilient Public Key Encryption with Keyword Search (KR-PEKS) which is secure under a chosen keyword attack without the random oracle. The construction of the KR-PEKS was extended from constructing a Public Key Encryption with Keyword Search from Identity-Based Encryption. Two modifications were created to support multiple keyword searches and remove the need for secure channels. However, the limitation of it is that the number of malicious users is restricted to K. Hence, the value of K should be set as a large number.

Finding a PEKS that is not based on bilinear forms has been an open problem. Crescendo *et al.* [68] proposed a scheme using a non-trivial transformation of Cocks' identity-based encryption scheme[69]. Hence, the scheme is based on a new assumption which is a variant of the difficulty of deciding quadratic residues modulo for a large composite integer. As PEKS can be viewed alongside anonymous identity-based encryption. However, it is less efficient than the original Cocks' scheme.

2) *Secure Channel Free PEKS:* Beak *et al.*[9] removes secure channel for PEKS schemes by proposing a new PEKS scheme SCF-PEKS. The basic idea to construct the scheme is to

introduce the public key and private key pair of a server, where the trapdoor and ciphertext are generated with the receiver's public key and server's public key. Therefore, trapdoors could be sent without the secure channel because they are protected by the server's public key. Then the server verifies the match of ciphertext and trapdoor via its private key. Overall, another key pair from the server is introduced into the scheme to preserve trapdoors resulting in the trapdoors being transferred without a secure channel. In addition, an idea of attaching an effective range of time for each keyword is presented in this paper as well.

Rhee *et al.* [70] upgraded Beak *et al.*'s scheme to meet a stronger security model. Specifically, under the enhanced security model, the public key can be published by an attacker without revealing the corresponding private key. Moreover, the relation between non-target ciphertext and a trapdoor is also allowed to be obtained. The scheme is proven as semantic secure under their defined new security model against chosen keyword attacks, referring to 1-BDHI and BDH assumptions.

Emura *et al.* [71] extends SCF-PEKS as an adaptive SCF-PEKS, which is secure even when an adversary is able to issue test queries adaptively. Moreover, lightweight adaptive SCF-PEKS is proposed. This is established by anonymous identity-based encryption only without extra cryptographic primitive. In addition, a more efficient but not fully generic construction is presented, where the server is utilised as a test oracle by malicious receivers.

3) *Against Outside KGA:* Byun *et al.*[72] defined *Single Keyword Search Scheme* and *Conjunctive Keyword Search Scheme*. Particularly, a conjunctive keyword search scheme enables a user to verify a combination of several keywords together, where trapdoors are generated for conjunctive keywords. In addition, this paper also identifies off-line keyword guessing attack for PEKS. Byun *et al.* analysed security vulnerability of Boneh *et al.*'s scheme and Park *et al.*'s scheme under keyword guessing attacks.

Following Byun *et al.*'s paper, Yau *et al.* [6] demonstrated off-line keyword guessing attacks on PEKS and SCF-PEKS. By presenting the vulnerabilities of these schemes under off-line keyword guessing attacks, Yau *et al.* concluded that SCF-PEKS is not secure without a secure channel. Moreover, it would still be possible for insider adversaries to break the schemes with a secure channel. The figure below compares four PEKS schemes for their performance against off-line keyword guessing attacks.

Beak *et al.* [10] proposed “IND-PKE/PEKS-CCA" by integrating a PKE and PEKS,

Consequences	Type of Adversary	PEKS Schemes							
		Without Secure Channel				With Secure Channel			
		A	B	C	D	A	B	C	D
Reveal a key-word sent in Trapdoor	Insider	X	X	X	X	X	X	X	X
	Outsider	X	X	X	X				
Determine a PEKS cipher-text is the result of encrypting which keyword via Test phase	Insider	X	X	X	X	X	X	X	X
	Outsider	X	X		X				

Figure 2.7: Summary of Consequences of Off-line Keyword Guessing Attacks [6] (A=BDOP-PEKS [7], B=PECK [8], C=SCF-PEKS[9], D=PKE/PEKS [10])

which is secure against chosen ciphertext attack. The scheme is based on randomness reuse techniques referring to the variation of ElGamal encryption schemes. Hence, the security of IND-PKE/PEKS-CCA in the random oracle model is based on the computational difficulty of Computational Diffie-Hellman (CDH) problem. Moreover, another generic construction of PKE/PEKS scheme is presented, which is slightly less efficient compared with the other one in this paper. The proposed PKE/PEKS are also extended to multi-receiver settings and multi-keyword settings.

Rhee *et al.* [73] answers the open problem proposed by Byun *et al.* [72] by proposing a secure searchable public-key encryption scheme with a designated tester (dPEKS) scheme against keyword guessing attacks. They defined consistency in dPEKS and analysed how keyword guessing attacks can break PEKS/dPEKS schemes. As a discrete the logarithm problem is difficult, the scheme proposed in the paper is computationally consistent and secure.

Rhee *et al.* [74] enhanced the security of dPEKS against off-line guessing attacks. Firstly, they upgraded the existing security model of dPEKS from the paper [10], where an attacker does not have to reveal the private key during the setup phase. It grants more power to attackers and requires higher security of schemes to meet the model. Moreover, they defined

“trapdoor indistinguishability" of dPEKS, where an adversary is trying to guess trapdoors for a keyword of his choice. The indistinguishability of trapdoors ensures that a trapdoor does not leak information of keywords. Moreover, the authors explored the connection between security against off-line keyword guessing attacks and trapdoor indistinguishability and further proposed a new scheme to satisfy these models.

A formal secure model of SCF-PEKS against off-line keyword guessing attacks was proposed by Fang *et al.* [5]. Then, constructed from DBDH assumption, SXDH assumption and the truncated q-ABDHE assumption, an SCF-PEKS scheme without random oracle against keyword guessing attacks and chosen keyword and ciphertext attacks is proposed. However, the computational efficiency of the scheme is expected to improve.

4) *Against Inside KGA:* Nevertheless, to explore the open problem published by Byun *et al.* [72], Jeong *et al.*[75] provided a negative result in that consistency of PEKS schemes leads to insecurity under keyword guessing attacks. Namely, they concluded that establishing a PEKS scheme to both meet consistencies and secure against keyword guessing attacks is impossible. However, the conclusion is proven under the original framework, which implies that the security issues might be addressed by adjusting the framework.

There are plenty of schemes that are proposed in regards to keyword search techniques for ciphertext. In this thesis, we upgrade an existing scheme to implement a file classification policy on provenance, while checking the authentication of the data users who encrypt provenance.

2.4.3 Attribute-based Encryption for Access Control

Besides searching keywords from the ciphertext of provenance, encrypting data is an approach to control access. Then, decryption keys could be associated with eligible users. Several previous works have already accommodated provenance-aware policies by treating the *typed provenance partition* as a special attribute. Namely, by accommodating provenance-aware policies, attribute-based access control encryptions can be employed to implement provenance-based access control policies.

Introduced By Sahai and Water [76], attribute-based encryption (ABE) is a more efficient public-key encryption scheme which implements integrated and complicated access structures. In ABE schemes a message is encrypted under a set of attributes. Receivers must obtain the attributes used as secret keys from a trusted party called central authority (CA), then they are able to decrypt the ciphertext and obtain the data if and only if there is a match between his secret keys and the attributes listed in the ciphertext. The original idea of ABE is presented as a fuzzy (error-tolerant) identity-based encryption (IBE) scheme [77, 78]. Since it was published as an efficient approach, ABE has attracted lots of attention in the public-key cryptography research community. Generally, ABE schemes can be classified into two types:

Ciphertext-Policy ABE (CP-ABE): In these schemes [79–83], a user's secret key is labeled with a set of attributes, while a ciphertext is encrypted by an access structure.

Key-Policy ABE (KP-ABE): In these schemes [76, 84–87], a user's secret key is associated with an access structure, while a ciphertext is encrypted by a set of attributes.

In a distributed system, an *access structure* is constructed to define eligible combinations of attributes from the users to access the objects. Given a universal set $\mathbb{P}$, if a subset $S' \subseteq \mathbb{P}$ satisfies an access structure, then all subsets $S \subseteq \mathbb{P}$ which contain S' also satisfy the access structure, we say that the access structure is *monotonic*. A (k,n)-*threshold access structure* is a type of structure, where given a universal set $\mathbb{P}$ with $|\mathbb{P}| = n$, a subset $S \subseteq \mathbb{P}$ satisfies the access structure if and only if $|S| \geq k$. In an ABE scheme, an access structure is defined by the encryptor (in CP-ABE) or the authority (in KP-ABE) to decide who is able to decrypt a ciphertext. For instance, in a KP-ABE scheme, the authority specifies an access structure and issues secret keys to users. A message is encrypted with attributes from the access structure and attributes will be listed in the ciphertext. If a user holds a set of attributes which satisfy the access structure, they can use their secret keys to decipher the ciphertext and obtain the message. In the opposite way, if a user's secret keys cannot satisfy the access structure, it is impossible for them to decode the ciphertext.

However, the original ABE scheme contains a limitation in that it can only express a threshold access structure. Subsequently, an ABE scheme for fine-grained access policy was presented by Goyal, Pandey, Sahai and Waters [84], where the *access tree* technique could express any monotonic access structure. Specifically, there is a tree access structure with leaf nodes consist of the attributes and interior nodes represent AND and OR gates. Each interior node ω of the tree specifies a threshold gate (k_ω, n_ω), where n_ω is the number of the

children of ω and $0 < k_\omega \leq n_\omega$. Therefore, when $k_\omega = 1$, it is an OR gate; when $k_\omega = n_\omega$, it is an AND gate. Only when a set of attributes satisfies the tree access structure, the secret embedded in the vertex of the tree could be reconstructed by the corresponding secret keys. Following Ostrovsky, an ABE scheme with a *non-monotonic access structure* was proposed by Sahai and Waters [85] to express more complicated access structures, where a secret key is labeled with a set of attributes including not only the positive but also the negative attributes.

Subsequently, Bethencourt, Sahai and Waters [79] presented the first CP-ABE which was proven to be secure in the generic group model. In contrast with a KP-ABE scheme, a CP-ABE scheme's access structure is constructed by the encryptor, but not the CA. Hence, the encryptor can determine access structure; while, in a KP-ABE scheme, this is decided by the CA. In 2007, Cheung and Newport [80] proposed another CP-ABE scheme but reduced the difficulty of breaking their scheme based on the DBDH assumption. Unfortunately, these CP-ABE schemes can only express a threshold access structure, hence, Waters proposed a more generic CP-ABE scheme [83] which employs *linear secret sharing scheme* (LSSS) technique [88] to express any access structure.

A dual policy ABE scheme [89] that combines a KP-ABE scheme with a CP-ABE scheme was proposed by Attrapadung and Imai. Two access structures are employed in this scheme: one is for the subjective attributes held by the users and the other one is for the objective attributes labeled in the ciphertext. Nevertheless, there is only one access structure in both a CP-ABE scheme and a KP-ABE scheme. Rial and Preneel [90] present a blind key extract protocol for the centralised ABE scheme [79] which is a blind centralised ABE scheme.

In practical systems, such as data outsourcing systems [91], cloud computing [92], or distributed systems [93], ABE has been used as a building block to express flexible access structures. Against the *collusion attacks*[76], it is a basic benchmark for ABE schemes, namely a group of users cannot combine their secret keys for decryption when none of them can decrypt independently. The most common technique used to restrain collusion attacks is randomisation. CA could randomise a user's secret key via selecting a random polynomial [76] [86] [87] or a random number [85] [80].

2.4.4 Multiple-Authority Attribute-based Encryption

Even though Sahai and Waters' work [76] made an enormous contribution, an open question was left regarding whether it is possible to construct an ABE scheme where a user's secret key can be granted by multiple authorities. Chase [86] addressed this issue by presenting a multi-authority KP-ABE scheme. This ABE scheme supports multiple authorities, where CA is one among them. A user has to obtain secret keys from all these authorities, and CA comprehends all the secret keys of the other authorities. Being different from one-authority ABE schemes, it is hard to combat collusion attacks in a multi-authority ABE scheme. If the multiple authorities can work independently, the scheme is particularly vulnerable to this attack. Chase [86] overcame this problem by introducing a global identifier (GID) to a multi-authority ABE scheme. All authorities connect a user's secret keys to his GID. CA employs a user's secret key and the other authorities' secret keys to compute a special secret key for the user, in order to enable ciphertext to be independent of user's GID. Even though this scheme could *not* support a decentralised system, it still made a significant contribution to boosting ABE schemes from a single authority to multi-authority.

M*ü*ller, Katzenbeisser and Eckert [94] proposed a distributed CP-ABE scheme and proved it to be secure in the generic group [79]. In addition, the pairing operations executed by the decryption algorithm are constant, but the limitation of this scheme is that there must be a central authority to generate the global key and issue secret keys to users. Another multi-authority ABE scheme based on the distributed key generation (DKG) protocol [95] and the joint zero secret sharing (JZSS) protocol [96] was presented by Lin, Cao, Liang, and Shao [97]. This could be without a central authority. At setup phase, the multiple authorities have to collaboratively run the DKG protocol twice and the JZSS protocol k times which is the degree of the polynomial selected by each authority. In addition, every authority must obtain $k + 2$ secret keys. However, this is k-resilient scheme, namely the scheme is secure if and only if the number of the colluding users is no more than k which is determined at the system setup phase.

Chase *et al.* [87] another multi-authority KP-ABE scheme was presented. The contribution of this scheme is that it removed the necessity of CA and improved the previous scheme [86]. They also specifically addressed the privacy issue of previous ABE schemes [86, 97]. The issue is that a user must submit his GID to each authority to obtain the corresponding secret keys, which would expose the user to being traced by a group of corrupted authorities.

The solution of this paper is that it employed the 2-party secure computation technique and provided an anonymous key distribution protocol for the GID. Consequently, a group of authorities is not able to cooperate to obtain a user's attributes by tracing his GID. However, it requires the multiple authorities to interact to set up the system. Specifically, each pair of authorities has to run a 2-party key exchange protocol to share the seeds of the selected pseudorandom functions (PRF)[98]. This is a $(N-2)$-tolerant scheme, which means the scheme is secure if and only if the number of the compromised authorities is no more than $N-2$, where N is the number of the authorities in the system. The security of this scheme was proven by reducing it to DBDH assumption and non-standard complexity assumption (q-decisional Diffie-Hellman inverse (q-DDHI)). At the end of this paper, the authors also presented an open challenging research problem on how to construct a privacy-preserving multi-authority ABE scheme without the need of cooperations between the authorities.

A fully secure multi-authority CP-ABE scheme in the standard model was present by Liu, Cao, Huang, Wong and Yuen [99]. This is based on the CP-ABE scheme [82]. It was constructed in the composite order ($N = p_1p_2p_3$) bilinear group. Multiple central authorities and attribute authorities co-exist and cooperate in this scheme. Specifically, the attribute authorities issue attribute-related keys to users and the CA distributes identity-related keys. By running the algorithm, before obtaining attribute keys from the attribute authorities, a user must obtain secret keys from the multiple central authorities.

In 2011, the cluster multi-authority ABE schemes added a new member named decentralising CP-ABE [100]. Based on the previous multi-authority ABE schemes that require collaboration among multiple authorities to set up the system, this scheme makes a contribution that no cooperation between the multiple authorities is required in setup phase and key generation phase. The scheme was constructed in the composite order ($N = p_1p_2p_3$) bilinear group and achieves full (adaptive) security in the random oracle model. In addition, it is independent of central authority. The improvement of this scheme is that an authority in the algorithm can join or leave the system freely without the necessity to re-initialise the system. The authors also pointed out two approaches to creating a prime order group variant of their scheme. Although it made progress theoretically, the two drawbacks of this scheme are obviously: one is that it's not efficient [83]; the other one is that a user's attributes can be collected by tracing his GID.

Via exploiting anonymous key issuing protocol [87], Li *et al.* [101] proposed a multi-authority cipher-policy ABE scheme with accountability. The contribution of this scheme is that a user can *only* obtain secret keys anonymously from $N - 1$ authorities. In addition, in this system, a user can be traced and monitored when they share their secret keys with others. Unfortunately, the multiple authorities initialising the system interactively make the computing of the scheme complicated and even impractical. To prove the security of the scheme, the simulation of the scheme was reduced to DBDH assumption, decisional linear (DLIN) assumption and q-DDHI assumption.

In this thesis, we borrow ideas from attribute-based encryption schemes to generate a new scheme. The new scheme is able to quickly generate keys under an access control structure based on a similar structure with keys. This technique can be utilised in provenance-aware systems because data owners for different sections of a file might generate related but different access control policies for the sections they contribute to. Our proposed scheme can be employed in such a way to generate encryption keys under these access control structures, in order to preserve the confidentiality of provenance.

2.5 Conclusion

In conclusion, we survey literature surrounding the topic *Access Control on Provenance*, including access control policies and cryptographic schemes to execute access control.

However, we identify that certain issues on this topic have not been addressed. For provenance access control, the policy languages tailored based on OPM can not support fine-grained access control policies. Hence, we propose an extended provenance model (OPM$^+$) and more fine-grained access control policy language for provenance. Moreover, the approach to defining provenance partitions in access control policies was not proposed properly and in enough detail. Hence, in the thesis, we tailored a policy language to define various types of provenance partitions under our self-defined OPM$^+$. Additionally, we provide algorithms to make evaluations based on our proposed access control policies.

In terms of provenance-based access control policies, which make access decisions for data based on its provenance, the existing policy models did not match the complexity of data structure of provenance. Hence, we define a fine-grained provenance-based access control

model to properly extract elements from provenance. Our proposed policy model distinguishes different types of elements from provenance and maps to a “four-valued" decision set. Moreover, the corresponding policy algebras are presented in this thesis as well.

Being inspired by existing policies, we propose Purpose-based Privacy Policies on provenance which map provenance partitions to allowed/prohibited access purposes. However, to the best of our knowledge, existing policy algebras have not merged purpose sets to distinguish purposes at different levels of sensitivity. Therefore, we provide policy algebras to address this issue.

Finally, cryptographic schemes to implement access control policies on provenance are also significant approaches in the facilitation of access control issues on provenance. Hence, based on existing techniques, we updated cryptographic schemes to make them fit access control policies on provenance, which are presented in Chapter 6 and Chapter 7 respectively.

3

PACLP: A Partition-Based Access Control Policy Language for Provenance

Even though the idea of partitioning provenance graphs for access control was previously proposed, defining partial provenance DAG for fine-grained provenance access control has not been thoroughly explored. Hence, we define a more fine-grained provenance access control policy language based on our proposed the extended OPM, where provenance subgraphs are defined properly. Moreover, we provide algorithms to implement the proposed policies. including retrieving applicable policies for a query, merging individual policy results as a final determination and transforming provenance graphs.

3.1 Introduction

3.1.1 Related Work and Motivations

Data provenance logs historical operations performed on data. In some situations, provenance information might be more sensitive than the data. For instance, although a programming project can be published to the public, the series of operations to generate the project should be kept private, to avoid leaking the techniques of establishing such a project. Therefore, the access control of provenance is a popular research topic in recent years. It allows eligible users to access the provenance data and protects it from unauthorised access.

At first glimpse, since provenance itself is data, there should be a vast amount of existing access control techniques that are available for provenance access control. However, this is not the case. Most traditional access control techniques cannot be applied straightforwardly, because the specific data structure of provenance cannot be easily supported by most traditional access control techniques. Particularly, in this chapter, the proposed policy language is based on the extended OPM which is able to store attributes of entities.

Several existing work have already explored this topic. Cadenhead *et al.*[42] extended an existing provenance access control language proposed by Ni *et al.*[2], which introduced regular expression to protect traditional data items as well as their relationships from unauthorised users. It is an XML-based structure policy language and associates grammar based on provenance graphs. In order to evaluate the effectiveness of their policy language, a prototype based on their architecture utilising Semantic Web technologies has been implemented. Subsequently, Danger *et al.*[44] presented approaches allowing policies to define subgraphs to be transformed by three levels of abstractions. This work proposed algorithms to transform provenance graphs and generate an accessible version for queries.

There are issues have not been addressed by the existing work. Although Danger *et al.* proposed an idea to return provenance sub-graphs by answering access queries, their approach relies upon explicitly enumerated sets of nodes. Namely, provenance subgraphs that are only partially defined cannot be used in defining access in these proposals. In another word, the current provenance access control language models are unable to define provenance subgraphs accurately on a summarised level. On the other hand, to reduce policy redundancy

and encourage the re-usage of policies, a collection of nodes should be defined by a few key features instead of listing them completely.

Therefore, we propose a new framework for provenance access control including a provenance access control language and evaluation algorithms to address the issues nominated above. Firstly, we define several types of provenance segments over the Extend OPM and define language elements to express them by XPath. The approach enables to define a collection of nodes by nominating their features, such as the starting node and the ending node. Secondly, we introduce provenance subgraphs as a policy condition. For example, if the provenance graph contains a subgraph saying *the object was submitted and graded*, then it can be accessed.

3.1.2 Our Contribution

A provenance access control language is proposed in this chapter that can define a collection of nodes by a defining certain characteristics of the collection. For example, a string of connected nodes can be defined in a policy by nominating only the starting node and ending node. As a more extended example a provenance logs that a student generates an assignment and the student modified this assignment a few times. The provenance also records that the student submitted the assignment to a professor and actions from the professor to grade it, attach comments, revisions, *etc.* Students are not allowed to know all the operations that happened after they submitted their assignments. However, all the nodes of the provenance subgraphs are hard to be listed, as they cannot be predicted. In this scenario, a policy in our proposal can be defined in such a way: the subgraphs that include nodes after “submit" cannot be accessed by students. Therefore, we define provenance segments to facilitate the definition of a collection of nodes in provenance DAG by a summarised approach.

Moreover, to the best of our knowledge, we are the first to introduce provenance segments as elements of provenance access control. Hence, in this section, we extend and complete existing provenance access control language to address these issues and make the following contributions.

- We provide the extended OPM which can support more fine-grained access control policy.

- We define different types of provenance partitions which are collections of nodes in provenance graphs.
- We propose a Segment-based Access Control Policy Language for Provenance (PACLP) by extending existing policy languages.
- Algorithms to retrieve applicable policies for a request and merge results of individual policies as a final decision are proposed. In addition, we present algorithms to transform an existing provenance DAG into a new graph which can be returned to users.

3.1.3 Chapter Organisation

The construction of this chapter is organised as follows:

- A discussion of provenance access control demands and the motivation to generate PACLP are present in Section 3.2.
- In Section 3.3, various types of segments of provenance graphs over XACML syntax and other basic of the language and algorithms are defined.
- Section 3.4 illustrates PACLP by defining language items.
- Section 3.5 presents algorithms to retrieve applicable policies and combine results of policies followed by conclusion in Section 3.6.

3.2 Provenance Access Control

3.2.1 Workflow of the Framework

The framework to answer an access request for a provenance graph is twofold and this is shown in Figure 3.1. Generally, it returns a new version of a provenance graph by answering a request from the users. Algorithm 2 retrieves applicable access control policies for a given request, as the target item for each policy confines matching subject and object of the policy. Then, algorithm 3 outputs a collection of allowed and prohibited vertices for each applicable policy and combines results of policies based on policy priorities. The policies also define

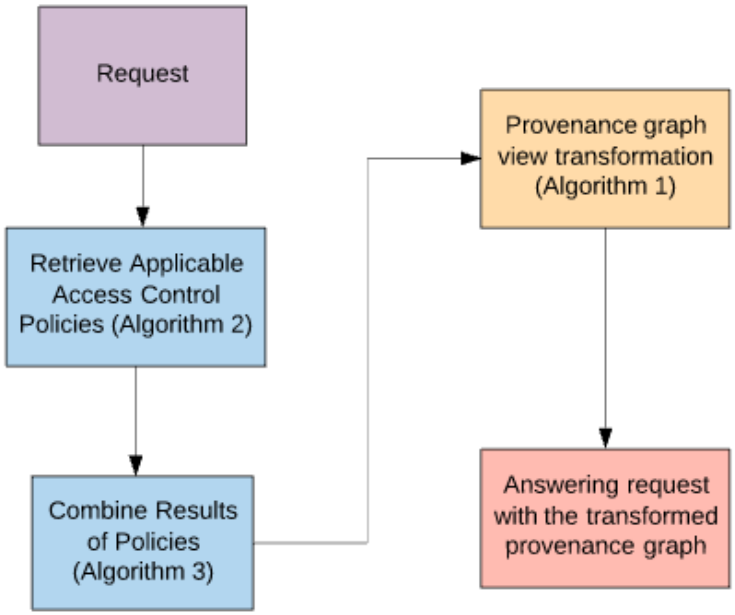

Figure 3.1: Workflow of the Framework

whether the collected vertices are needed to unite with relevant vertices in the graph, in case vertices cannot be listed thoroughly when policies are tailored.

The targeted provenance graph will then be transformed according to the results of policies. For “permit-take-precedence", it reveals the collection of allowed vertices as a new provenance graph. For “deny-take-precedence", it hides prohibited vertices by deleting or replacing clusters of vertices, where the clusters of vertices are divided in the Algorithm 1. In particular, in provenance-aware systems, vertices for each category and labels are listed by Vertices Cluster Dictionary. The policy collected vertices will be united as clusters according to the Vertices Cluster Dictionary, and replaced vertices will be substituted by the corresponding labels. In addition, the clusters of vertices will be joined with relevant vertices if it is required by provenance access control policies.

3.3 The Basics of the language

In this section, we define the basics of our provenance access control language including several types of provenance segments over an extended OPM and how to express them by XPath. Briefly, these collections of nodes can be defined by node classifications, node

attributes or terminals of a string of nodes. In terms of the motivation, an access control policy might be unable to predict all the nodes in advance; or a policy just summarily defines provenance segments to meet certain features instead of listing all the nodes thoroughly. Firstly, we need to define our extended version of the OPM.

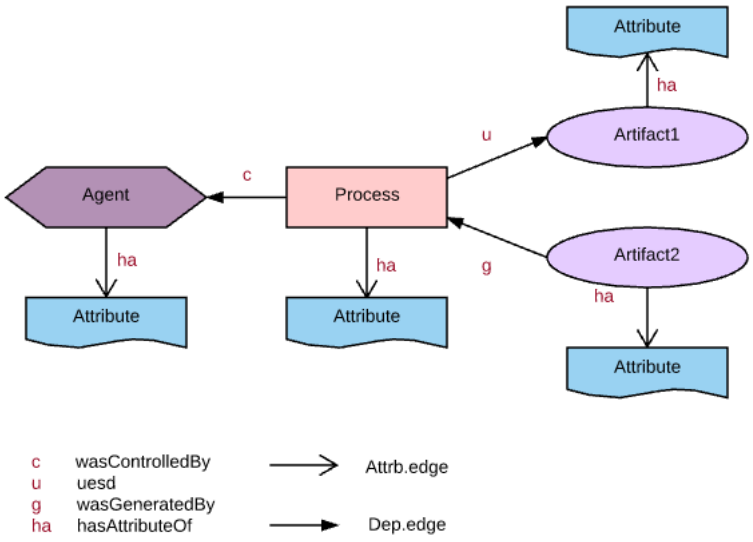

Figure 3.2: OPM$^+$ Schema

Definition 1 (Open Provenance Model$^+$(OPM$^+$)) is an extension of OPM, which records how is a piece of data derived. The model is defined by a triple <T, L, G>:

- T is the vertex types: agent (Ag), artifact (A), process (P) and attribute (Att). Each vertex in a provenance graph is one of these types. In Figure 3.2, an artifact is represented by the shape of oval, which is an object or as a piece of data, such as "$homework_1$", "*comments*", and *etc.*; a process is an action which is the operation executed on a piece of data, such as "*submit*" and "*review*"; an agent is a subject that sponsors an action including "$user_1$" and "*professor*".

- L is the relationship labels: used (u), wasGeneratedBy (wgb), wasControlledBy (wcb), wasTriggeredBy (wtb), wasDerivedFrom(wdf) and hasAttributes (ha). Each edge in a provenance graph will be labeled as one of these labels. The Labels describe the relationships between the vertices.

- G is a labelled DAG, where G = <V, E>, E defines the allowable relationships between the elements, E = { (P, A, used), (A, P, wgb), (P, Ag, wcb), (A, A, wdf), (P, P, wtb),

(Ag, Att, ha), (P, Att, ha), (A, Att, ha) }

Given the OPM$^+$ <T, L, G>, an OPM$^+$ instance is defined by a provenance graph G_i = $< V_i, E_i >$, where V_i is a set of entities and $E_i \in V_i \times V_i \times L$. Let τ: $V_i \rightarrow T$ be a function that maps an entity to its type, we say G_i is valid if for each entity v $\in V_i$, $\tau(v) \in$ T, and for each edge (v, v′, l) $\in E_i$, (τ (v), τ (v′), l)$\in$ E. We extend the definition of OPM in paper [45].

Definition 2 (Attribute Node) The extended provenance model attaches context information as attribute nodes in DAG, where each attribute node consists of attribute items and attribute values. For instance, let $Attribute_i$ for a process be {timestamp: 1/5/2017; system condition: Linux; location: Sydney}. In this model, provenance can be classified into base provenance data and (optional) attribute provenance data that is associated with main entities in the graph (ag, p, or a). Attribute provenance data is classified into three categories: Agent-related Attributes, Process-related Attributes, and Artifact-related Attributes.

- Agent-related Attributes: Agents trigger and execute operations, and their attributes include IDs, activated roles *ect.* Identities are usually unique labels to identify users. Activated roles are the roles users employ by taking actions and can play a key role in distinguishing the sensitivity of content and in making access decisions. For instance, when Alice adds a piece of data into a document in the role of teaching assistant, she grades assignments of students. The comments and scores can only be accessed by Alice and the owner of assignments. While when Alice edited data with a role of student, the content of assignments can be read by other students enrolled the same subject.

- Process-related Attributes: Processes are operations performed on data and result in the change of data, their related attributes include temporal aspects when operations are performed, such as locations, timestamps, system conditions *etc.* They can influence access decisions. For example, operations in provenance can be accessed if they were executed before 2016.

- Artifact-related Attributes: Artifacts are objects including input messages, output messages, and source data. The related attributes can include object size, permitted usages defined by the data producers *etc.* Some meta-data that might normally be recorded as attributes may not be held in this way in provenance data, as they are

recorded within the structure of a provenance graph. For example the generating agent and time of data.

Particularly, the dependency linking the attribute node with the main entity is “att (has attributes of)". There are two approaches to capturing attributes nodes in the graph-based data model [14].

- The attributes of vertices *ag*, *p*, or *a* recorded as individual sets that are attached to the corresponding vertex.
- The attributes of vertices *ag*, *p*, or *a* recorded as one set which is attached to the *p* vertex.

Nguyen *et al.* prefer to attach attributes of one operation as one attribute set to *p* vertex. We choose the former approach, i.e., storing attribute sets separately, where each attribute set links to the node it describes, in order to identify a node directly according to its attached attributes. Specifically, vertices can be defined by their attributes in PACLP. For example, a policy can define all *agent* vertices attached to a role of “doctor" can be accessed. It can be noticed that the attribute “doctor" storing at a vertex of *agent* is more convenient to selecting targeted vertices, comparing with storing it at a close *process* vertex. Following, we define partitions of provenance DAG based on the extended OPM model.

Definition 3 (Typed Vertex) A collection of nodes can be defined by the vertex types: Agent (ag), Process (p), Artifact (a), and Attributes (att). Each typed vertex collection is a set of nodes in a provenance DAG. Hence, for a graph $G_i = < V_i, E_i >$ we define a family of functions $TypedV_T\ (G_i)$, as

$$V_T = TypedV_T\ (G_i), \forall v_m \in V_T,\ v_m \in V_i \bigwedge \tau(v_m) = T:$$

$$\text{(I) } V_{Ag} = TypedV_{Ag}\ (G_i), \forall v_m \in V_{Ag},\ v_m \in V_i \bigwedge \tau(v_m) = Ag$$

$$\text{(II) } V_A = TypedV_A\ (G_i), \forall v_m \in V_A,\ v_m \in V_i \bigwedge \tau(v_m) = A$$

$$\text{(III) } V_P = TypedV_P\ (G_i), \forall v_m \in V_P,\ v_m \in V_i \bigwedge \tau(v_m) = P$$

$$\text{(IV) } V_{Att} = TypedV_{Att}\ (G_i), \forall v_m \in V_{Att},\ v_m \in V_i \bigwedge \tau(v_m) = Att$$

Each member of the family of functions returns all vertices of the specified type in a given provenance graph. For the one labeled (I), it returns all the agent vertices in provenance DAG G_i. Similarly, we define another function to return named vertices of a given type.

$$V'_T = TypedV'_T\ (G_i; Vertice_{name(a)}, Vertice_{name(b)}, ...Vertice_{name(n)}),$$

$$\forall v_m \in V'_T,\ v_m \in V_i \bigwedge \forall \tau(v_m) = T$$

The function V'_T which defines vertices in provenance graph G_i under the type of T with certain vertices names. For example, $V'_P = TypedV_T\ (G_1; Grade, Submit)$ identifies all nodes in the graph named “Grade" or “Submit" under type P.

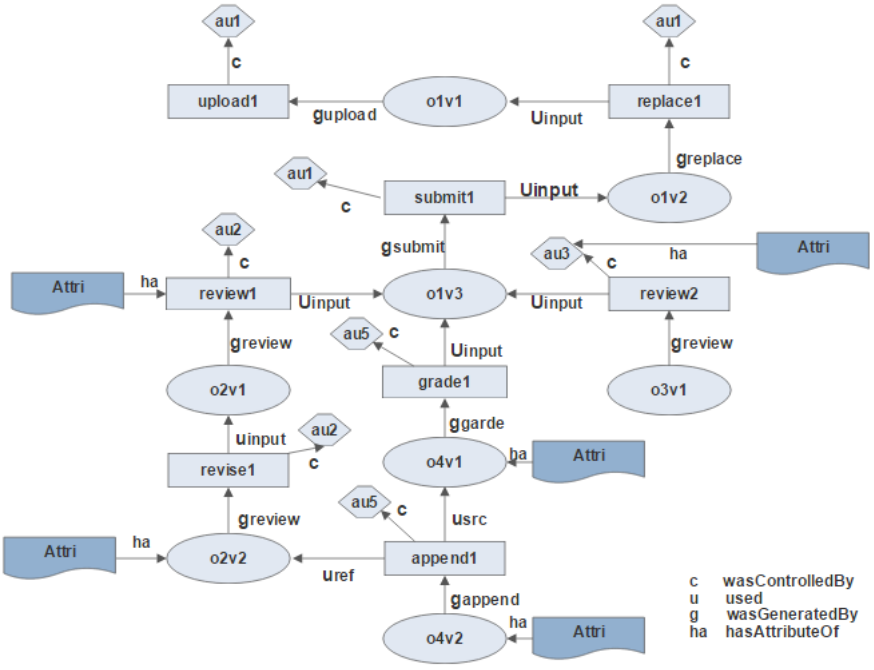

Figure 3.3: Sample Provenance Graph under OPM$^+$

Definition 4 (Attributed Vertex) Vertices can be defined by their attached attribute values as well. We define the functions as below:

$$V_{Att} = AttV\ (G_i; Attibute_a, Attibute_b, ...Attibute_n)$$

$$\text{where } \forall v_m \in V'_{Att},\ v_m \in V_i \bigwedge \exists v'_m \in V_i \bigwedge \tau(v'_m) = Att$$

$$\&(v_m, v'_m, ha) \in E_i,\ Attibute_a | Attibute_b | Attibute_n \in v'_m$$

$$\textbf{and } V'_{Att} = AttV_T\ (G_i; Attibute_a, Attibute_b, ...Attibute_n)$$

$$\text{where } \forall v_m \in V'_{Att},\ v_m \in V_i \bigwedge \exists v'_m \in V_i \bigwedge \tau(v_m) = T \bigwedge \tau(v'_m) = Att$$

$$\&(v_m, v'_m, ha) \in E_i,\ Attibute_a | Attibute_b | Attibute_n \in v'_m$$

The functions above return a collection of vertices that is attached by given attributes. The difference between the two functions is that the latter one only returns vertices under given types. For instance, $V_{Att} = AttV\ (G_1; [1/1/2016 - 30/6/2016])$ defines process nodes which are executed in the first half year of 2016 in provenance graph G_1. In terms of the motivation to define Attributed Vertex, in most scenarios, exact values of nodes are difficult to predict. Hence, a specific vertex in a provenance graph is required to be identified by its attaching attributes as well. In addition, this mechanism enables the definition of nodes to be more efficient.

Definition 5 (Effect and Cause Edges). Dependency edges are classified as *Cause Edges (CE)* and *Effect Edges (EE)*. For a given vertex, we define its connected edges linking to the vertex that triggers it as *CE*, and its connected edges linking to the vertex that results from it as *EE*.

We distinguish directions of edges aiming to ensure operations in a provenance path happened sequentially. Specifically, if a string of vertices which are all linked by CE, we can conclude that the operations in a provenance path happened in sequence.

Definition 6 (Provenance Path). A path p = { (v_i, v_{i+1}, ... v_{i+n}) | n≥2}, starts from v_i and ends at v_{i+n}, which is a collection of vertices that forms a line in a provenance DAG. Moreover, from v_i to v_{i+n}, if all the vertices are connected by cause edges in a path, we define it as a *directional provenance path*. It indicates that all operations in a *directional provenance path* happened in a timed sequence. While, a *general provenance path* can include effect edges, where processes recorded in a *general provenance path* may not happen according to the time order.

There are two approaches to defining vertices in a provenance path. One is to define a vertex by the functions defined above, including $TypedV_T\ (G_i)$, $TypedV'_T\ (G_i)$, $AttV\ (G_i;$

$Attibute_a$, $Attibute_b$, ... $Attibute_n$), and $AttV'$ (G_i; $Attibute_a$, $Attibute_b$, ... $Attibute_n$). The other one is to define the names of vertices straightly, such as “submit", “grade", “homework" *etc.* It should be noticed that the names in access control policies are abstract names which are a class of vertices. For example, “homework" is the abstract name for “$homework_1$", “$homework_1$", ... “$homework_n$". We propose a sample provenance graph under OPM$^+$ in Figure 3.3. We generated sample provenance graphs Figure 3.3 - Figure 3.7 based on example figures in Park *et al.*'s paper [49].

(XPath Symbols). We employ XPath (XML Path Language) expression to define provenance path. XPath is a query language defined by the World Wide Web Consortium (W3C) for selecting nodes from an XML document. In addition, XPath can be used to compute values (e.g., strings, numbers, or Boolean values) from the content of an XML document[102].

Provenance paths are defined over XPath. For example, a directional provenance paths can be defined as (v_i//v_j/v_k) describes provenance paths that start from vertex v_i and end by v_k, include v_j with at least one vertex between v_i and v_k. In this directed provenance path, vertices are connected by effect edges only.

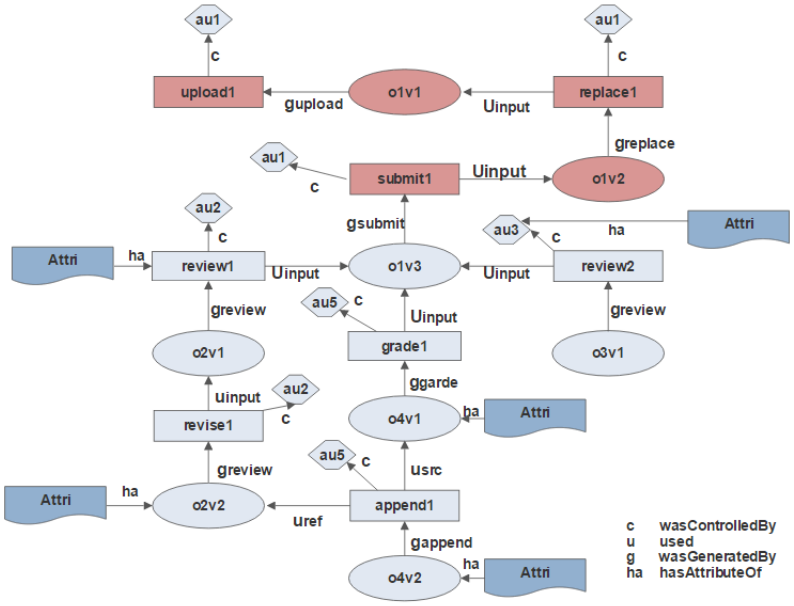

Figure 3.4: Sample Provenance Path A

In path expressions, it will be useful to be able to distinguish between directional and general paths. Keyword qualifiers are used in our language for this purpose. For example:

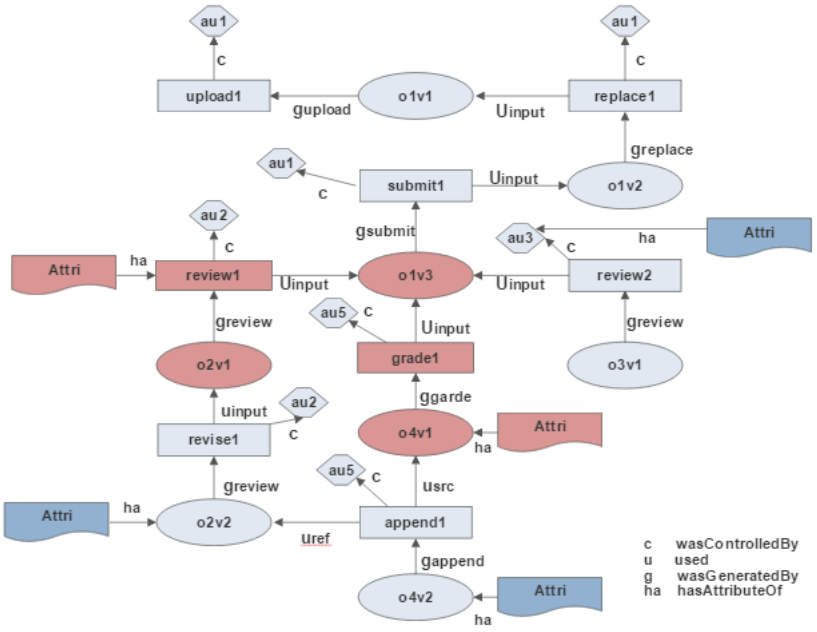

Figure 3.5: Sample Provenance Path B

- directional $(TypedV'_P(G_i, upload)//TypedV'_P(G_i, submit))$

  The first example will match any directed provenance path which starts from a *Process* vertex with a name of $upload_1$ and ends at a *Process* vertex with a name of $submit_1$. The details of the syntax will be explained below. Because it is a directed path, all the nodes in the path are connected by *CE*. In Figure 3.4, the example expression matches the illustrated the provenance path, which can also be represented as follows:

  $\rightarrow$ (upload1, o1v1, replace1, o1v2, submit1): <upload1, o1v1, g_{upload}>, <o1v1, replace1, u_{input}>, <replace1, o1v2, $g_{replace}$ >, <o1v2, submit1, u_{input} >.

- general $(TypedV'_A(G_i, o2v1)//TypedV'_A(G_i, o4v1))$

  The second example will match any general provenance path which starts from a *Artifact* vertex with a name of $o2v1$ and ends at a *Artifact* vertex with a name of $o4v1$. The details of the syntax will be explained below. Different from directed path, the edges from the two terminals of a path could include *EE*. In the Figure 3.5, the example expression matches the illustrated the provenance path, which can also be represented as follows:

  $\rightarrow$ (o2v1, review1(attri), o1v3, grade1, o4v1(attri)): <o2v1, review1, g_{review}>, <review1, o1v3, u_{input}>, <o1v3, grade1, u_{input}>, <grade1, o4v1, g_{grade}>.

- directional $(TypedV'_{Ag}(G_i, au1), \backslash v+, TypedV'_A(G_i, o1v2))$

The last example will match any directed provenance path which starts from an *Agent* vertex with a name of $au1$ and ends at a *Artifact* vertex with a name of $o1v2$. In the given example graph, there are three coincident paths.

→ (au1, upload1, o1v1, replace1, o1v2): <au1, upload1, c>, <upload1, o1v1, g_{upload} >, <o1v1, replace1, u_{input} >, <replace1, o1v2, $g_{replace}$ >;

→ (au1, replace1, o1v2): <au1, replace1, c>, <replace1, o1v2, $g_{replace}$ >;

→ (au1, submit1, o1v2): <au1, submit1, c>, <submit1, o1v2, u_{input} >.

Particularly, even though in the given sample, we distinguish the same type of operations, replace as “replace1" and “"replace2". In policies, the same type of operations is usually viewed equally, regardless of their appearing orders in a graph.

Definition 7 (Subgraph). Let G=<V, E>, S=<V′, E′ >. S is a subgraph of G, if V′ ⊆ V, E′ ⊆ E. Namely, a subgraph is a set of vertices { (v_j, v_{j+1}, ... v_{j+n}) | n≥2} in a provenance DAG. A partition P of a graph G is a connected subgraph of G.

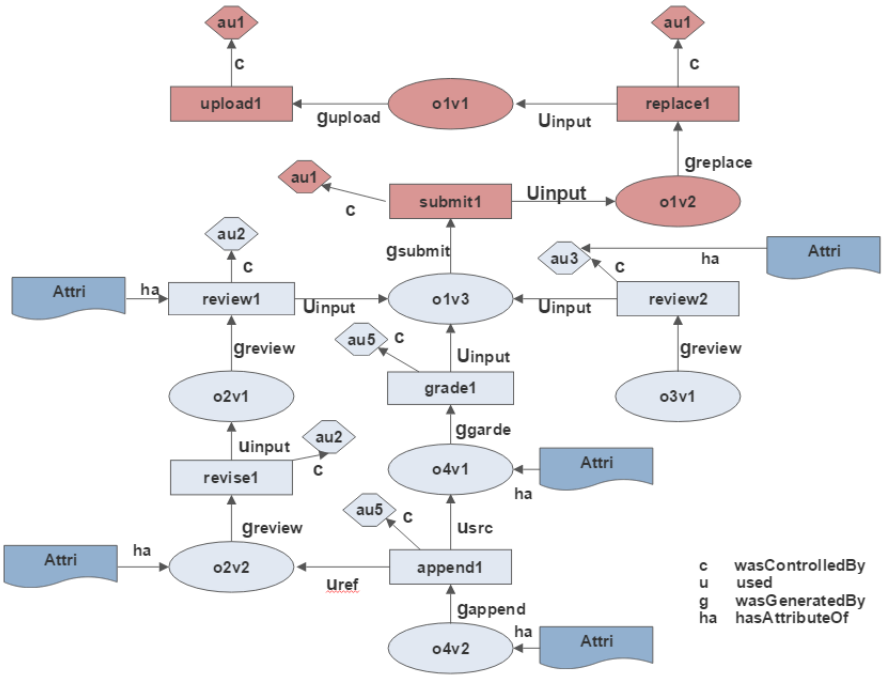

Figure 3.6: Sample Subgraph A

In our policy language, subgraphs can be defined by specifying a vertex expression. Namely, a subgraph that consists of the given vertices is the target of a policy. For instance, a

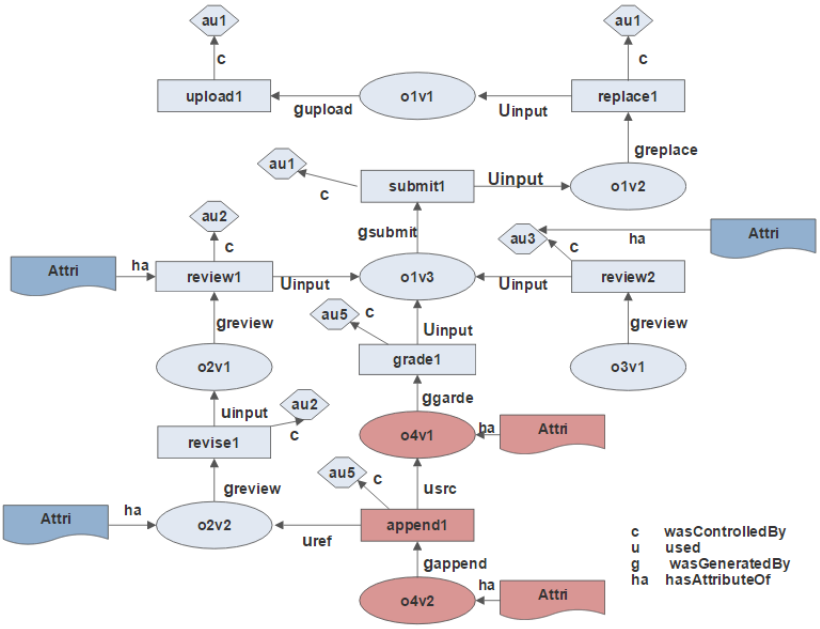

Figure 3.7: Sample Subgraph B

policy defines a subgraph by nominating a starting node and an ending node of the subgraph as the subgraph (v_i// v_j), where v_i and v_j are the starting and ending points respectively.

Particularly, a subgraph may start or end at terminals of a provenance graph regardless of what exactly the vertices are. Hence, in PACLP language, a terminal of a provenance is expressed as $\triangleleft$ which is a starting point of a provenance DAG and $\triangleright$ which is an ending point of a provenance DAG. To be more specific, subgraph (v_j/following::*) are those graphs beginning at the starting vertex of a provenance graph and ending at Vertex v_j. subgraph (v_i/preceeding::*) are those subgraphs starting at v_i and finishing at the ending vertex of a provenance graph.

The Figure 3.6 highlights a subgraph ($upload//submit1$), which begins with a node “upload 1" and ends at a node “submit 1". Between the two terminals, it concludes all nodes including vertices of the type of *Agent*. subgraphs ($o4v1$/following::*) is another example between vertices $o4v1$ and the end of the provenance DAG, shown in Figure 3.7.

Definition 8 (Vertices Category Dictionary (VCD)). This is a document listing categories of vertices, and each category collects vertices under a semantic domain. Namely, vertices within one category are related to each other, they record entities for related operations. Every category is referred by a label such as “Clinical Trial", “Library", etc. When nodes in a category are replaced for transformation, they will be replaced by the label which is also

defined in VCD. We present two examples below. They list vertices for two categories with labels *University* and *Hospital* .

VCD will be utilised in Algorithm 1 to transform the view of provenance graphs according to access control policies, where VCD determines which clusters of nodes should be processed as a union. If the relevant policy requires replacement of particular nodes by a label, then the collection of nodes under a category is replaced by the label of the category. Moreover, during the transformation of provenance graphs, if required by policies, a set of nodes defined in policies should be extended to cover other nodes within the same category in the provenance graph. The following is an example of a VCD.

University: "Submit|wasSubmittedBy", "Grade|wasGradedBy", "Review|wasReviewedBy", "Professors", "Teaching Staff", "Student", "HDR Candidates", "Assignments", "Research Fellows", "Exams"
Hospital: "Diagnose|wasDiagnosedBy", "Monitor|wasMoniteredBy", "Surgery", "Nurses" "Doctors", "Patients"

Table 3.1: Example Vertices Category Dictionary

Following, we define two functions that are used in the algorithms. They are conjunction and extension. A cluster is a partition that is identified for the purposes of deletion and replacement for transformation. All vertices in a cluster must have the same label from the VCD. Let C be a cluster of nodes in a provenance graph $Prov$= (V, E), C $\subseteq$ V. Category() is a function returns the label for vertex/vertices, which returns null if the vertices set do not take a shared label. The function *Con* extends a cluster with all connect's vertices that are within the same category defined in VCD, where:

• C' = Con (C) = { C$\cup$V$'$: $\forall v_i \in V'$, $v_i \in V$ & Category (v_i) = Category (C) & $\exists e_i \in E$: $(v_i, v_j, l) \rightarrow v_j \in$ C }

We employ the function *Con* in Algorithm 1 for the viewing the transformation of provenance graphs, because each cluster defined in VCD tends to be a collaboration of operations.

Hence, the neighbour nodes under the same category might reveal clues to the hidden cluster of nodes. Thus, depending on different security requirements, relevant connecting nodes should be joined within the hidden clusters when they are defined as *Conjunction* transformation mode in policies.

Let C be a cluster of nodes in a provenance graph *Prov*= (V, E). The function *Ex* combines all nodes in the provenance DAG, if they are within the same category defined in VCD, where:

• $\mathscr{C}'$ = Ex (C) = {$\forall C_i \in \mathscr{C}'$, Category ($C_i$) = Category (C), $C_i \subseteq V$; & $\forall C_j \in (\mathscr{C}' - C_i)$ $(C_i \cap C_j) = \emptyset$ }

Particularly, *Ex* may generate a cluster set which may consist of more than one set. We employ the function *Ex* in Algorithm 1. For higher security requirements defined by access control policies, all nodes in the provenance DAG of the same category should be collected for transformation.

Definition 9 (Partitions Removal Operator Rem_P). Let $P = (V_R, E_R)$ be a partition of provenance graph G = (V,E), $V_P \subseteq V$, $E_P \subseteq E$, which needs to be removed from the graph, in order to generate a new graph $G' = \{V', E'\}$ = Rem_P (G, P), V_{Rattri} is the attribute vertices of V_P, where:

•V' = V $\setminus (V_P \cup V_{Rattri})$

•E' = E $\setminus \{\forall e_i : e_i \in E, e_i = \{v_i, v_j, l_k\} \& ((v_i \notin v' \& v_j \in v') or (v_i \in v' \& v_j \notin v'))\}$

• $type'$ morphism function:

-$\forall v \in V', type'(v) = type(v)$

-$\forall e \in E' \cap E, type'(e) = type(e)$

$$\text{-}\forall e \in E' \setminus E, type'(e) = \begin{cases} wdf & if \quad type(e) = wdf^{+} \\ u & if \quad type(e) = u^{+} \\ wgb & if \quad type(e) = wgb^{+} \\ wtb & if \quad type(e) = wtb^{+} \\ c & otherwise \end{cases}$$

The function Rem_P generates a new provenance graph $G' = \{V', E'\}$ by removing the collection of nodes P from $Prov$. Rem_P removes all the nodes in P including all the attributes attached to nodes in P, and it deletes the edges in P and those connecting P with effect or causes nodes. In terms of the types of edges, they are changed from indirect relationships to their analog direct relationships or “caused by".

Definition 10 (Partitions Replacement Operator Rep_P). Let $P = (V_P, E_P)$ be a partition of provenance graph G = (V, E), $V_P \subseteq V$, $E_P \subseteq E$, which needs to be replaced from the graph, in order to generate a new graph $G' = \{V', E'\}$ = Rem_P (G, P),, V_{Rattri} is the attribute vertices of V_P, where:

•V' = V $\setminus (V_P \cup V_{Attri}) \cup \{v_a\}$

•E' = E $\setminus \{\forall e_i : e_i \in E, e_i = \{v_i, v_j, l_k\} \& ((v_i \notin v' \& v_j \in v') or (v_i \in v' \& v_j \notin v')) \cup \{(v_P, v_c) \in e | v_c \in ca(P), v_P \in P\} \cup (v_a, v') \in e\}$

• $type'$ morphism function:

-$\forall v \in V' \cup \{v_a\}, type'(v) = type(v)$

$$\text{-}type'(v_a) = \begin{cases} Artifact, & if \forall v \in R, type(v) = Artifact \\ Process, & otherwise \end{cases}$$

-$\forall e \in E' \cap E, type'(e) = type(e)$

-$\forall e \in E' \setminus E, type'(e)$ is defined according to the TG graph.

The function Rep_P deletes P in the original graph and replace it with v_a which is a label that can be checked by a dictionary. Then, it generates a new provenance graph $G' = \{V', E'\}$. The Rem_P and Rep_P are defined by following the functions in Danger *et al.*'s paper [44].

	wdf	u	wgb	wcb	wtb
wdf	wdf^+	$0\ .^*$	wgb^+	0	0
u	u^+	0	wtb^+	0	0
wgb	0	wdf^+	0	c^+	wgb^+
wcb	0	0	0	0	0
wtb	0	u^+	wcb^+	wtb^+	0

Table 3.2: Edge Merging Table

wdf	Artifact was derived from another artifact
wdf^+	Indiract was derived from relationship
u	Process used an artifact
u^+	Indirect used relationship
wgb	Artifact was generated by a process
wgb^+	Indirect was generated by relationship
wcb	Process was controlled by an agent
wcb^+	Indirect was controlled by relationship
wtb	Process was triggered by another process
wtb^+	Indirect was triggered by relationship
c	Two entities are causally related
c^+	Indirect was caused by relationship

Table 3.3: Annotation of Edges

Definition 11 (Merging Labels). When a vertex is deleted for transformation, edges beside vertex node should be merged as a new edge to generate a connected new graph. How to generate a new label is determined by the *Transformation Label* defined in access control policies. Specifically, if the value of Transformation Label is *Original dependency*, the two edges should be combined based on their original values. Each possible combination of two edges is mapped to a merged edge type, which is defined in the Edge Merging Table. On the other hand, the other value of Transformation Label is *Fault dependency*, where a merged edge type is transformed as “wasCausedBy" regardless of what the original labels are.

Because edges of vertices can reveal the types of vertices and further leak clues for provenance graphs. Organising the transformation type of edges is defined in access control policies according to specific security requirements.

3.4 Partition-based Access Control Language (PACLP)

Based on the demands for provenance access control discussed above, we propose Partition-based Access Control Policy Language on provenance (PACLP) which extends the access control language [42], which enables a policy to determine allowed or prohibited provenance partitions for accessing. The users send requests about which provenance graphs they would like to access, and the policies tailored under PACLP languages defines which graphs or partitions can be accessed. Our proposed PACLP language facilitates several contributions. Firstly, language item PACLP tailored under the OPM$^+$ which stores attributes in order to support more fine-grained policies. Secondly, PACLP utilises provenance partitions we defined which are expressed over *XPath*, where each policy determines which collections of nodes in a provenance DAG can be accessed. Therefore, it maximises access to available information instead of hiding a whole graph to protect partially unavailable attributes in a provenance graph. Thirdly, another difference of PACLP as distinct from traditional access control is that we employ provenance partitions as conditions in policy, as provenance partitions provide clues for data sensitivity and vulnerability.

3.4.1 The System Assumption

The system model for PACLP is shown in Figure 3.8, where the system consists of four parties which are *Administrator*, *Server*, *Users*, and *Database*.

Users send queries to the *Server* in order to access provenance stored in a database.

Administrator generates access control policies and sends them to the *Server*.

Server collects policies from administrators and (optional) data producers. When Server receives queries from users, it generates results based on the policies and delivers the results to the database.

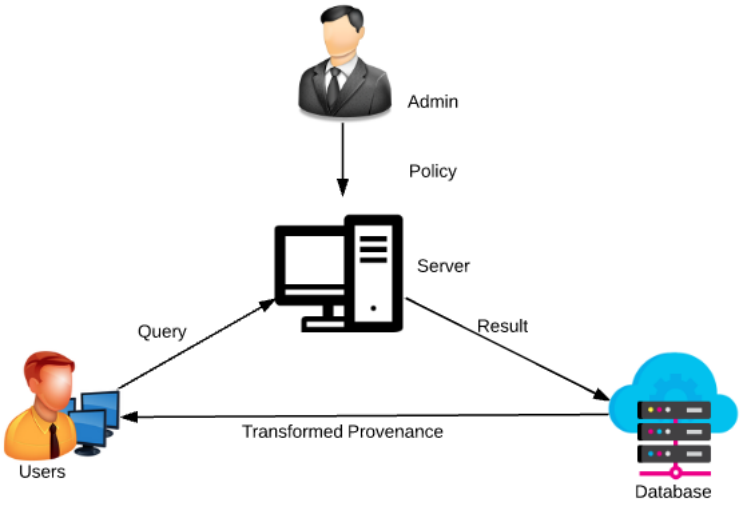

Figure 3.8: System Model

Database transforms the target provenance graph based on the results to hide unavailable partitions and sends it to users.

3.4.2 Language Items

Our provenance access control policy PACLP was tailored over XACML syntax, which consists of Target, Condition, Obligation, Effect, and Transformation items. Each item includes one or more tags. Initially, PACLP is constructed over the OPM$^+$ which attaches attributes sets to main entities, in order to support more fine-grained access control policies. Moreover, based on provenance partitions we defined above, PACLP employs partitions as conditions to confine applicable policies and determines accessible or prohibited partitions in order to transform a version for requestors. In this section, we define each language items in PACLP.

Each policy consists of five items including *Target*, *Conditions*, *Obligations*, *Transformation*, and *Effect*. *Target* contains *Subject*, *Object*, *Access Purpose*, and *Restriction* tags. *Transformation* contains *Provenance Partition*, *Transformation Scope*, *Transformation Mode*, and *Transformation Label*. There are certain items and tags that are optional in a policy, which will be shown in details below.

Target consists of three tags including *subject*, *object*, *access purpose* and *restriction*, which specifies the effective scope of a policy. Only when a request matches the *target*, the

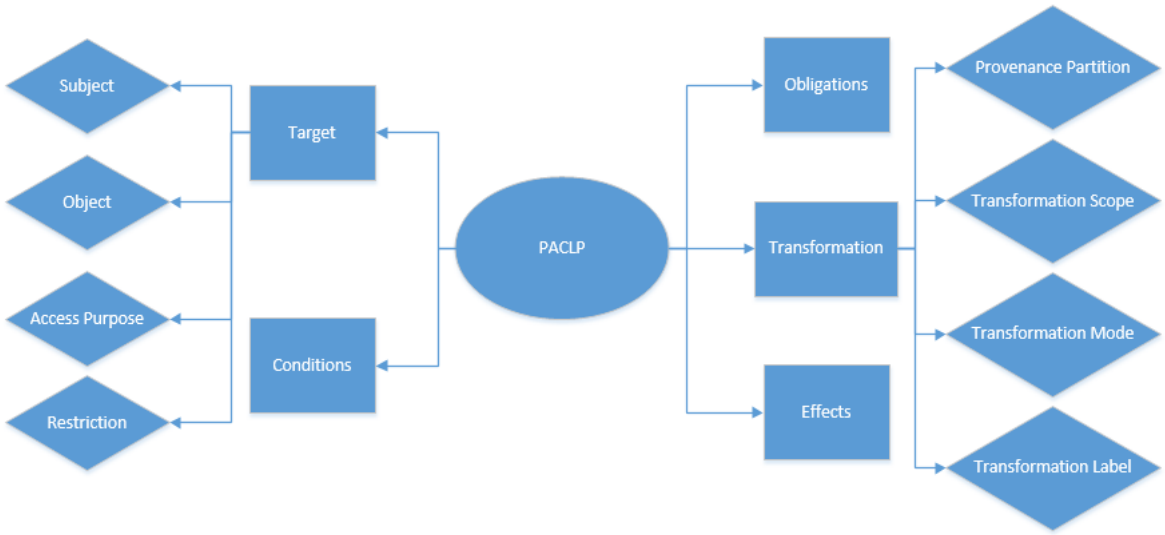

Figure 3.9: PACLP Schema

policy is applicable to the request. To be more specific, only when the requestor's name, objective provenance graphs in a request and contextual information meet conditions defined in *Target* of a policy, the policy imposes its effects on the request and generate a result.

We believe that we are the first to introduce our self-defined provenance partitions as conditions in a provenance access control language. Most existing provenance access control languages only utilise traditional conditions such as *subjects*, *objects*, *actions* and certain *attributes* to determine and control accessibility. However, we believe, to control accessibility for provenance graphs, record items in provenance should also be used as conditions in policies. In a simple example, if a provenance contains a record "$Revise \mid wasRevisedBy$", then subgraph after the first process of "$Revise \mid wasRevisedBy$" can be accessed by the requestors. Semantically, records or partitions in provenance could be criteria to determine the sensitivity of provenance graphs. A provenance DAG includes certain records which might be more sensitive in a system. For instance, if there is a record "collecting data by government security department" or "security department labeled a piece of data as sensitive information", then the following processes recorded in provenance might be relevant with sensitive operations which can not be accessed by the public. Hence, provenance partitions should be introduced as a condition for access control. Below, we introduce the four items respectively.

- Subject can be any collection of names or roles, such as students, university staff,

market manager and even any user. Particularly, each category is compatible with its sub-category. For instance, {*teaching assistant*} ⊆ {*university staff*}, as a teaching assistant is a type of university staff. Namely, when a requestor with a role of *teaching assistant*, meets the required role of *university staff* in *target*.

- Object is a collection of provenance graphs, which is listed as graph ID or names, a category of graphs *etc.* For instance, {provenance graph 001-008} is a set of provenance graphs with the ID number from 001 to 008.

- Access Purposes is a set of purposes for which the requestors visit the object. The example purposes could be *education*, *governmental supervision*, *research etc.* This is an optional item in access control policies.

- Restriction is an optional element to specify restrictions to a subject and/or object, which is in conjunction with *subject* and *object* to confine the applicable targets of a policy. As the objects are provenance DAG, the restriction can be provenance partitions which the provenance DAG should contain or avoid.

In the target item, we omit the *scope* element from the original language established in Ni *et al.*'s paper. As their access control language is based on their self-defined provenance model, where each provenance graph under the model logs attributes for one operation. Correspondingly, the *scope* element specifies if the evaluated results can be extended to its ancestor graphs. However, our access control language is based on the extended OPM by which operations performed on a piece of data are organised in one graph. In addition, the default action to access a provenance graph is *read*. Hence, we omit the *action* elements in target and assume the access actions in all the queries are *read*.

Particularly, in our language, *restriction* only confines the effective scope of policies, which is different from the language proposed by Danger *et al.* [44]. They defined that nodes can be accessed or prohibited in *Target* for their language. To reduce redundancy, we re-organise the syntax and the structure of the language, by displaying provenance partitions for access to the item of *Transformation*. In other words, in PACLP, *Target* is tailored only for matching a policy with queries instead of nominating the accessible or deniable provenance subgraphs. Here, we present a simple example to illustrate the differences. <restriction> agent.medical record == ABC clinic </restriction>, it screens target graphs with a record "medical records are generated by ABC clinic", instead of permitting or denying access to

this piece of record in provenance DAG.

Introducing provenance partitions into *Target* facilitates fine-grained policies. It enables a policy to define that the given graphs of the policy should contain nominated partitions. In term of semantic meaning, it defines provenance graphs that contain certain partitions are within the effective range of the policy. Just as in the example, the graphs must contain two nodes which are “wasSubmittedBy" or “Submit" and “wasGradedby" or “Grade". As they are in a directed path, where all the vertices are connected by cause edges *CE*, the operation of *Grade* must be performed after *Submit*.

```
<Target>
<subject> students </subject>
<object> any provenance graph </object>
<restriction> timestamp.initial_operation ≤1/1/2016; directed provenance
path (wasSubmittedBy|Submit//wasGradedby|Grade) </restriction>
</Target>
```

The example *Target* defines that the policy governs queries from users with a role of student. For the objective provenance graphs, the first operation recorded in the provenance DAG must be executed before 1/1/2016. In addition, the graphs must contain a path that starts with a vertex named “wasSubmittedBy" or “Submit" finish at a vertex named “wasGradedby" or “Grade", regardless of how many vertices are recorded between them. Moreover, in this example, initial operation refers to the earliest process in a provenance graph.

As each language element might consist of assertions, we introduce the access tree structure [103] to facilitate the organisation of conditions within each element.

Access Structure. This is a new item in our language. We replace the subject with an attribute tree in each policy. Attributes trees are set with monotone access structure. We assume $\{P_1, P_2, ...P_n\}$ be a set of parties. A collection A $\in 2^{\{P_1,P_2,...P_n\}}$ is monotone if $\forall$ B,C: if B $\in$ A and B $\subseteq$ C then C $\in$ A. An access structure (respectively, monotone access structure) is a collection (respectively, a monotone collection) A of non-empty subsets of $\{P_1, P_2, ...P_n\}$, i.e., A $\in 2^{\{P_1,P_2,...P_n\}} \setminus \{\emptyset\}$. The sets in A are called the authorised sets, and the sets not in A are called the unauthorised sets. In our policies, the parties are taken as attributes. Then, the

access structure A contains a set of authorised attributes.

Attribute Tree. Let $\mathcal{T}$ be a tree with root r. The leaf nodes are set as attributes, and each non-leaf node represents a threshold gate which is described by a threshold value and its children. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x < num_x$. When $k_x = 1$, the threshold gate is an *OR* gate; when $k_x = num_x$, it is an *AND* gate; when $1 < k_x < num_x$, it is an n-in-m gate, it exists to satisfy at least any k_x of its children attributes.

Satisfying the Access Tree. Assume $T_x(r)$ is a subtree of $\mathcal{T}$ with a root node x. For every subtree, if x is a non-leaf node, it returns 1 if and only if at least k_x of its children nodes returns 1; if x is a leaf node, it returns 1 if the attribute satisfies x or belongs to a subclass of x. The example below indicates that the applicable subjects have to be professors in the computer science or electrical engineering department who are under 55 years of age.

Condition confines the applicable access requests in terms of context conditions. This optional item sets restrictions on context conditions of requests, e.g. limitation on access time and location. It is a crucial item to achieve fine-grained access control. The following example *condition* restricts access only to be executed under Linux system environment.

```
<Condition> system == Linux </Condition>
```

Obligations specifies that operations should be executed before and/or after evaluations, in conjunction with the enforcement of policies. File owners may ask visitors to get their permissions in order to access the files. Obligations may demand users execute operations after the queries are permitted. For instance, data owners might require visitors to report their access. In the following example, if the access is permitted, users have to report their access within 10 days. The report includes visiting time, the purpose of visiting, feedback *etc.*, for statistical and data tracking.

```
<Obligations>
<obligation>
<operation> require each access </operation>
</obligation>
<obligation>
<operation> report each visit </operation>
<report> access time, access purpose, feedback </report>
<temporal constraint> 10 days </temporal constraint>
<fulfill on> access </fulfill on>
</obligation>
</Obligations>
```

Transformation indicates how the provenance graphs should be transformed in order to hide sensitive information. It specifies how each provenance partition should be processed by replacing or removing. The transformation elements consist of four items including provenance partitions, transformation scope, transformation mode, and transformation label.

Firstly, we illustrate how these *provenance partitions* are defined in a provenance access control policy and present some examples.

- Vertices. This collects one or more types of vertices in a provenance DAG. These can be Agent, Process, Artifact, or Attributes. Data owners might allow operations to be read, but keep the performers anonymous. The following examples collect agent vertices and the process vertices with the values of “wasGradedBy" or “Graded".

```
vertices (TypedV_Ag(G_i))
vertices (Typed'_P(G_i, wasGradedBy|Graded))
```

- Provenance Path. This is a single line of vertices in a provenance DAG. Based on the directions of edges that link vertices, we defined two types of provenance paths: directed path, and general path. In directed paths, only cause edges connect vertices from the original to the destination; while in general paths, there can exist effect

edges between the two terminals. The following directed path example is from node “wasGradedBy" or “Graded" to node “wasSubmittedBy" or “Submit".

Particularly, in a directed path, processes from the original to the destination are listed according to time order, because all the nodes are linked by cause edges.

`directed` $(TypedV'_{Ag}(G_i, wasGradedBy\|Graded)//\texttt{b+}$ $TypedV'_{Ag}(G_i, wasSubmittedBy\|Submit))$

- Subgraph. A subgraph is a collection of vertices with a nominated origin and/or destination, which can be expressed as subgraphs ($v_i \setminus v + v_j$). The following first example of subgraph defines all operations performed in 2016 in provenance graph G_i. The other example represents a partition from a vertex with given value to the end of the whole graph, which expresses all the operations happened since 2016 in a graph.

`subgraphs` $(AttV_P(G_i, 1/1/2016)//AttV_P(G_i, 31/12/2016))$
`subgraphs` $(AttV_P(G_i, 1/1/2016)/following :: *))$

Transformation scope defines the scope of nodes for transformation. According to specific security requirements, the conjunctive vertices which may reveal clues should be hidden as well. *Transformation scope* takes three possible values. They are original, conjunction, and extension. Vertices Classification Dictionary was defined in Definition 8.

- Original indicates the vertices defined by access control policies do not extend to other vertices in the given provenance graph.
- Conjunction represents that a cluster of vertices should integrate with the connecting nodes which are in the same category referring to the VCD. To facilitate graph transformation, the VCD lists categories of nodes and the corresponding labels. The labels replace the removed vertices when the graph was transformed. Then, if the neighbour nodes of a cluster belong to the same category, the cluster should extend to include these neighbour vertices.

- Extension is a function to return a set of clusters in the given provenance graph. For a given cluster of vertices, all vertices in the provenance graph are within the same category and should be collected as a set of clusters, regardless of whether these vertices are connected with the given cluster or not.

Transformation mode indicates how to process the vertices that can be accessed or collected by provenance access control policies. The two possible modes are “replace" and “remove".

- Replace. When transforming a new provenance graph, replace clusters of vertices with the label referring to the VCD. The label is a summarised term for a category of the vertex.
- Remove which removes clusters of vertices specified by access control policies and emerges edges besides the removed nodes.

In terms of *transformation label*, the edges connecting the transformed nodes matter.

- Original dependency When a cluster of nodes is removed or replaced, Original dependency means keeping the original dependencies by merging two of the edges beside a removed node referring to the Edge Merging Table.
- Fault dependency indicates to delete original labels and replace them as “wasCausedBy", in order to prevent the labels to reveal clues of removed vertices.

The example *Transformation* item defines two provenance partitions to be transformed. The first one is a *subgraph* beginning with an *Artifact* vertex *O3V1* and ending at *Artifact* vertices *O8V1*. As the transformation scope is “original", the subgraph does not include any other vertices. It should be replaced with a label due to the fact that the transformation mode is “replace". Moreover, all connecting edges for the cluster of nodes should be changed with “wasCausedBy". The second hidden partition is a *Process* vertices Submit|was SubmittedBy. If the neighbour nodes are within the same category defined by VCD, they will be included by a partition, which will be deleted according to the transformation mode. In terms of edges, keep the original edges or merge them by referring to the Edge Merging Table.

<Transformation>
<partition>
subgraphs ($TypedV'_A(G_i, o3v1)//TypedV'_A(G_i, o8v1)$ >)
</partition>
<scope> original </scope>
<mode> replace </mode>
<label> false dependency </label>
<partition>
vertices ($TypedV'_P(G_i, Submit|wasSubmittedBy)$)
</partition>
<scope> conjunction </scope>
<mode> remove </mode>
<label> original dependency </label>
</Transformation>

Effect defines results of a policy. The values of effects are *Absolute Permit*, *Deny and Necessary Permit*, and *Finalising Permit*, which are listed according to the priority from highest to lowest. Lower priority policies comply with higher priority policies. There is a need to set different effects because policies naturally take varying degrees of importance depending on the power of the generator. Moreover, it addresses the issue of conflicting policies.

Partitions in policies with various effects correspond to different operations. Namely, partitions in a *permit* policy are permitted access, while partitions in a *deny* policy should be hidden to visitors.

- Absolute Permit is given the highest priority, namely, it gives the permit result to an applicable policy regardless of the results of other policies. In *Absolute Permit* policies, partitions will be permitted access despite the effect of other policies.

- Deny takes the second priority after *Absolute Permit*. When transforming a provenance graph, the algorithm holds partitions permitted by *Absolute Permit* policies. A policy with *Deny Effect* evaluates the rest of the part of graphs. The policies that define partitions cannot be accessed regardless of effects of lower priority policies.

- Necessary Permit takes the third priority. It works on the part of a graph not affected by *Absolute Permit* and *Deny* policies, and nominates accessible partitions.

- Finalising Permit is the effect with the weakest priority and which will be evaluated last. The motivation of classifying policies into necessary permit and finalising permit is to meet flexibility and convenience for different granularity of levels of administration[2].

3.4.3 The Grammar

In this section, we define a grammar for each tag in our proposed PACLP.

<exp>::= <char>+ (“."<char>+)?

<char>::= [a-z] | [A-Z] | “_" | “−"

<reg>::= “?" | “+" | “*”| <num> | “["<ver>+ | <char>+“]"|“["^ <ver>+ <char>+“]"

| \v | “|"

<bool>::= “AND" | “OR" | “|"

<op>::= “==" | “⩽" | “⩾" | “<" | “>"

<num>::= ([0-9])+

<sp>::= “[" <exp> “]"

<ver>::= <char>+ | <func>+

<ver-ep>::= <ver> | “\v <reg>+

<path kw>::= “directed" | “general" | “subgraph"

<provenance partition>::= <path kw> <ver-ep> (, <vertex-ep>)+

<transformation mode>::= <char>+

<transformation scope>::= <char>+

<dependency label>::= <char>+

Following, we define the set of strings accepted by each element in our language.

name=(<char> | <num>)+

subject= “<subject>"<name>+ “</subject>"

object= “<object>"(“any provenance name"| <name>)+ “</object>"

restriction=“<restriction>"(<exp> <num>)+ (<op> | <sp> <reg>?) (<exp> <num>?)+ (<bool> (<exp> <num>?)+ (<op> | <sp> <reg>?)? (<exp> <num>?)+)∗“</restriction>"

condition= “<condition>"(<exp> <num>)+ (<op> | <sp> <reg>?) (<exp> <num>?)+ (<bool>(<exp> <num>?)+ (<op> | <sp> <reg>?)? (<exp> <num>?)+)∗“</condition>"

effect= “<effect>"<char>+ <num>“</effect>"

target= “<Target>"<subject> <object> <access purpose>? <restriction>? “</Target>"

transformation= “<Transformation>"<provenance partition> <transformation mode> <transformation scope> <dependency label>“</Transformation>"

policy= “<Policy ID>" <target> <condition>? <obligation>? <transformaiton> <effect>“</Policy ID>"

The grammar defined above allows us to evaluate the PACLP for correctness. In addition, it enables a parser to unambiguously translate the policy into a form that can be employed by the appropriate server in our proposed architecture.

3.4.4 Case Study

We give two sample policies generated over the proposed PACLP. In a policy, *condition* and *obligation* are optional items. Therefore, the values of these items can be “none".

In the sample Policy ID=1, it restricts requests from student users who are from the Computer Science or Electrical Engineering departments, and objective graphs must contain the provenance paths that record “Alice did submit". Only if a request meets the target section of the policy, the policy will be selected as an applicable policy for the request. As the *effect*

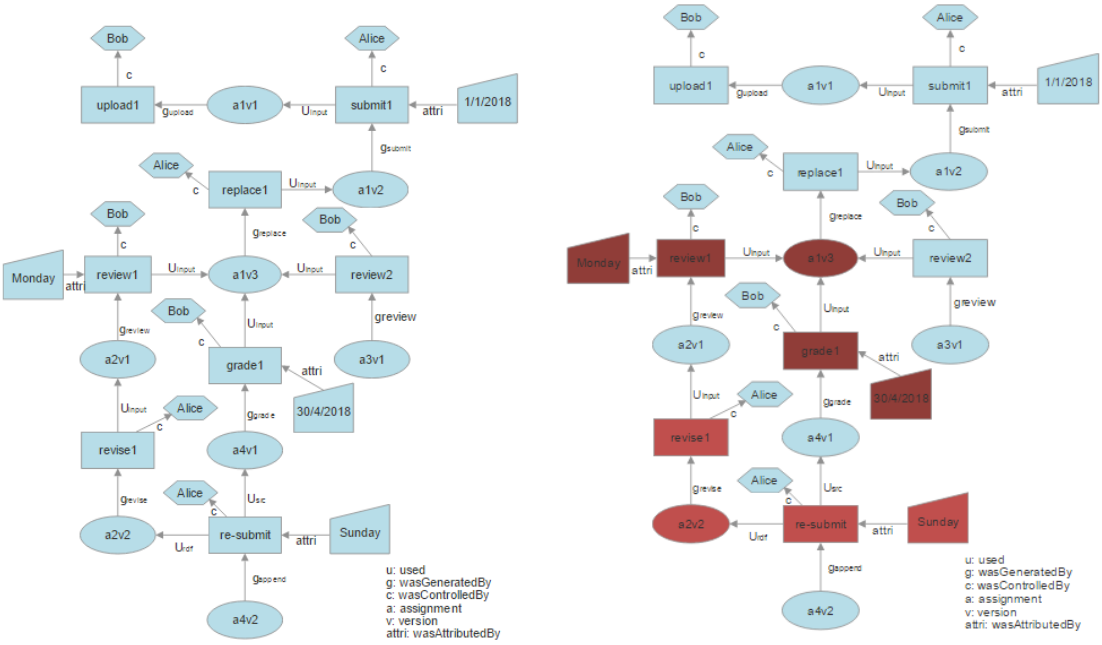

Figure 3.10: Case Study

of the policy is *Deny*, which indicates the nominated partition cannot be accessed unless it was permitted by a policy with *Absolute Permit* effect, because the priority of *Absolute Permit* is higher than *Deny*. Hence, the policy denies access to the nominated subgraphs. The first subgraph starts from a *Process* vertices Review and ends at *Process* vertices Grade. As the attribute values restrict, the “Review" process cannot be taken at Sunday, and matched “Grade" vertex must attach with an attribute value “30/4/2018". The partition should be replaced without a conjunction with any other vertex, as the transformation model is “Original". Similarly, The second partition nominated is that “the owner of the data did modify at and re-submit at 30/5/2018". However, the partition should integrate with the all the connecting node which are in the same category referring to the VCD. Both of the removed partitions should be replaced with labels referring to the VCD. The label is a summarised term for a category of the vertex. In addition, the partition connected edges should be replaced by “wasCausedBy" due to the “false dependencies" of *transformation label*.

<Policy ID=1>
<Target>
<subject> role:student & department: computer science | electrical engineering </subject>
<object> any provenance graph </object>

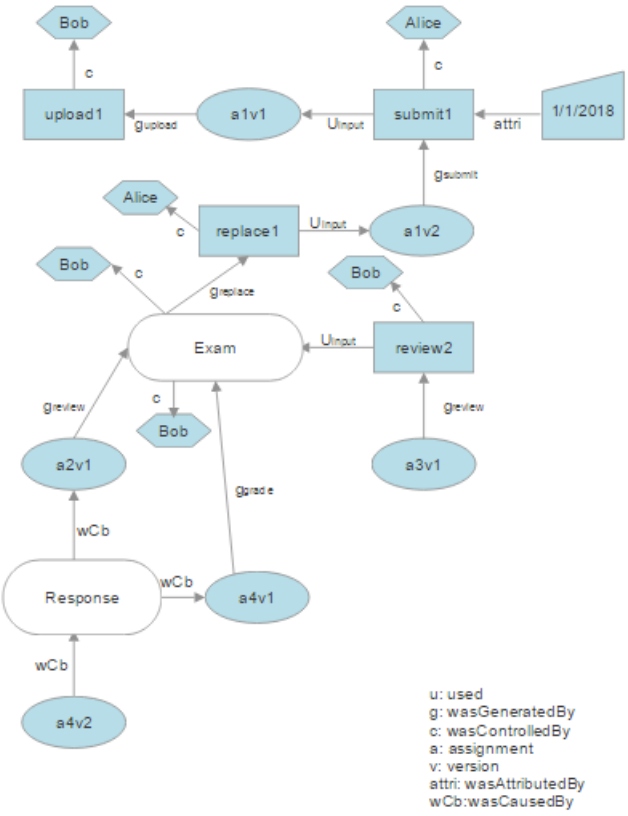

Figure 3.11: Case Study

<restriction> directed ($TypedV'_{Ag}(G_i, Alice)$// $T_P(G_i, wasSubmittedBy|Submit)$)

</restriction>

</Target>

<Condition> system==Linux </Condition>

<Effect> Deny </Effect>

<Obligation> report each access </Obligation>

<Transformation>

<partition>

subgraph ($TypedV'_P(G_i, Review)$[not@day='Sunday']//$TypedV'_P(G_i, Grade)$[@date='30/4/2018'])

</partition>

</scope> original </scope>

</mode> replace </mode>

</label> false dependency </label>

<partition>

subgraph ($TypedV'_{Ag}(G_i, Owner)\,TypedV'_P(G_i, Modify)$//$TypedV'_P(G_i, Re-submit)$[@date='30/5/2018'])

</partition>

</scope> conjunction </scope>
</mode> replace </mode>
</label> false dependency </label>
</Transformation>
</Policy ID=1>

3.5 the Algorithms

We propose Algorithm 1 for the view transformation of provenance graphs according to the results of PACLP. The input of Algorithm 1 is results of access control policies and a targeted provenance graph, and its output is a transformed graph which can be returned to requestors. Based on the default setting of each system, the view transformation can be processed by two approaches, which are *Permit Takes Precedence* and *Deny Takes Precedence*. The former returns allowed partitions generated by access control policies and the latter deletes or replace denied partitions.

Briefly, this algorithm arranges collection of vertices nominated by policies as a cluster. Particularly, the function *Sort* introduced in Algorithm 1 divides vertices into clusters, where the clusters will be deleted or replaced as a union. The vertices in the same category defined by VCD and attached with the same transformation tags can be divided into a cluster for further processes. The terminology $C_i.Tscope$ is the value of transformation scope defined by policies, which is attached to vertices by Algorithm 2. $C_i.Tscope$ nominates whether other vertex belongs to the same class in a given provenance graph should be added into a cluster. The goal to introduce such a mechanism is to extend the effective scope of a policy, in case vertex which might leak sensitive information cannot be thoroughly listed in a policy. Similarly, $C_i.Tmode$ and $C_i.Tlabel$ also tags for each cluster, which is defined by the policies.

Algorithm 1 Graph Transformation

INPUT:

g: OPM graph % the DAG to be transformed

AC: % a set of vertices which can be accessed, defined by policies

DC: % a set of vertices which can not be accessed, defined by policies

restr: conditions of elements of the partition

Ttags: Tmode + Tscope + Tlabel %they are attached with AC&DC, as the output of Algorithm 3

OUTPUT:

g': a transformed graph of g which can be accessed by the requestors.

FUNCTIONS:

Sort: dividing a set of vertices as clusters of vertices according to VCD&Ttags, where each cluster are vertices in the same category of VCD with the same Ttags, to return a set of clusters

CountCV: count the number of vertices connected with CauseEdges for a cluster of vertices

EffectCV: count the number of vertices connected with EffectEdges for a cluster of vertices

RUN ALGORITHM:

CV = ∅ % CauseVertices

EV = ∅ % EffectVertices

$\mathscr{C}$ = ∅

if Permit takes precedence **then**
  $\mathscr{C}$ = Sort(AC)
  for $C_i \in \mathscr{C}$, where i= 1, 2,...n **do**
    if $C_i.Tscope$ = conjunction **then**
      C_i = Con(C_i)
    end if
    if $C_i.Tscope$ = extension **then**
      C_i = Ex(C_i)
    end if
    $\mathscr{C}'$ = V\$\mathscr{C}$
    g′ = Rem_p(g, $\mathscr{C}'$)
  end for
else if Deny takes precedence **then**
  $\mathscr{C}$ = Sort(DC)
  for $C_i \in C$, where i= 1, 2,...n **do**
    if $C_i.Tscope$ = conjunction **then**
      C_i = Con(C_i)
    end if
    if $C_i.Tscope$ = extension **then**
      C_i = Ex(C_i)
    end if
  end for
  for $C_i \in C$, where i= 1, 2,...n **do**
    if ($C_i.Tmode$==‘Remove’)&(CV⩽1) &(EV⩽1) **then**
      $g' = Rem_p(g, C_i)$
      if $C_i.Tlabel$=Original dependency **then**
        merged edges refers to the Edge Merging Table
      else if $C_i.Tlabel$=Fault dependency **then**
        merged edges is “wasCausedBy"
      end if
    else
      $g' = Rep_p(g, C_i)$
      if $C_i.Tlabel$=Fault dependency **then**
        merged edges is “wasCausedBy"
      end if
    end if
  end for
end if
result g'

Following, to evaluate provenance access policies we propose two algorithms which are generated over our proposed language. The goal of Algorithm 2 is to retrieve all the applicable policies in a system for one request. It verifies whether *subject* and *object*, *access purpose* and *restriction* in a query meet *Target* in policies. Algorithm 3 combines applicable policies for a request and generates allowed and forbidden collections of vertices. The algorithm combines policies based on the priorities of policies.

Algorithm 2 Retrieval Applicable Policies

INPUT:

q: a query;

g: OPM graph; %the targeted DAG in a query

POL: all the policies;

OUTPUT:

POL_{ap}: all the applicable policies for a query;

FUNCTIONS:

FunctionRESTRI: verify if the subject and object meet all the restrictions in a policy, return {0, 1}.

RUN ALGORITHM:

$POL^* = \emptyset$;
for all $pol_i \in POL$ **do**
  if g $\subseteq$ object.target.pol_i& subject $\subseteq$ subject.target.pol_i& Access Purpose $\subseteq$ access purpose.target.pol_i; **then**
    if FunctionRESTRI (a, pol_i) = 1; **then**
      $POL_{ap} = POL_{ap} \bigcup pol_i$
    end if
  end if
end for
result=POL_{ap}

When the server receives a query, it runs Algorithm 2 to identify applicable policies for the query. In this language, *Target* which includes *subject*, *object*, *access purpose* and *restriction* confines applicable subject and object. The *condition* element defines context restrictions. However, Algorithm 2 screens applicable policies based on *Target*. *Condition* was verified in Algorithm 3, because the four effect values affect with different binary results of *Condition*. Specifically, the effect value of *Necessary Permit* denies provenance partitions when the *Condition* of this policy is evaluated as a false, while the other three effect values affect when *Condition* is evaluated as true.

Algorithm 3 Merging results of individual policies

INPUT:

g: OPM graph;

POL_{ap}: all the applicable policies;

OUTPUT:

AC % Allowed Collection of vertices

DC % Denied Collection of vertices

FUNCTIONS:

FunctionABSOLUTE: select all the absolute permit policies

FunctionDENY: select all the deny polices

FunctionNECCE: select all the necessary permit policies

FunctionFINALISING: select all the finalising permit policies

RUN ALGORITHM:

g′=∅; AC = ∅; DC = ∅

RemainderDAG = ∅; Obligation = ∅

for all $pol_i \in AbsolutePermitPolicies$ **do**

  if $condition.pol_i$ = 1 **then**

    AC = AC$\bigcup$ $Tpartition.pol_i$ & $Ttags.pol_i$ % Ttags = Tmode + Tscope + Tlabel

    Obligation=Obligation$\bigcup$ $obligation.pol_i$

  end if

end for

RemainderDAG = g / ViewPartition

for all $pol_i \in DenyPolicies$ **do**

  if $condition.pol_i$ = 1 **then**

    if $vertex_m$ ∈Tpartition.$pol.i$ & $vertex_m$∩RemainderDAG≠ ∅; **then**

      DC = DC$\bigcup$ $vertex_m$ & $Ttags.pol_i$

      Obligation = Obligation $\bigcup$obligation.pol_i

    end if

  end if

end for

RemainderDAG = g / DC

for all $pol_i \in NeccessaryPermitPolicies$ **do**

  if $condition.pol_i$ = 0 **then**

    if $vertex_m$ ∈Tpartition.$pol.i$ & $vertex_m$∩RemainderDAG≠ ∅; **then**

      DC = DC$\bigcup$ $vertex_m$ & $Ttags.pol_i$

      Obligation=Obligation $\bigcup$ $obligation.pol_i$

    end if

  end if

end for

RemainderDAG = g / DC

for all $pol_i \in FinalisingPermitPolicies$ **do**

  if $condition.pol_i$ = 1 **then**

    if $vertex_m$ ∈Tpartition.$pol.i$ & $vertex_m$∩RemainderDAG≠ ∅; **then**

      DC = DC$\bigcup$ $vertex_m$ & $Ttags.pol_i$

    end if

    Obligation=Obligation $\bigcup$ $obligation.pol_i$

  end if

end for

result={AC, DC, Obligation}

Algorithm 3 combines results of applicable policies and returns a collection of allowed and forbidden vertices, because the effect of policies can have different priorities, such as *Absolute Permit* > *Deny* > *Necessary Permit* > *Finalising Permit*. Where the algorithm runs from the policies with the *Absolute Permit* effect, the partitions allowed to access will be definitely accessed regardless of results of other policies. Policies with denying effects only have an effect on the remainder of the DAG. The partitions hidden by the “denying" policies cannot be accessed by ignoring the rest of the applicable policies. Then, “necessary permit" policies and “finalising permit" policies supplement accessible partitions. The output of Algorithm 3 is a set of permitted vertices and a set of denied vertices that are attached with transformation tags. The transformation tags indicate the transformation scopes of the vertices, how to hide prohibited vertices, and how to deal with the connected edges. Particularly, Algorithm 1 utilises the results of Algorithm 3 to transform the view of provenance graphs, in order to answer the request.

3.6 Evaluation

In order to evaluate the performance of our proposed PACLP, we design and implement an experiment to implement policy generation and individual policy results combination. We deploy the implemented prototype onto a virtual machine with 16GB memory and 3.40 GHz CPU. The 20 sample provenance graphs and 200 policy conditions and tags were generated.

Firstly, we generate 40 random PACLP policies under our proposed policy models and 10 provenance-based access control policies under the language (we name it LPAC for short) proposed in paper [42], in order to compare the time span for generating these policies. Among the 40 random PACLP policies, we generate 10 policies over the four policy effects including *Absolute Permit*, *Deny*, *Necessary Permit* and *Finalising Permit*. We run the experiments ten times and record the average time of the results, shown in the figure below. However, we could find from results that generating PACLP takes more time than LPAC proposes in paper [42], because PACLP has more policies items than LPAC.

Following, we select 300 random subgraphs/partitions/provenance paths from the sample provenance graphs. We measure the numbers of subgraphs/partitions/provenance paths can be expressed by PALCP and LPAC, under the model of OPM$^+$. We draw figures of the

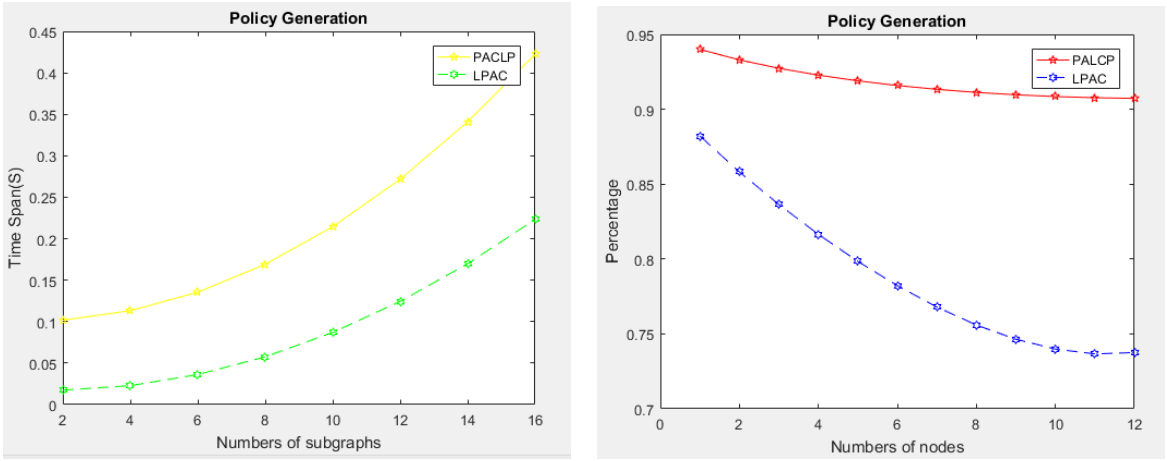

Figure 3.12: Experiment

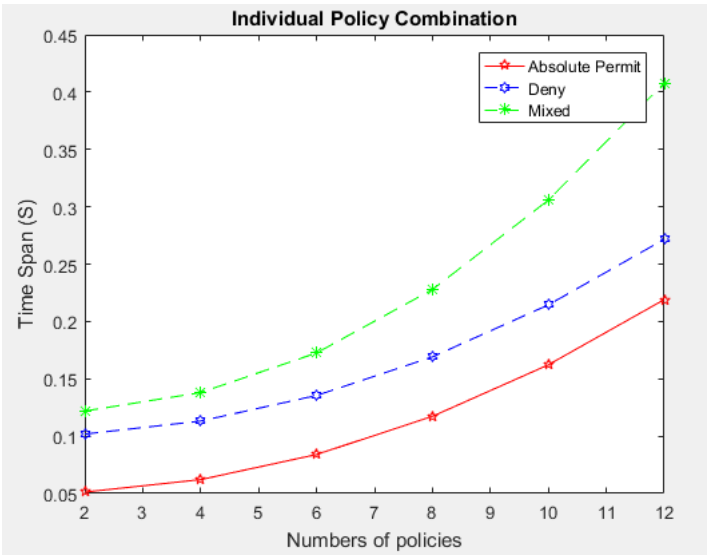

Figure 3.13: policy results combination

average values when we ran the experiments 10 times. Obviously, PACLP can express more random sample provenance partitions comparing with the other policy language model under the provenance model which stores entity attributes. The reason is that we introduce more language elements to express different types of provenance partitions. We especially focus on the ability of PACLP to nominate vertices based on the attached attribute values.

At last, we implement the processes to merge the results of individual policies. The selected three scenarios are (1) all 20 sample policies are those with the effect of *Absolute Permit*, (2) all 20 sample policies with the effect of *Deny*, (3) 20 sample policies with all the effect randomly. As above experiments, we ran the system 10 times. And draw the average time spans to the figure below. Apparently, merging policies with all effect of *Absolute Permit*

takes the shortest time. Because to run the algorithms for merging those policies calls the least functions. On the contrary, it has to call most functions to merge policies with all the types of effects. Hence, the experiment takes the longest time among the three experiments. Apparently, the experiments demonstrate the process that PALCP can be applied in a project. In another word, we can generate access control policies under this model and transfer a provenance graph based on policies.

3.7 Conclusion

In this chapter, we propose a fine-grained provenance access control language PACLP under OPM$^+$ storing attribute sets to extend the existing languages. In PACLP various types of partitions are defined over XPath. Our provenance access control language aims to define which partial graph can be accessed or cannot be accessed under conditions and restrictions. This fine-grained access control policy model not only hides all the sensitive vertices and edges in a provenance graph but also maximises eligible information to be accessed.

We also present three algorithms in this framework. The algorithms enable retrieval of all applicable access control policies for a given request and merge individual policy results as a final decision on the request. Moreover, this chapter also present algorithms to transform provenance graphs, in order to generate a new provenance graph with eligible information for access requestors. At last, we did experiments to evaluate the performance of PACLP.

4

A fine-grained Policy Model for Provenance-based Access Control

A fine-grained provenance-based access control policy model is proposed in this chapter. This method employs provenance as conditions to determine whether a piece of data can be accessed because historical operations performed on data could reveal clues about its sensitivity and vulnerability. Particularly, our proposed work provides a four-valued decision set which allows showing status to match a restriction particularly. This framework consists of target policy, access control policy, and policy algebras.

Different from the policy we proposed in Chapter 3, provenance subgraphs/partitions are mainly used as policy conditions in this framework. While provenance partitions mainly appear as policy results in PACLP. The goals of the two frameworks are also different, which will be explained in the following section.

4.1 Introduction

4.1.1 Related Work and Motivations

In recent years, provenance access control research includes two main classes. These are *provenance access control* which determines access right to provenance itself and *provenance-based access control* which evaluates access rights to data based on their provenance. We proposed a provenance access control policy language in Chapter 3 and present a provenance-based access control policy model in this chapter.

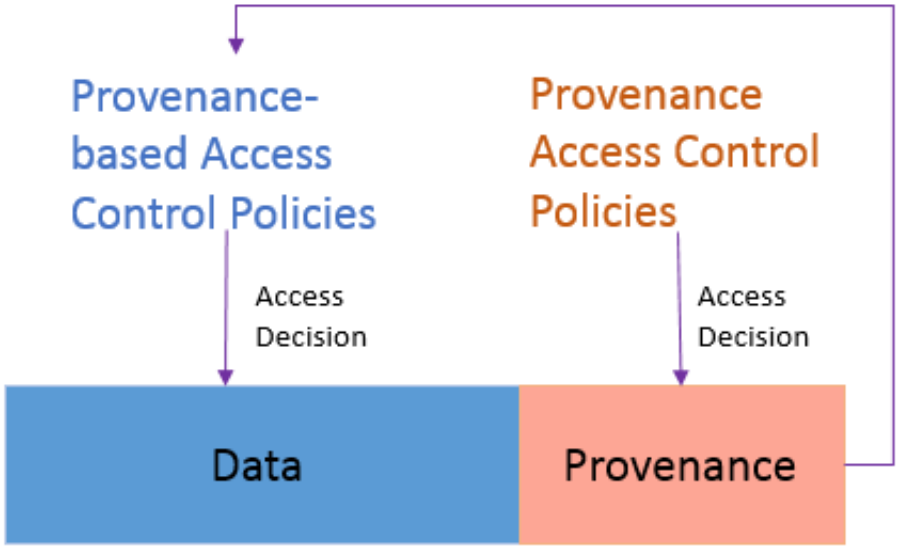

Figure 4.1: Provenance Access Control Policies v.s. Provenance-based Access Control Policies

Provenance-based access control involves using provenance of data to evaluate the accessibility of that data, where provenance records origins of data and operations performed on them. Historical operations executed on documents could be crucial conditions for determining who can access them and which operations are permissible for execution. The mechanism also benefits fine-grained access control as provenance contains a large number of attributes. To be more specific, let sample provenance-based access control policies be: *(1) an assignment can be submitted if it has never been submitted before; (2) a file can only be reviewed by its authors after the file has been submitted and graded.* It is easy to find from the examples that whether a piece of data can be accessed is determined by operations recorded in its provenance. Provenance proposes significant restrictions for fine-grained access control.

Several provenance-based access control policy models [14][49] and history-based access control [57] have been proposed. Park *et al.*[49] proposed a family of provenance-based access control models which utilised a notion of dependency as the key foundation for access control policy specification. They proposed a basic provenance-based access control model $PBAC_B$, which facilitates additional capabilities beyond those available in traditional access control models by introducing provenance as access control conditions. Based on the basic model $PBAC_B$, a family of PBAC models was defined by extending three criteria, which are (1) the kind of provenance data in the system, (2) whether policies are based on acting user dependencies and object dependencies, and (3) whether the policies are readily available or need to be retrieved. The three models $PBAC_U$, $PBAC_A$ and $PBAC_{PR}$ extend one of these three criteria respectively. However, combined results for individual policies are only basic conjunctive or disjunctive connectives between rules.

There are still issues have not been addressed by previous provenance-based access control policies. Firstly, the existing work did not achieve a proper fine-grained policy model. Partial matching of a provenance graph has not been explored well enough. Partial matching a provenance graph is a scenario that not all elements of a provenance subgraph can be found in the given provenance graph. For example, let a policy condition be: *a doctor diagnosed a patient at 22/4/2018*. The partial matching scenario could be that a doctor diagnosed a patient, but the timestamp is not 22/4/2018 or missing. Proposing a mechanism to introduce partial matching a provenance graph is meaningful, and values to show the partial matching status should be presented in a fine-grained policy model. From this point, policies algebras to merge those partial matching values are also missing from the existing work.

Therefore, in this chapter, we propose a framework to bridge the gap. First of all, this framework is based on OPM$^+$ which we proposed in Chapter 3. The access control policies are tailored to aware the situations of partial matching provenance graphs. In addition, corresponding policy algebras to merge atomic policy values also are proposed.

In order to state the benefits of our proposed approach. Let a sample policy be: an assignment was submitted before March 2017, and marked before April 2018. If a provenance graph contains the same partition, we say it as 100% match. On the contrary, if a provenance graph does not contain any elements of the partition, we say it as 0% match. However, if the assignment was submitted and graded, but the processes did not complete before March 2018 and April 2018 respectively. We say it partial match, where the status is between 0%

match and 100% match.

Comparing with most existing work [14][57][49], we did not select normal two-valued decision set (yes or no), but introducing a technique to show the status of the partial match. We believe the idea is more close to the nature of provenance-based access control. Because provenance is a type of metadata consists of various elements, including entities, attributes, and dependencies. We distinguish these elements and define “a partial matching" when given provenance only matches a part of elements.

4.1.2 Our Contribution

In this chapter, we propose a sound fine-grained provenance-based access control framework that consists of policy models, policy evaluation, and policy algebras. Target section confines the effective range of a policy. Namely, target section specifies which policy can match a given request. While the access control section evaluates applicable queries and outputs a result.

Different from the approach we proposed in Chapter 3 where provenance subgraphs are returned as results of policies, provenance subgraphs are mainly employed as conditions of policies in this framework. Particularly, the target section consists of some atomic targets. We introduce provenance partitions as atomic targets. The results of an atomic target are evaluated based on whether the provenance DAG contains the partition. In addition, a four-valued decision set is defined for each atomic policy. Namely, the decision does not simply restrict to *Yes* or *No*, but allows for a medium status between them. Moreover, we organise atomic policies under a tree structure and defines a sound collection of new operators to merge results of four-valued result sets.

We propose three types of atomic targets. Path atomic target only takes attributes from provenance. The determination is based on whether certain operations that are recorded in provenance have been performed on the data. Associated atomic target takes attributes from both provenance and requests (subject, object, *etc.*), which implies that the determination is based on whether the requestors have performed operations on the data. The latter atomic targets perform differently from traditional access control. For instance, an associated atomic target could express such a scenario: if the requestor was editing a file before and have not submitted it, the access is permitted. Notably, for this scenario, the “requestor" is the subject

of a query, and the editing and submitting are processes from a provenance. At last, the attributes from requests can also be defined as conditions, which is named as request atomic target.

Moreover, in this framework, these atomic targets (conditions) are organised under a tree structure. This model takes a policy tree structure which combines atomic policies. We borrow a mechanism of policy tree structure from Crampton *et al.*'s paper [19]. By borrowing the structure, we replace the attribute triple as a provenance partition. This can express a series of historical operations performed on data. The defined provenance partitions can express both logic time and real-time operation strings. We define a "four-valued" decision set for each atomic target (conditions). In addition, we rearrange logic operators and create several new operators to combine atomic policies.

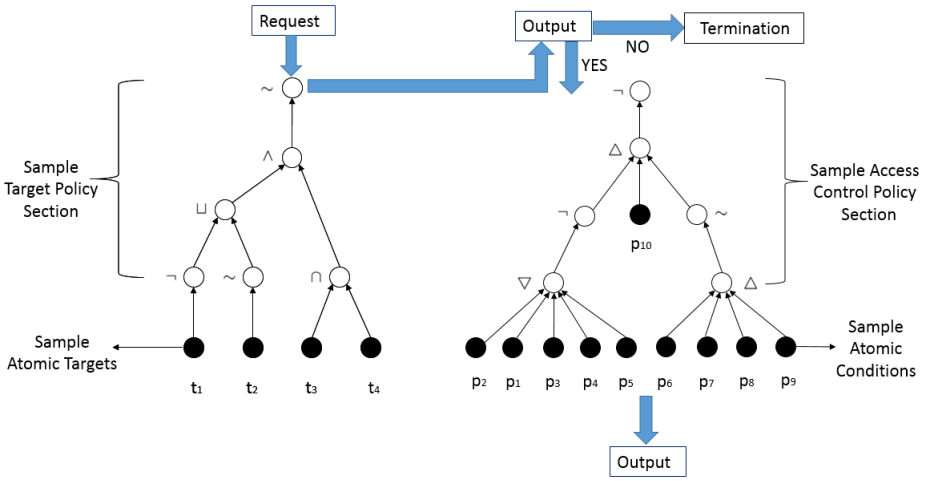

Figure 4.2: The framework of Proposed Provenance-based Access Control

4.1.3 Chapter Organisation

The chapter was organised as follows:

- Section 4.2 illustrates system assumptions of provenance-based access control policies;
- Section 4.3 defines atomic targets and demonstrates how to evaluate atomic targets are presented;
- Section 4.4 focuses on the access control section; and the case study was provided in section 4.5;

- In Section 4.6, we provide approaches to preserve the integrity of provenance;
- Section 4.7 is the evaluation of this framework and 4.8 concludes this chapter.

4.2 the System Assumption

To establish the framework of provenance-based access control model and policy algebras, we propose a system assumption shown in Figure 4.1, where the system consists of four parties including *Administrator*, *Server*, *Data Producers Data Visitors* and *Data Storage*. Following, we illustrate each party.

- *Data Visitors* send queries to the *Database* in order to access files stored in the database.
- *Administrator* generates provenance-based access control policies and sends them to the *Server*.
- *Data Producers* generate files and attached provenance which are sent to *Data Storage*. They might also produce self-defined access control policies and send them to the *Server*.
- *Server* collects policies from administrators and (optional) data producers. The server evaluates results based on the policies and delivers the results to databases.
- *Data Storage* stores files and provenance, which receives queries from users and responds to queries according to the results generated by the *Server*.

In a provenance-aware system, access control policies can be generated by system administrators as well as data owners. When a data visitor sends a request to a server in order to access a piece of data, the server needs to retrieve the provenance of targeted data, as the policies are tailored to make access decisions based on data provenance. After receiving results made by the server, the database implements the results by granting or denying users' requests.

It can be noticed that most access control mechanisms take evaluations based attributes from queries or current system conditions. While for provenance-based access control policies, conditions and restrictions can be extended to provenance which logs historical operations of data.

4.3 Target Policies

The framework of provenance-based access control consists of access control policies and policy algebras, where each policy consists of target section and access control section. Each provenance-based access control policy consists of a target section and an access control section. The *target* section confines applicable requests of each policy. Only if the objective provenance graphs in a request meet the conditions of *Target*, can the policy affect the request. Further, the access control section evaluates accessibility for the request. We give definitions of syntax for targets.

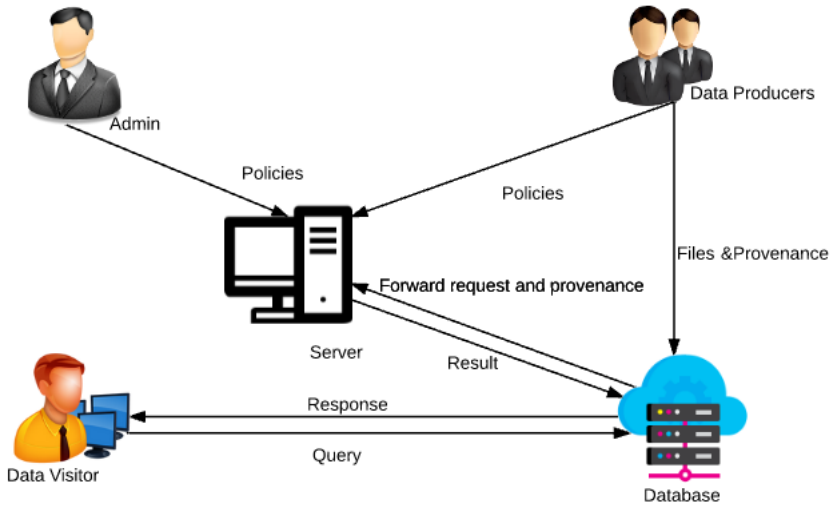

Figure 4.3: System Assumption

In this framework of provenance-based access control model, we define three types of atomic target. They are *path atomic target*, *request atomic target*, *associated atomic target*, which are illustrated concretely in this section.

The definition of the target is different from the “target" item we proposed in Chapter 3. However, the function or idea is similar. Both the “target" in the policy model is a part of the policy, which confines the effective scope of a policy.

4.3.1 Atomic Target

Definition 4 (Path Atomic Target). We define four types of atomic targets from provenance graphs:

-$null_T$ is a target;
-(v_{type}, v_{name}) is a target, where v_{type} is a vertices type and v_{name} is a vertices name;
-$(v_{type}, v_{name}, a, f)$ is a target, where v is a vertices, a is an attribute value and f is a binary predicate.
-a target is a string of (v_{type}, v_{name}) or $(v_{type}, v_{name}, a, f)$ expressed over XPath.
-$(v_{type}, v_{name}$, x, f) is an alternative type of atomic target, where (v_{type}, v_{name}) is a vertices, x is an attribute in a query, and f is a binary predicate.

First of all, the atomic target models we propose above are named as *path atomic target*, as they extract elements from provenance. A signal vertex can be viewed as an extreme form of a provenance path. An atomic target could be one or a string of quaternions, which illustrates a dependency path. v_{type} is the type of a vertex, which is one element in the set of $V^* = \{Ag \cup A \cup P\}$.

v_{name} is an element in a finite set of abstracted names for vertices, where an abstracted name expresses a class of vertices names. For instance, *User* is the abstracted name for all the users including $user_1$, $user_2$ *etc.* In terms of the motivation to utilise abstract value names in policies, provenance is captured after provenance-based access control policies are defined, and provenance keeps developing along with growing files. Hence, the exact names of vertices could be countless and unpredictable. Therefore, it is very difficult for policy generators to predict specific names for all the vertices, and we utilise abstracted names of vertices in provenance-aware policies. In provenance-aware systems, provenance graphs can be interpreted as Typed Provenance Model (TPM) [3] for policy checking purposes. This bridges the gap between policies and provenance graphs. In TPM, the names of vertices are interpreted as the abstract names or names for a class of vertices.

a is an attribute for vertices. The extended provenance model OPM^+ stores context information as attribute nodes in a provenance DAG. Under this model, provenance can be classified as base provenance data and (optional) attribute provenance data that is associated with main entities in the graph (ag, p, or a). Attribute provenance data was classified into three categories: Agent-related Attributes, Process-related Attributes, and Artifact-related Attributes.

In the quaternion Qua = $(v_{type}, v_{name}, a, f)$, f is a binary predictor. There are five possible values of a binary predictor, they are =, $\leqslant$, <, $\geqslant$, >. Moreover, as equality (=) is adopted most frequently, we assume equality is the default value and can be omitted from the atomic targets.

We propose an example of a quaternion for a vertices, let a quaternion be (*Agent*, *User*, *Female*, =). In the example, *Agent* is the vertices type; *User* is the abstracted vertice's name in a graph; *Female* is an attribute for the vertices; and = is the binary predicate. Namely, a user vertices of type *Agent* with female attribute matches this atomic target. For another example, (*Process, Submit/wasSubmittedBy*, *1/1/2016*, <) is a quaternion, and a process named “Submit" or “wasSubmittedBy" executed before *1/1/2016* matches it.

A dependency path could be a single vertex in a provenance graph. A dependency path can also be a string of connected vertices, which expresses a set of operations executed on a piece of data. Whether a provenance consists of given dependency paths can be defined as a condition in an access control policy, as provenance-based access control determines if a piece of data is accessible based on if there have been certain operations performed on the data. We propose a few simple samples to illustrate what provenance-based access control policies can do as following:

(1) *An exam paper can only be downloaded if it has been reviewed at least three times but has not been graded;*

(2) *A paper can only be accessed if it has not been submitted;*

(3) *A file can only be accessed if it was generated originally before 2016 and was reviewed after 2017;*

(4) *If a piece of data was revised by Alice in 2016, and never been edited by Bob, then it can be accessed by students;*

The examples above are sample provenance-based access control policies, which make access decisions based on certain operations recorded in provenance. One or a series of

operations can be defined as a provenance path in policies.

Particularly, in certain scenarios, a dependency path defined in advance may only list a few vertices in a string instead of all the vertices, such as nominating the starting point and ending point of a provenance path. Thus, it can address the difficulty that all vertices in a provenance path are hardly predicted in advance. For example, let a dependency path be (v_i, \v+, v_j, \v{2}, v_k). It presents a path starting at vertices v_i, followed by several other vertices regardless of the number of them and of vertices v_j. The path ends at v_k, where there are two vertices between v_j and v_k.

Directed D_{PATH} are those connected by all types with the *CT*, where the processes in a path are presented according to a timed sequence. Hence, via identifying the directions of labels connecting vertices, it ensures that operations were executed according to time order in a directed D_{PATH}.

Based on the above definitions of items, these elements of our proposed provenance-based access control can express D_{PATH} in both logic time and real time. The concept of both is defined in Ravari's paper [104]. The difference of timestamp for them is that a logic counter-based clock ticks every time an event occurs and the other one records absolute real-time. Namely, logic time compares the order of execution for operations but not the actually executed time. Below, we illustrate how to express D_{PATH} based on logic time and real time.

Scenario 1 (logic time-based paths) A D_{PATH} records that an assignment was submitted then was graded. It allows operations between them such as replacement and modification, regardless of the numbers of operations. Following, it was reviewed by a coordinator. The D_{PATH} is expressed as:

<directional D_{PATH} >::=

<(Process, wasSubmittedBy|Submit) (\v+)

(Process, wasSubmittedBy|Grade)

(Process, Append)>

In this example shown in Figure 4.3, the provenance path is expressed by some vertices over XPath, where each vertices is defined by the form of (v_{type}, v_{name}) or $(v_{type}, v_{name}, a, f)$. ^ and $ (which are starting and ending symbols of XPath) are omitted, as no ambiguity arose. It is a directed path at which vertices are all connected by *CT*. Hence, via distinguishing directions of dependency types, the system can make sure the operations in the string happened according to the time order. The D_{PATH} starts with an artifact was submitted, followed by $\backslash v+$ which indicates by several vertices regardless of the number. After that, the assignment was graded and appended.

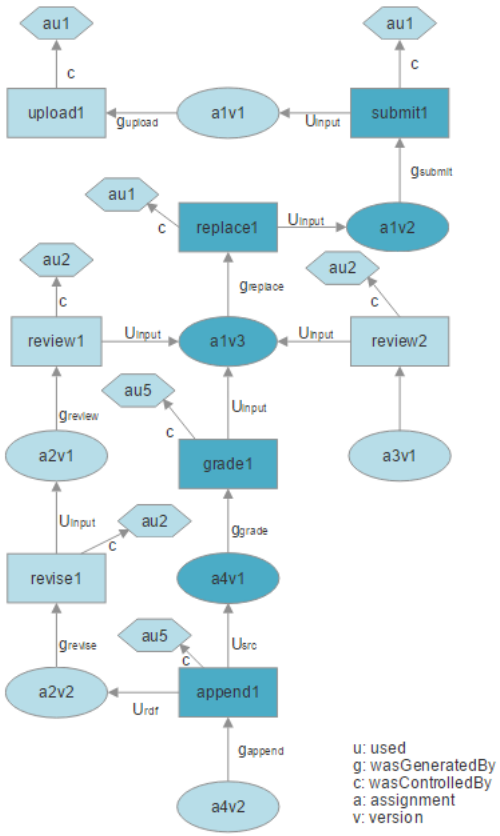

Figure 4.4: Sample Dependency Path A

Scenario 2 (Real Time-based Paths) A D_{PATH} records that an assignment was submitted before 1/12/2016 and was graded at 3/12/2016. It allows operations between them such as replacement and modification, regardless of the numbers of operations. It was subsequently reviewed by a coordinator at 5/12/2016. The D_{PATH} is expressed as following.

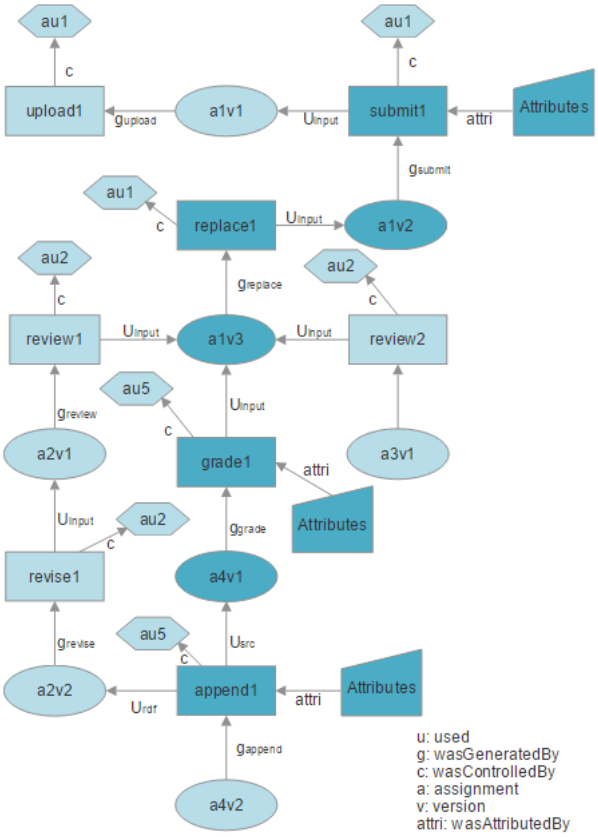

Figure 4.5: Sample Dependency Path B

< D_{PATH} >::=

<(Process, Submit, 1/12/2016, ⩽) (\v+)

(Process, wasGradedBy|Grade, 3/12/2016, =)

(Process, Append, 5/12/2016, =)>

Different from the previous example for a logic time D_{PATH}, the real-time D_{PATH} defines the time order of operations via their attributes (timestamps). \$v$+ indicates that there may exist vertices between the operations submission and grading, where \$v$ is a space and + is a symbol for once or more repeating the first symbol. Particularly, in this example shown in Figure 4.4, as it does not restrict the dependency types in the paths, there might include *ET*. Hence, it only ensures nominated vertices happened in the range of a defined timeframe, but not denote that all the operations in the path have to be performed according to time order.

We propose the last atomic target which consists of attributes from both provenance

graphs and access requests. Access requests are from users, and the provenance is stored in the cloud with its connecting data. Following, we should a sample to illustrate that what the atomic target can express:

(1) *If a requestor is a user who submitted the assignment, it can be accessed by the requestor;*

Apparently, the requestor is a subject of the request and the operation of submission is recorded in the provenance.

x is an attribute in a query, which could be action, subject, or object. Following the previous example, we can define an atomic target policy as (*Agent, submitter, requestor,* =). The v_{type} is *agent* which is the type of a vertex, while the name of the vertex is *submitter*. “requestor" represents the name or ID of the users sending a query, and the binary predictor is “=".

From the model of atomic targets, it can be concluded that the provenance-based access control we proposed control accessibility for data not only based on the provenance but also on the connection of provenance and requests.

At last, attributes in requests can also be defined as restrictions for a atomic target. For example, $object_1$, $action_1$ and $subject_1$ can be atomic targets. Traditional access control policies usually employ *object*, *subject*, and *action* in a query as conditions. It defines when *object*, *subject*, and *action* are within a range, the given requests can (or can not) be accessed. By defining the *request atomic target*, we set an interface between our proposed provenance-based access control model and traditional access control.

Moreover, the attributes in requests also include other data labels, such as *categories* of a file, *usages* and *etc.* However, every request *atomic target* takes a form as a single value.

4.3.2 Atomic Target Evaluation

In terms of evaluation for an atomic target with respect to a request, an atomic target matches the query if the provenance graph meets all the attributes in the atomic target.

Otherwise, it does not match the atomic target.

We begin with the *request atomic target*, as it takes the form as a single value. When the attribute in a query meets the value of attributes, the result is 1_T, otherwise, the result is $\times_T$.

Because the models of *path atomic target* and *joint atomic target* takes the same form. We illustrate the evaluation principles for them together. We hope there exist values of a result to represent the conditions between a full match or complete not match. Hence, we present a four-valued result set $Dec_T = \{1_T, 0_T, \perp_T, \times_T\}$ for the evaluation, where 1_T is absolute match and $\times_T$ represents completely not match. The rank of them is $1_T > 0_T > \perp_T > \times_T$.

(1) The *"null"* target matches all requests, which implies it returns 1_T for all objective provenance graphs.

(2) The target (v_{type}, v_{name}) matches if the provenance graph contains vertices with the vertex type and name, which returns the value of 1_T. However, it returns $\perp_T$ when v_{name} is not match and returns $\times_T$ when both of do not match.

(3) When the quaternion (v_{type}, v_{name}, a, f) or (v_{type}, v_{name}, x, f) is met, it returns 1_T; if it only matches v_{type} but none of the other three values, it returns 0_T; if the provenance graph only meets (v_{type}, v_{name}) but not the value, it returns $\perp_T$; $\times_T$ denotes that it does not include the vertex v in the graphs.

(4) In terms of D_{path}, which is a string of nodes defined by the quaternion of (v_{type}, v_{name}, a, f) or (v_{type}, v_{name}, x, f), it returns 1_T when it meets all quaternion; it returns 0_T, if it meets all the v_{type}, v_{name} in the string but not the other two values; it returns $\perp_T$ if it only meets all v_{type}; it returns $\times_T$ when the provenance DAG can not meet all the v_{type} in the string.

We denote the evaluation of a given request q as $[\![t]\!]_T(\mathrm{q}) \in Dec_T$. We utilise the subscript T to denote that it is the evaluation of targets, in order to distinguish the results for target section and access control section. However, the subscript can be emitted when no ambiguity can arise. Initially, queries for *null* target returns 1_T, and an evaluation for a query with empty provenance graph returns $\times_T$.

$$[\![\ \mathrm{null}\ ]\!](\mathrm{q})=1_T;\ [\![\mathrm{n}]\!](\varnothing)=\times_T$$

Via the definition of atomic targets and evaluation principles, we propose a more a fine-grained provenance-based access control policy model. In this model, different types of attributes extracted from a provenance graph are distinguished. Namely, the four values in the decision set denote the cases that a provenance graph meets different classes of elements in an atomic target.

4.3.3 Operators

A target section may consist of several atomic targets, which are organised under a tree structure. In the tree structure, the leaf nodes are atomic targets, and non-leaf nodes are logic operators to merge the results of each atomic targets, in order to output one final decision.

The operators are tailored for our proposed four-valued decisions set, which includes binary operators, unary operators, and u-ary operators. We introduce binary and unary operators [19] which correspond to the weak and strong Kleene operators [105] for the four values target decision set $\{1_T, 0_T, \perp_T, \times_T\}$ and define several new operators.

We define operators **not** t, **opt** t, t_1 **and** t_2 and t_1 **or** t_2 of targets as follows:

$$[\![\textbf{not } t]\!](q) = \neg\, [\![t]\!](q);\ [\![t_1 \textbf{ and } t_2]\!](q) = [\![t_1]\!](q) \sqcap [\![t_2]\!](q)$$
$$[\![\textbf{opt } t]\!](q) = \sim [\![t]\!](q);\ [\![t_1 \textbf{ or } t_2]\!](q) = [\![t_1]\!](q) \sqcup [\![t_2]\!](q)$$

Here, **not** t and **opt** t are unary operators. The total order of Dec_T is that $1_T > 0_T > \perp_T > \times_T$. In binary operator **and**, the result is the lower value between the two sides. To the contrary, **or** outputs the higher values from the two inputs. However, if we change the Dec_T order as $\times_T > \perp_T > 0_T > 1_T$, **and** (in $1_T > 0_T > \perp_T > \times_T$)= **or** (in $\times_T > \perp_T > 0_T > 1_T$).

X	¬X	∼X	⋆X
1_T	0_T	1_T	$\perp_T$
0_T	1_T	$\perp_T$	$\times_T$
$\perp_T$	$\perp_T$	0_T	1_T
$\times_T$	$\times_T$	$\times_T$	0_T

Table 4.1: Unary Operators

⊔	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	1_T	1_T	1_T
0_T	1_T	0_T	0_T	0_T
$\perp_T$	1_T	0_T	$\perp_T$	$\perp_T$
$\times_T$	1_T	0_T	$\perp_T$	$\times_T$

Table 4.2: Binary Operator ⊔

Hence, when we change the priority order for the values in the Dec_T set, we can enumerate all the operators as **and** for the orders $1_T > 0_T > \times_T > \perp_T >$ ($\cup$), $1_T > \perp_T > 0_T > \times_T$ ($\cap$), $1_T > \times_T > \perp_T > 0$ ($\sqsubset$), $0_T > 1_T > \perp_T > \times_T$($\wedge$), $0_T > \times_T > \perp_T > 1_T$($\vee$), $\times_T > 1_T > 0_T > \perp_T$($\supset$), $\times_T > \perp_T > 0_T > 1_T$($\subset$), $\perp_T > \times_T > 0_T > 1_T$($\vdash$) and $\times_T > 1_T > 0_T > \perp_T$($\dashv$).

$\sqcap$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	0_T	$\perp_T$	$\times_T$
0_T	0_T	0_T	$\perp_T$	$\times_T$
$\perp_T$	$\perp_T$	$\perp_T$	$\perp_T$	$\times_T$
$\times_T$	$\times_T$	$\times_T$	$\times_T$	$\times_T$

Table 4.3: Binary Operator $\sqcap$

$\cup$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	0_T	$\perp_T$	$\times_T$
0_T	0_T	0_T	$\perp_T$	$\times_T$
$\perp_T$	$\perp_T$	$\perp_T$	$\perp_T$	$\perp_T$
$\times_T$	$\times_T$	$\times_T$	$\perp_T$	$\times_T$

Table 4.4: Binary Operator $\cup$

$\cap$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	0_T	$\perp_T$	$\times_T$
0_T	0_T	0_T	0_T	$\times_T$
$\perp_T$	$\perp_T$	0_T	$\perp_T$	$\times_T$
$\times_T$	$\times_T$	$\times_T$	$\times_T$	$\times_T$

Table 4.5: Binary Operator $\cap$

$\sqsubset$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	0_T	$\times_T$	$\perp_T$
0_T	0_T	0_T	0_T	0_T
$\perp_T$	$\perp_T$	0_T	$\perp_T$	$\perp_T$
$\times_T$	$\times_T$	0_T	$\perp_T$	$\times_T$

Table 4.6: Binary Operator $\sqsubset$

$\wedge$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	1_T	$\perp_T$	$\times_T$
0_T	1_T	0_T	$\perp_T$	$\times_T$
$\perp_T$	$\perp_T$	$\perp_T$	$\perp_T$	$\times_T$
$\times_T$	$\times_T$	$\times_T$	$\times_T$	$\times_T$

Table 4.7: Binary Operator $\wedge$

$\vee$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	1_T	1_T	1_T
0_T	1_T	0_T	$\perp_T$	$\times_T$
$\perp_T$	1_T	$\perp_T$	$\perp_T$	$\times_T$
$\times_T$	1_T	$\times_T$	$\times_T$	$\times_T$

Table 4.8: Binary Operator $\vee$

$\supset$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	0_T	$\perp_T$	1_T
0_T	0_T	0_T	$\perp_T$	0_T
$\perp_T$	$\perp_T$	$\perp_T$	$\perp_T$	$\perp_T$
$\times_T$	1_T	0_T	$\perp_T$	$\times_T$

Table 4.9: Binary Operator $\supset$

$\subset$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	1_T	1_T	1_T
0_T	1_T	0_T	0_T	0_T
$\perp_T$	1_T	0_T	$\perp_T$	$\perp_T$
$\times_T$	1_T	0_T	$\perp_T$	$\times_T$

Table 4.10: Binary Operator $\subset$

$\vdash$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	1_T	1_T	1_T
0_T	1_T	0_T	0_T	0_T
$\perp_T$	1_T	0_T	$\perp_T$	$\times_T$
$\times_T$	1_T	0_T	$\times_T$	$\times_T$

Table 4.11: Binary Operator $\vdash$

$\dashv$	1_T	0_T	$\perp_T$	$\times_T$
1_T	1_T	0_T	$\perp_T$	$\times_T$
0_T	0_T	0_T	0_T	$\times_T$
$\perp_T$	$\perp_T$	0_T	$\perp_T$	$\times_T$
$\times_T$	$\times_T$	$\times_T$	$\times_T$	$\times_T$

Table 4.12: Binary Operator $\dashv$

Moreover, the target can consist of a collection of atomic targets $\{t_1, t_2...t_n,\}$ which are combined via these operators. For a target expression, binary operators take the same priorities, and it runs from left to right for the same priority operators. Binary operators take priority over an unary operator, but operators in brackets should be calculated first.

$$\neg (t_1 \wedge t_2 \cap t_3 \cup t_4)$$

$$\sim (\neg t_1 \sqcup \sim t_2 \wedge (t_3 \cap t_4))$$

In the first expression above, the calculation should start from the operators in brackets. If there are no brackets, it should begin with the unary operator $\neg$. As the operators inside of the brackets are all binary, they are calculated from left to right, then, calculating $\neg$ is last. In the second expression, it is obvious that the sub-expression inside of the first bracket should be calculated first. $\neg t_1$, $\sim t_2$ and $t_3 \cap t_4$ need to be calculated first, as an unary operator and operators in brackets take priority. Then, $\sqcup$ and $\wedge$ combine results of both in brackets, which is taken as $\sim$ in the end.

A solid target in a policy could be illustrated as a construction of a tree with nodes, where leaf nodes are atomic targets and non-leaf nodes are logic gates. We employ a policy tree structure to combine these atomic targets. The lower expression above was illustrated in Figure 4.5.

The result of the target expression indicates whether a policy matches a given query. Only $1_T \in Dec_T$ for target evaluation represents that the policy is applicable. More formally,

-If t evaluates to 1_T, it evaluates p;

-If t evaluates to 0_T, $\perp_T$ and $\times_T$ it stops further evaluation.

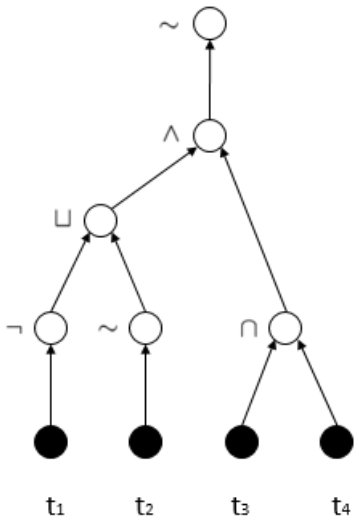

Figure 4.6: Target Tree A

However, to make the expressions more brief, we define u-ary operators which perform on more than two atomic elements. The **and** and **or** pair for multi-elements are $\triangle$ and $\triangledown$, corresponding to $\sqcup$ and $\sqcap$ in binary operators. They inherit the same Dec_T priority order as $\sqcup$ and $\sqcap$, which is $1_T > 0_T > \perp_T > \times_T$.

More formally, for $\triangle$, it selects the highest value among candidates as a result. While $\triangledown$ takes the lowest value among candidates as a result. For instance, $\triangle[1_T, 0_T, \perp_T, \times_T]= 1_T$, $\triangledown[1_T, 0_T, \perp_T, \times_T] = \times_T$.

Even though the u-ary operators can be translated as a series of binary operators, it enables the atomic tree structure to be briefer and reduces the numbers of operators by combining many binary operators as one. Hence, to illustrate u-ray operators, we provide an example displayed under the tree structure in Figure 4.6. In this example, u-ary operators are utilised to combine atomic target as: $\triangledown[t_1, t_2, t_3, t_4, t_5]$; $\triangle[t_6, t_7, t_8, t_9]$.

4.3.4 Target Equivalence

For two targets t_1 and t_2, if for all requests, the evaluation are always the same, $[\![t_1]\!](q)=[\![t_2]\!](q)$, then t_1 and t_2 are equivalent targets, $[\![t_1]\!] = [\![t_2]\!]$. We show following priorities of our target operators.

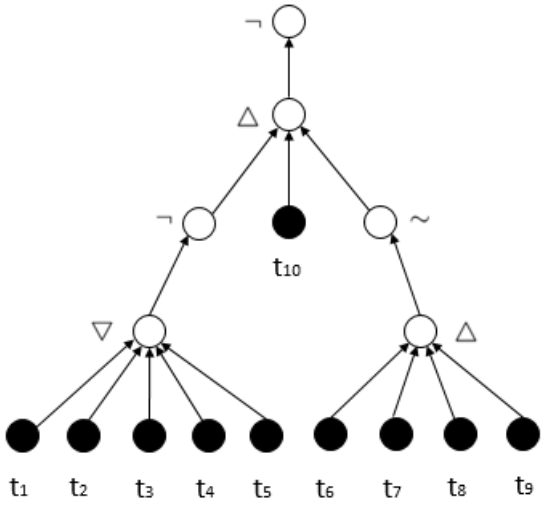

Figure 4.7: Target Tree B

Proposition 1 ∀ t and t',

$$[\![\neg(\neg t)]\!]=[\![\ t\ ]\!];\ [\![\sim(\sim t)]\!]=[\![\ \sim t\ ]\!]$$

$$[\![\sim(\neg t)]\!] \neq [\![\neg(\sim t)]\!]$$

$$[\![\neg(t \sqcup t')]\!] \neq [\![(\neg t) \sqcap (\neg t')]\!];\ [\![\neg(t \sqcap t')]\!] \neq [\![(\neg t) \sqcup (\neg t')]\!]$$

$$[\![\sim(t \sqcup t')]\!]=[\![(\sim t) \sqcup (\sim t')]\!];\ [\![\sim(t \sqcap t')]\!]=[\![(\sim t) \sqcap (\sim t')]\!]$$

Proof The above equations can be proven by checking the four-valued "truth" table which are the definitions of the operators.

Proposition 2 ∀ t_1, t_2 and t_3.....t_n,

$$\triangle[t_1, t_2, t_3] = t_1 \sqcup t_2 \sqcup t_3;$$

$$\triangledown[t_1, t_2, t_3] = t_1 \sqcap t_2 \sqcap t_3;$$

$$\triangle[t_1, t_2 \ldots\ldots\ t_n] = t_1 \sqcup t_2 \sqcup \ldots\ldots\ t_n;$$

$$\triangledown[t_1, t_2 \ldots\ldots\ t_3] = t_1 \sqcap t_2 \sqcap \ldots\ldots\ t_3;$$

Proof The above equations can be proven by checking the four-valued "truth" table which are the definitions of the operators.

4.3.5 On functional Completeness

By way of motivation, we wish the operations we defined could express all functions which users would like to define. Hence, we prove $\forall f: Dec_T^n \rightarrow Dec_T$, n$\in Z^*$, where f can be expressed by at least one combination of $\neg$, $\sim$, $\star$ and $\cup$. This property is proven by the logic expressions defined by $\neg$, $\sim$, $\star$ and $\cup$ over Dec_T is functional completeness. We prove this by introducing a theorem from Jobe [106].

Theorem 1 (Jobe) 1962 The three-valued logic E expressed over the set 1,2,3 and defined by the operators $\bullet$, E_1 and E_2, given in the figure below, is functionally complete.

$\bullet$	3	2	1	E_1	E_2
3	3	2	1	3	1
2	2	2	1	1	2
1	1	1	1	2	3

Table 4.13: Over the Set {3,2,1}

Theorem 2 J-operators and functional completeness.[106] In order to present a new and simple proof of functional completeness, we define the operators $J_k(P_1, P_2, ...P_n)^l$, which have the following interpretation. $J_k(P_1, P_2, ...P_n)^l$ represents the truth table of a possible formula of order n which has the truth value k in the i^{th} row and the truth value 1 in all other positions, where (1≤k≤M) and ($1 \leq i \leq M^n$).

Corollary 2 Logic operations $\cup$, $\neg$, $\sim$, and $\star$ is functionally complete for expressions over four-valued set $\{1_T, 0_T, \perp_T, \times_T\}$.

$\bullet$	1_T	0_T	$\perp_T$	$\times_T$	E_1	E_2	E_3
1_T	1_T	0_T	$\perp_T$	$\times_T$	0_T	1_T	$\perp_T$
0_T	0_T	0_T	$\perp_T$	$\times_T$	1_T	$\perp_T$	$\times_T$
$\perp_T$	$\perp_T$	$\perp_T$	$\perp_T$	$\times_T$	$\perp_T$	0_T	1_T
$\times_T$	$\times_T$	$\times_T$	$\times_T$	1_T	$\times_T$	$\times_T$	0_T

Table 4.14: (Over the Set $\{1_T, 0_T, \perp_T, \times_T\}$

Proof Based on the theorem in Jobe's paper[106], the operators we tailored for our four-valued set is functionally complete.

4.4 Access Control Policy

When the *target* section is evaluated to be true, this indicates that the request is within the effective range of the policy. In the access control section, we still employ partitions in provenance as *atomic conditions*. The access control policy section determines that the access request can be allowed/denied if the provenance graphs contain certain provenance partitions in the policy. Based on both target section and access control section employing provenance partitions as conditions, the model of atomic condition takes the same form as the atomic target. However, the difference is that each section may take different provenance partitions as conditions.

Here, we formally define atomic conditions. Let $\{1_P, 0_P, \perp_P, \times_T\} \in Dec_P$ is a decision set;
- p $\in Dec_P$ is an atomic condition, where p is a value in the decision set;
- a provenance partition is an atomic condition;
- (t$\overset{\diamond}{\rightarrow}$p{tag}) is a policy; and t is an atomic target, where $\diamond$ is $\prec$ or $\succ$ ($\prec$ and $\succ$ are two approaches to transform atomic *target* as atomic condition);
- ($^{*}t \overset{\diamond}{\rightarrow}$p{tag}) is an atomic condition; where * is a logic unary expression over an atomic target, and $\diamond$ is $\prec$ or $\succ$;
- $p_1 \bullet p_2$ which is a conjunction policy of p_1 and p_2, where $\bullet$ is an operator connection two atomic conditions;
- $pbd_P\ p$ permit by default policy is a policy p, where pbd_P can be treated as an operator to map a four-valued decision set to a two-valued decision set (Yes or No);
- $dbd_P\ p$ deny by default policy is a policy p, where pbd_P can be treated as an operator to map a four-valued decision set to a two-valued decision set (Yes or No);

Similarly, with the target section, we still utilise a tree structure to organise atomic conditions. The tree structure enables proper visualisation of a policy and supports a flexible

and powerful combination of atomic conditions. Even though atomic conditions employ different provenance partitions, they take the same model to extract elements from provenance.

Atomic conditions could be provenance partitions already existing in the target section or in newly defined provenance partitions. To be more specific, atomic access control policies could recall conditions in *target* to reduce redundancy. Their conditions could be re-used as a form of transformation which will be introduced in the later part of this section. So it becomes apparent that the atomic access control policies could also define new provenance partitions as conditions.

The following two truth tables define how to re-use the transformation of atomic target policies as atomic access control policies. *Atomic targets* are forwardly or adversely referred to a value in the four-valued decision set $\{1_P, 0_P, \perp_P, \times_P\}$. Just as the tables shown below, the transformation takes the form as $\mathrm{t}\overset{\diamond}{\rightarrow}\mathrm{p}\{\mathrm{tag}\}$, and the $\{\mathrm{tag}\} \in \{1, 0, \times, \perp\}$. When t is referred to p forwardly, the tag $\times$ keeps the results of t; $\perp$ changes $\times_T$ and $\perp_T$ as $\perp_P$; *0* changes $\times_T$ and $\perp_T$ and 0_T as 0_P; *1* increases all the values as 1_P. On the contrary, *1* for inverse reference keeps values of t; *0* decreases 1_T as 0_P; $\perp$ decreases 1_T, 0_T as $\perp_P$; and $\perp$ downgrades all values of t as $\perp_P$.

$t \prec p$	1	0	$\perp$	$\times$
1_T	1_P	1_P	1_P	1_P
0_T	1_P	0_P	0_P	0_P
$\perp_T$	1_P	0_P	$\perp_P$	$\perp_P$
$\times_T$	1_P	0_P	$\perp_P$	$\times_P$

Table 4.15: $t \prec p$

$t \succ p$	1	0	$\perp$	$\times$
1_T	1_P	0_P	$\perp_P$	$\times_P$
0_T	0_P	0_P	$\perp_P$	$\times_P$
$\perp_T$	$\perp_P$	$\perp_P$	$\perp_P$	$\times_P$
$\times_P$	$\times_P$	$\times_P$	$\times_P$	$\times_P$

Table 4.16: $t \succ p$

Moreover, to support all the possibility of references for the atomic target, *atomic targets* can be transformed by unary operators before referred to p, which takes the form of ${}^{*}t \overset{\diamond}{\rightarrow}\mathrm{p}\{\mathrm{tag}\}$. We show an example of logic expression over t in Table 17 as: $\sim(\neg t)$, where t $\in \{1_T, 0_T, \perp_T, \times_T\}$.

t	$\sim(\neg t)$
1_T	0_P
0_T	$\perp_P$
$\perp_T$	1_P
$\times_T$	$\times_P$

Table 4.17: t$\rightarrow\sim(\neg t)$

$\sim(\neg t) \prec p$	1	0	$\perp$	$\times$
1_T	1_P	0_P	0_P	0_P
0_T	1_P	0_P	$\perp_P$	$\perp_P$
$\perp_T$	1_P	1_P	1_P	1_P
$\times_T$	1_P	0_P	$\perp_P$	$\times_P$

Table 4.18: $\sim(\neg t) \overset{\prec}{\rightarrow}$ p

$\sim(\neg t) \succ p$	1	0	$\perp$	$\times$
1_T	1_P	0_P	$\perp_P$	$\perp_P$
0_T	1_P	1_P	1_P	1_P
$\perp_T$	1_P	0_P	0_P	0_P
$\times_T$	1_P	0_P	$\perp_P$	$\times_P$

Table 4.19: $\sim(\neg t) \overset{\succ}{\rightarrow}$ p

Lemma 1 $\forall t \in Dec_T$, ${}^*t \overset{\diamond}{\rightarrow}$ptag supports all the possibilities for (t, p), where $\diamond$ is $\prec$ or $\succ$, * is unary expressions over $\neg$ and $\sim$.

Proof ${}^*t \overset{\diamond}{\rightarrow}$p is functionally completed, as ${}^*t$ can express all the order for a four-valued set and $\diamond$ takes both forwardly and reversely. We firstly list all the orders of a four valued set via $\neg$ and $\sim$.

t	$\star(\neg(\star t))$	$\sim t$	$\neg t$
1_T	1_T	1_T	0_T
0_T	0_T	$\perp_T$	1_T
$\perp_T$	$\times_T$	0_T	$\perp_T$
$\times_T$	$\perp_T$	$\times_T$	$\times_T$

Table 4.20: All Orders of a “Four-valued" set by $\neg$ and $\sim$

Therefore, we can conclude the functional completeness of ${}^*t \overset{\diamond}{\rightarrow}$p.

A conjunction of policies is a policy. The motivation to define pbd_PP and dbd_PP is that we need to transfer a four-valued decision set to a two-valued result (permit or deny access). However, it could be processed by combining results of all applicable policies. This transferring can be defined by each policy, or each system can define pbd_P or dbd_P as the

default process. A permit by a default policy returns 1_P if the output of P is 0 or 1; while it returns $\perp_P$ otherwise. On the contrary, a deny by default policy returns 1_P if the output of P is 1; while it returns $\perp_P$ if the output of P is 0 or $\perp_P$. We define the truth table of both as below.

P	pbd_PP	dbd_PP
1_P	1_P	1_P
0_P	1_P	$\times_P$
$\perp_P$	1_P	$\times_P$
$\times_P$	$\times_P$	$\times_P$

Table 4.21: Truth Table for pbd_PP and dbd_PP

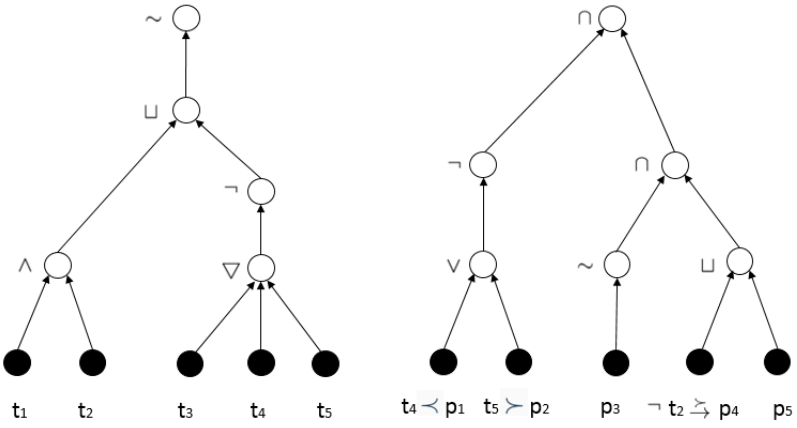

Figure 4.8: Example Policy Trees

$$\sim (t_1 \wedge t_2) \cup (\neg(\triangle[t_3, t_4, t_5]))$$

$$\neg\, (t_4 \prec p_1 \vee t_5 \succ p_2) \cap (\sim p_3 \sqcup ((\neg t_2 \overset{\succ}{\rightarrow} p_4) \cap p_5\ )$$

In Figure 4.6, we provide a policy example which includes target policy and access control policy. For this example, atomic target t_2 t_4, and t_5 was assigned as atomic conditions by $t_4 \prec p_1$, $t_5 \succ p_2$, and $\neg t_2 \overset{\succ}{\rightarrow} p_4$.

Clearly, the frameworks of Chapter 3 and Chapter 4 introduce provenance partitions as conditions of policies. However, in provenance access control policies, the decision set is a two-valued. And we defined a four-valued decision set for provenance-based access control policies. Because that provenance-based access control policies mainly depend on attributes of provenance as conditions. The access control policies proposed in Chapter 3 is to access

provenance. Hence, the evaluation for provenance partitions is more fine-grained, where we employ a four-valued decision set in the framework.

4.4.1 Policy Operators

The operators defined in the previous section can be used for access control policy tree structure. In the policy tree structure, logic operations are non-leaf nodes in the policy trees. Here, we introduce two operations inspired by the operators provided by Crampton and Morisset [19]. Compared to other operators we proposed, the operators that we define here are aware the orders of inputs. Namely, when the orders of two inputs switch, the output is not always the same.

$\oplus$	1_P	0_P	$\perp_P$	$\times_P$
1_P	1_P	x	1_P	1_P
0_P	y	0_P	0_P	0_P
$\perp_P$	1_P	0_P	$\perp_P$	$\perp_P$
$\times_P$	1_P	0_P	$\perp_P$	$\times_P$

Table 4.22: Idemponent

$\oplus_\cup$	1_P	0_P	$\perp_P$	$\times_P$
1_P	1_P	1_P	1_P	1_P
0_P	0_P	0_P	0_P	0_P
$\perp_P$	1_P	0_P	$\perp_P$	$\perp_P$
$\times_P$	1_P	0_P	$\perp_P$	$\times_P$

Table 4.23: Idemponent $\oplus_\cup$

$\oplus_\cap$	1_P	0_P	$\perp_P$	$\times_P$
1_P	1_P	0_P	1_P	1_P
0_P	1_P	0_P	0_P	0_P
$\perp_P$	1_P	0_P	$\perp_P$	$\perp_P$
$\times_P$	1_P	0_P	$\perp_P$	$\times_P$

Table 4.24: Idemponent $\oplus_\cap$

$\triangleright$	1_P	0_P	$\perp_P$	$\times_P$
1_P	1_P	1_P	1_P	1_P
0_P	0_P	0_P	0_P	0_P
$\perp_P$	1_P	0_P	$\perp_P$	$\perp_P$
$\times_P$	1_P	0_P	$\perp_P$	$\times_P$

Table 4.25: First-applicable

$\triangleleft$	1_P	0_P	$\perp_P$	$\times_P$
1_P	1_P	0_P	1_P	1_P
0_P	1_P	0_P	0_P	0_P
$\perp_P$	1_P	0_P	$\perp_P$	$\perp_P$
$\times_P$	1_P	0_P	$\perp_P$	$\times_P$

Table 4.26: First-applicable

We list the truth tables for $\oplus$, $\triangleright$ and $\triangleleft$ as above. In Table 18, the outputs are no longer

symmetrical at two sides of the diagonal, where inputs of $\{1_P, 0_P\}$ and $\{0_P, 1_P\}$ differ. For $\oplus$, the priority of four valued sets are $1_P = 0_P > \perp_P > \times_P$. Here, we employ $_\cup$ and $_\cap$ to decide x and y when 1_P meets 0_P. In particular, $\oplus_\cup$ takes the former one as the result, while $\oplus_\cap$ takes the latter one as the result. Based on $\oplus$, we can define more operations with the priority order s such as $1_P > 0_P = \perp_P > \times_P$, $1_P = \perp_P > 0_P > \times_P$ *etc.*

Similarly, $\triangleright$ is another logic operator that is aware of the order of inputs, and $\triangleright$ emphasises the first input. When the first input is 1_P or 0_P, it covers the second input and was outputted as the result. While, when the first input is $\perp_P$, the second input is the result. $\triangleleft$ effects with the opposite rule by emphasising the second input.

4.5 A case study

In order to better deliver the idea, we generate a sample policy and a provenance graph. Thus we output a result of the policy based on the provenance graph.

Target Policy Section:

Atomic Target (1): Bob uploads a piece of data and Alice submits it;

Atomic Target (2): The data was reviewed on Monday and was graded at 30/5/2018;

Access Control Policy Section:

Atomic Condition (1): Bob replaces the data;

Atomic Condition (2): it was revised after reviewed;

Atomic Condition (3): it was re-submitted on Wednesday;

A request was sent to access a piece of data in a database, the server retrieves its provenance graph and makes a decision based on the sample policy. Begin with the target section, referring to the provenance graph, we can see Bob uploaded the data “a1v1" and Alice submitted it. Hence, the results for the atomic target (1) is total match 1_T. Next, in the provenance graph, the data was reviewed at Monday but graded at 30/5/2018 instead of 30/4/2018, which indicates that the provenance graphs partial match the atomic target (2), where its result is 0_T. The results of the atomic targets were merged by the operator $\sqcup$. Referring the Table 4.2 of this chapter, $1_T \sqcup 0_T = 1_T$. It shows that the given provenance graph matches that target section,

therefore, we move to the access control policy section. Similarly, as Bob did not replace the data, the result for atomic conditions (1) is $\perp_P$ which merged by the operator $\triangle$ with outputs of atomic condition (2) $\cap$ atomic condition (3). The final output of this policy is 1_P. The request to access the data was accepted.

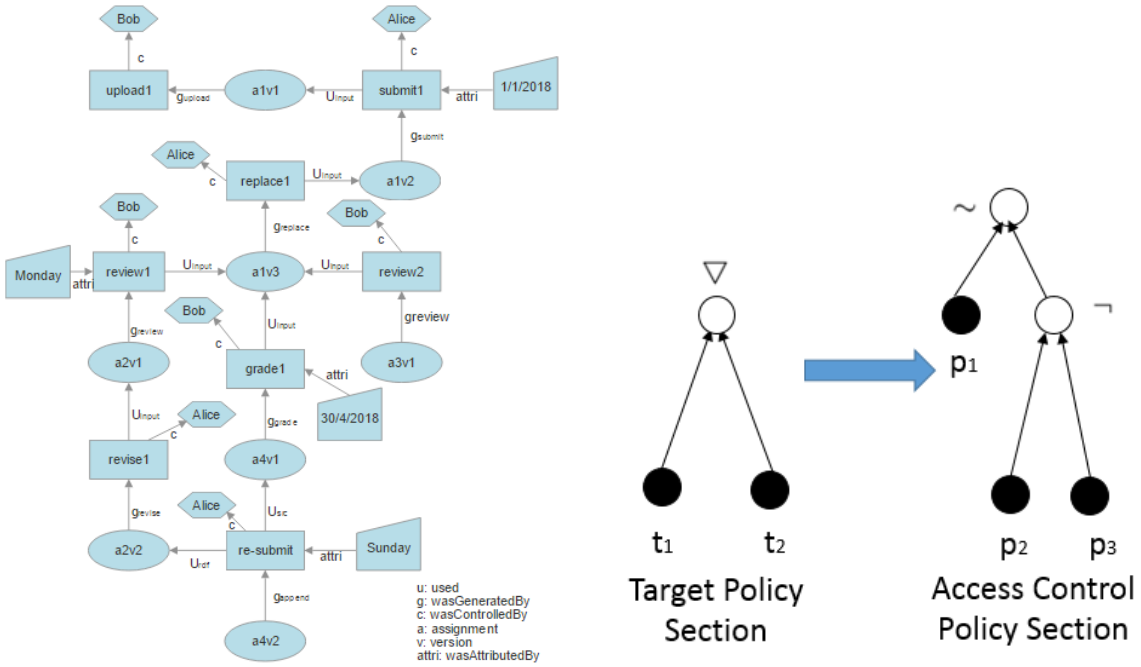

Figure 4.9: Experiment

4.6 On Integrity of Provenance

In this provenance-aware system, as provenance is taken as a condition to evaluate accessibility, malicious manipulation or integrity damage to provenance graphs might result in forged results of an evaluation. In such a scenario, an access control policy prohibits access to a query when the targeted object is graded. Hence, if the graded operation in provenance is lost or maliciously tampered with, the output of the atomic target would be changed from 1_P to $\perp_P$, which might further result in a different result of the policy tree.

Hence, the systems can employ cryptography mechanisms to protect and confirm the integrity of provenance and then make access control evaluation based on the provenance graphs. In simple terms, the server can generate a key pair and send provenance owners the public key to sign a digital signature with provenance. When provenance graphs are invoked

to evaluate a query, the server checks the digital signature then process the evaluation.

4.7 Evaluation

The claimed contributions of this framework are that we introduce a four-valued decision sets for each condition, and define a set of operators to merge the values of the “four-valued" decisions. Apparently, the claimed requirements were established in the demonstration. The experimental policies were successfully generated under this policy model. In addition, the evaluation criterion can be applied to generate results of policies based on specific provenance graphs.

In order to proof the availability and practicality of our proposed framework and to evaluate its performance, we did experiments to implement policy generation and policy algebras.

We did the experiments on a 3.40 GHz Intel Computer with 16GB of memory. The machine’s operating system is Microsoft Windows 8 and the database is Oracle Database. The goal of the experiments is to evaluate the time taken for generating policy and the time taken to process the policy algebras. In this experiment, the size of the generated policies is also examined in terms of atomic policies and policy operators.

We implemented synthetic datasets based on the version proposed by Wisconsin Benchmark [107]. We set up a random provenance-based access control policy generator to generate atomic policies and policy operators. Each policy contains a random number of atomic policies ($\geqslant$2) and a random number of operators ($\geqslant$1).

At first, the average time span to obtain provenance-based access control policies and size of policies are measured. The whole policy generation processes include random atomic policies selection to random operators selection. In the figures below, the left y-axis shows the average time span (in seconds) to generate policies, and the x-axis indicates the numbers of atomic targets.

Next, we simulate the process of policy algebras. Different from the process above, the x-axis shows the numbers of operators for each policy, and the y-axis displays the time spans to process the policy algebras. Obviously, the more operators of a policy take, the longer time span to cost as the figure showed.

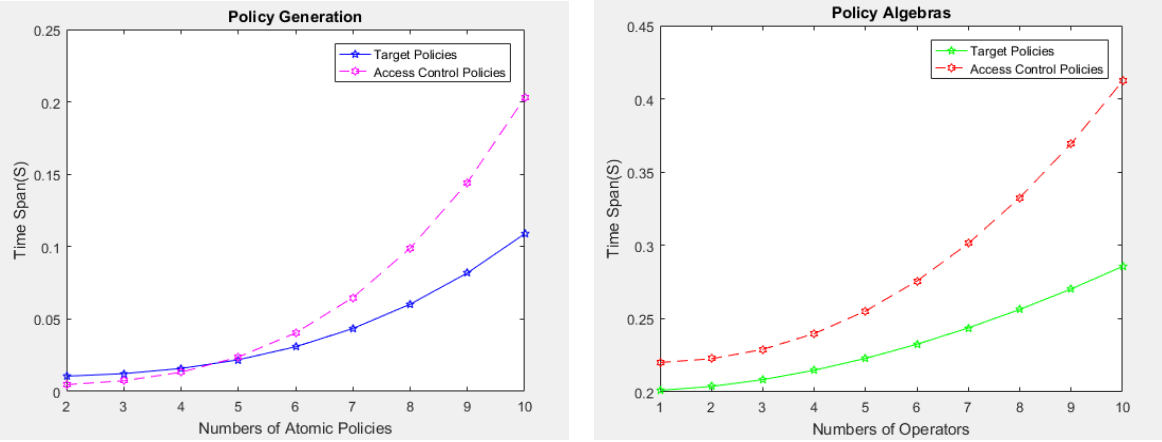

Figure 4.10: Experiment

4.8 Conclusion

In this chapter, we propose a fine-grained provenance-based access control policy model by defining atomic condition and policy algebras, which utilise provenance as conditions to determine accessibility of data. Specifically, several types of atomic targets are provided in this chapter. Path atomic target takes attributes in provenance as conditions, which implies that if certain operations are executed on the targeted data, it can or can not be accessed. Associated atomic target takes both attributes of provenance and requests as conditions. This indicates that if the requestor has performed certain operations on the targeted data, it can or can not be accessed. Each atomic target takes the form of a string of vertices, where each vertex is defined by a quaternion of attributes.

Moreover, the results for an atomic target of our provenance-based access control policy is one element in a four-valued set. New logic operators to merge the four-valued results are proposed in the chapter. We also did experiments to implement the proposed framework. The time span to generate policies and implement policy algebras are measured.

5

Purpose-based Access Policy on Provenance and Data Algebra

Following the previous two chapters, we propose purpose-based access policies in this chapter. Different from provenance-based policies that determine if a piece of data can be accessed or not, purpose-based access policies determines for what purposes can data be accessed. To determine that data can only be accessed for allowed purposes is a crucial mechanism of access control. Because under certain scenarios, even though data is allowed to be accessed, it can only be accessed for the permitted purposes for security consideration. Particularly, the purposes can be classified as different sensitivity levels. Accordingly, we tailor policy algebras to include internal and external policy operators for hierarchical purposes, in order to merge purpose sets generated by individual policies.

5.1 Introduction

In provenance-aware systems, to preserve security and privacy of data and determine proper access, traditional access control policies cannot address all the issues. Only generating decisions about whether a piece of data allowed to be accessed cannot meet all the requirements of security protection. Access policies specify laws or preferences for intended purposes, retention, condition, obligation *etc.* They also confine accessibility of data, such as for which purposes can data be accessed. Hence, purpose-based access policies can specify restrictions that traditional access control policies can not realise, including determining allowed and prohibited access purposes based on provenance.

Provenance can contribute to access policies. It can be noticed that provenance is closely connected with access policies and can provide crucial information regarding the sensitivity of data because provenance is a file which records historical operations performed on a piece of data, where these historical operations can be related to which purposes the data can be accessed for. Hence, provenance can be employed as conditions to map intended usages. For example, an access policy can be defined as: only if a piece of data was submitted to an educational institution, it can be used for the purposes of *research* and *education*. Clearly, access purposes in our provenance-based access policies are determined by whether the provenance contains the given partitions.

5.1.1 Related Work and Motivations

Several existing works proposed access policies [108][109] which map attributes in queries, roles and system conditions to permitted usages, retention, condition, obligation, *etc.* Byun *et al.* [108] presented a comprehensive approach for privacy-preserving access control based on the notion of purpose. In their model, purpose information associated with a given data element specifies the intended use of the data element. A key feature of their model is that it allows multiple purposes to be associated with each data element and also supports explicit prohibitions, thus allowing privacy officers to specify that some data should not be used for certain purposes. A discussion arisen by Byun *et al.*[108] is that it would be more advantageous to organize purposes in a directly acyclic graph instead of a tree construction. However, In our framework, an acyclic graph is a better option to organise purposes, because it can support the possibility that a node could have more than one ancestor and where it is

closer to the nature of the relationship of purposes.

Based on the idea they proposed, we utilise provenance subgraphs as conditions for access policies. because both the processes that have previously been done and also who executed the processes are vital conditions in determining for what purposes a piece of data can be accessed for. In our proposed provenance-based access policy, we not only employ provenance partitions as conditions, but also distinguish the hierarchies of purposes, in order to aware the different sensitivities of purposes. The hierarchies of purposes are mainly when conflicted policy results arise. Another reason to define the different hierarchies of purposes is that, when merging purpose sets determined by each policy, purposes in different hierarchies should be combined via different operators. For instance, let a collection of purpose be {*Analysis, Admin, General-Purpose*}, the sensitivity of the three elements are *Analysis* > *Admin* > *General-Purpose*. It indicates that the *General-Purpose* is the least sensitive purpose, which can be granted to the public, while the purpose of *Analysis* is the most sensitive purpose in the set and can only be accessed by advanced users.

Accordingly, policy algebras to merge these purpose sets are also tailored in this chapter. This will enhance the performance of algebras to distinctively combine purposes in different hierarchies. To the best of our knowledge, previous access policy algebras to merge purpose sets have not distinguished purposes in various hierarchies. Hence, it is attractive to define functions which can combine purposes in different hierarchies by different operators, especially, when combining two collections of purposes. We hope to conjunct usages with a lower rank and to take an intersection of usages with a higher rank.

Here, we provide a scenario as the application of this policy. When a request was sent to a database aiming to access a piece of data. The database server refers to the access policies and decides for which purposes can the nominated data being accessed. Let a sample Policy *A* be: if the data was collected by the government, it can be used for data analysis; if the data was reviewed by a university, it can be used for research and education. By confirming with its provenance graph, the data are permitted to access for data analysis$_{HH}$ which is the higher hierarchy purpose and research and education$_{LH}$ which are the low hierarchy purposes. Similarly, Policy *B* permits the data can be accessed by the same user for auditing$_{HH}$ and education, marketing$_{LH}$. The database system regulates that higher hierarchy purposes granted by different policies are merged by the operator *union* and lower hierarchy purposes are merged by the operator *intersection*. Hence, the final permitted purposes are

$\{$data analysis, auditing$\}_{HH}$ = $\{$data analysis$\}_{HH} \cup \{$auditing$\}_{HH}$, and $\{$education$\}_{LH}$ = $\{$research and education$\}_{LH} \cap \{$education, marketing$\}_{LH}$.

5.1.2 Our Contributions

In this chapter, we propose a framework for purpose-based access policies on provenance. Firstly, we define the semantics and syntax of atomic access policies which map conditions to a set of permitted or prohibited usages. To the best of our knowledge, we are first to utilise provenance as conditions to tailor access policies. We also define how to specify a set of purposes for each policy.

Even though purposes were classified based on various sensitivities in previous work, corresponding algebras have not previously been proposed. Concretely, to combine two sets of purposes, purposes with different sensitivities are merged by different operators in this work. Therefore, we particularly design functions for algebras of access policies involving purposes. In addition, the algebras in this framework consist of internal algebra and external algebra.

5.1.3 Chapter Organisation

This chapter was organised as:

- Provenance-based Access Policies is defined formally in Section 5.2, where we present its system assumption, syntax, semantics and a case study.
- In Section 5.3 and Section 5.4, we propose the algebras by defining internal and external operators.
- In Section 5.5, experiments to evaluate the policies are presented and the conclusion is presented in Section 5.6.

5.2 Purpose-based Access Policy on Provenance

In this chapter, we propose access policies that determine a set of allowed or prohibited purposes based on provenance. As provenance logs operations performed on a piece of data, the access purposes were determined according to the historical records. Historical operations provide crucial clues for the sensitivity of data. Hence, the fine-grained access policies map provenance and attribute in a request to a collection of allowed or prohibited purposes.

5.2.1 System Architecture

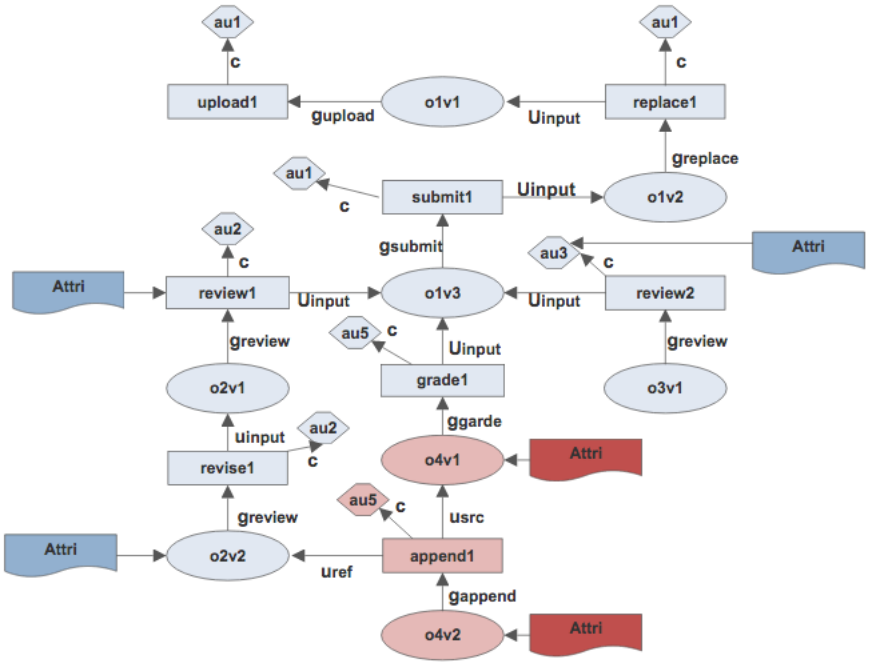

Figure 5.1: System Architecture

The system architecture for purpose-based access policy is illustrated in Figure 5.1. This system architecture can be applied to other policy models proposed in the previous two chapters. In Chapter 3 and Chapter 4, we only showed the system assumption. For the consideration of the balance of chapters, we provide the system architecture in this chapter. The system architecture consists of Policy Administration Point (PAP) which generates access policies and Policy Manager that retrieves policies or results of policies from other parties. Context handler forwards access requests received by Policy Enforcement Point (PEP) to Policy Decision Point (PDP) where the decisions are made according to policies. Moreover,

Policy Information Point (PIP) collects required attributes for the policy evaluation from subject, environment, provenance *etc.* Particularly, provenance partitions are retrieved by the context handler and sent to PDP, aiming to generate access purposes. There are two unique features of the system architecture of this framework. One is retrieving provenance partitions as conditions to generate access purposes, the other one is that PDP merges policies from both internal policies and external policies to generate a final set of access purposes for a given piece of data or query. During the process to produce data, it can be executed by more than one data owners who would like to generate their own access policies. It motivates us to propose a policy algebra to merge results of policies from various parties. In the end, the final decision generated by PDP is sent to PEP for enforcement.

5.2.2 Semantics

Firstly, we define basic elements for the purpose-based access policies, including provenance partitions as atomic policies, access trees to organise atomic policies, purpose graphs, and purpose sets. Purpose graphs which list all possible access purposes are defined by a system, where each policy determines a set of allowed and prohibited purposes in the purpose graphs.

Definition 1 (Provenance Partition). We define *provenance partition* in Chapter 3 of this thesis, which is a connected subgraph in provenance DAG. For a purpose-based access policy on provenance, provenance partitions are utilised as conditions of a policy. Concretely, in our framework, a provenance partition is a collection of vertices in a provenance DAG under the OPM$^+$, which represents one operation or a series of operations recorded in provenance. The historical transactions propose cues for the sensitivity of data and are employed as policy restrictions, which can determine access purposes. Here we provide sample policies to illustrate the connection between provenance and purposes.

(1) *If a document has been reviewed by at least three educational experts, then it can be accessed with the aim of education.*

(2) *If a file has been submitted before 2016 and graded by a teaching staff, it can not be accessed for the purpose of revision.*

(3) *If a piece of data was collected by the government, it can not be accessed for academic research.*

(4) *If the data has been uploaded after two different reviews, it can be accessed for market analysis.*

(5) *If the files have been reviewed after 2016, they can not be accessed for audit and direct-use.*

(6) *If the files have been edited by Alice and revised in Sydney, they can be accessed for service-maintenance and service-updates.*

The conditions of purpose-based access policy on provenance and provenance-based access policy are the same. The difference between them is that the former maps the conditions to access purposes, and the latter maps conditions to a decision value. The decision of access policy indicates if the data can or not be accessed. While access purposes are for which usages can a piece of data to being accessed. Here we define the atomic conditions of purpose-based access policy as:

-$null_T$ is a condition;

-(v_{type}, v_{name}) is a condition, where v_{type} is a vertices type and v_{name} is a vertices name;

-$(v_{type}, v_{name}, a, f)$ is a condition, where v is a vertices, a is an attribute value and f is a binary predicate.

-$(v_{type}, v_{name}$, x, f) is a condition, where (v_{type}, v_{name}) is a vertices, x is an attribute in a query, and f is a binary predicate;

-a target is a string of (v_{type}, v_{name}) or $(v_{type}, v_{name}, a, f)$ or $(v_{type}, v_{name}$, x, f) expressed over XPath.

A provenance partition is defined over XPath, which is a collection of vertices. It represents historical operations performed on a piece of data. Provenance partitions are conditions of the policies. Moreover, the attributes of a query could also be utilised as conditions, which is similar to the approach proposed in Chapter 4 of this thesis. We do not repeat it here.

Definition 2 (Atomic Condition Evaluation). The four-valued set for atomic policies is

$\{1_p, 0_p, \perp_p, \times_p\}$. Between totally matching a provenance partition (1_p) and totally not matching it ($\times_p$), there are two values (0_p, $\perp_p$) to represent the status between them. Specifically, if a provenance graph meets all attributes of a partition, it outputs 1_p representing a total match; if a provenance graph contains all the types of vertices name in a partitions, but it does not meet all the attribute values, it output 0_p; if a provenance graph only contains all the vertices types in a partitions, but not meet vertices names and attributes, it outputs $\perp_p$; if a provenance graph does not meet any item, its result is $\times_p$.

Definition 3 (Tree Structure). The condition of each policy is a combination of provenance partitions which are organised under a tree structure. In the tree structure, leaf nodes are atomic conditions (provenance partitions) and non-leaf nodes are operators to merge evaluation results of atomic conditions. The tree structure organises conditions that are merged by logic operators. The output of an access tree structure is “Yes" or “No". “Yes" indicates to grant a collection of access purposes, while “No" indicates not to grant a collection of access purposes. Particularly, the operators to merge results of each condition include unary, binary and u-ary defined in Chapter 4 of this thesis. They are still applicable to the framework in this chapter.

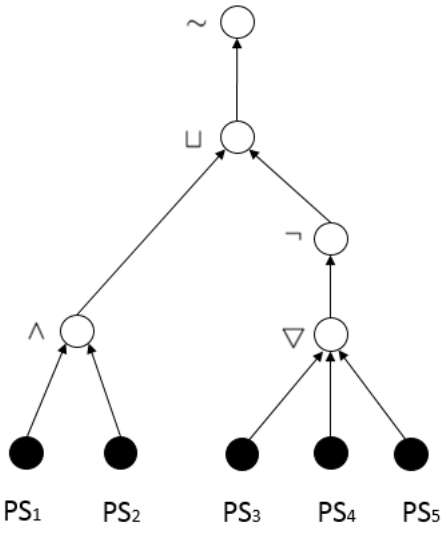

Figure 5.2: Access Tree of Provenance Segments

Definition 4 (Purposes). An access purpose P implies the aim that a piece of data is being accessed for, such as *military*, *education*, *research*, *etc.* Purpose-based access policies define a collection of allowed and/or prohibited purposes for data.

In our framework for purpose-based access policies on provenance, purposes with various sensitivities are organised as a form of a directed acyclic graph (DAG). It is denoted as a Purpose Graph (PG). In a PG, each node represents a purpose P. Nodes in a PG are displayed in several layers and are linked by edges, where the relationship between an ancestor node and a descendant node are generalisation and specialisation. In other words, purposes near the root are more general, while the purposes on the leaf side are more specialised. Hence, each descendent node is a more specialized purpose of its ancestor node. The sensitivity of the hierarchical purposes varies, based on the layers they display. We distinguish the hierarchies of purposes aiming to classify them based on the sensitivities, and further employ functions to merge purposes with different hierarchies by different operations.

Particularly, there are existing works [108][109] organising purposes with a tree structure, where each descendant can only have one ancestor. However, in our framework, we replace the tree structure as a DAG. Because in a DAG, a single node can be associated with more than one ancestor. It is more close to the nature of relationships of purposes. We propose an example purpose DAG in Figure 5.3. In this example, *Admin* is a descendant of *General Purpose*. Hence, it is a more specific purpose comparing with *General Purpose*.

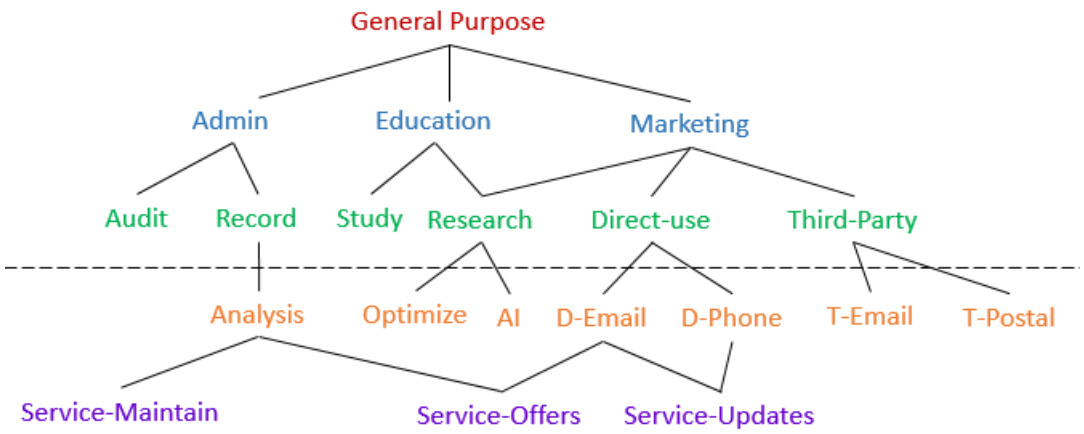

Figure 5.3: Purpose DAG

Definition 5 (Purpose Sets). A purpose set defined in each access policy is a collection of purposes in a purpose DAG. Formally, let a set of purposes $\in PG$ be a purpose set, which is denoted as PS.

- Ancestors (P), denoted by $P^{\uparrow}$. In a PG, $P^{\uparrow}$ is the set of all ancestor nodes of P, including p itself. For instance, in the example purpose DAG provided above, Analysis$P^{\uparrow}$ =

{Analysis, Record, Admin, General Purpose}.

- Partial Ancestors (P), denoted by $P^{\uparrow\alpha}$. In a PG, $P^{\uparrow\alpha}$ is the set of α ancestor nodes of P including p itself. For instance, Analysis$P^{\uparrow 3}$ = {Analysis, Record, Admin}.

- Descendants (P), denoted by $P^{\downarrow}$. In a PG, $P^{\downarrow}$ is the set of all ancestor nodes of P, including p itself. For instance, Admin$P^{\downarrow}$ = {Admin, Audit, Record, Analysis, Service-Maintain, Service-Offers}.

- Partial Descendants (P), denoted by $P^{\downarrow\beta}$. In a PG, $P^{\downarrow\beta}$ is the set of β ancestor nodes of P including p itself. For instance, Admin$P^{\downarrow 3}$ = {Admin, Record, Analysis}.

- $P^{\updownarrow}$ denotes the set of all ancestors and descendants of P in a PG, where $P^{\updownarrow} = P^{\uparrow} + P^{\downarrow}$. For instance, Record$P^{\updownarrow}$ = {General Purpose, Admin, Record, Analysis, Service-Maintain, Service-Offers}.

Definition 6 (Allowed Purpose and Prohibited Purpose). A purpose set could be denoted as Allowed Purpose (AP) and Prohibitive Purpose (PP). Namely, AP indicates a collection of permissible purposes by which users can access for, while PP indicates a collection of prohibited purposes which not allow access for users.

5.2.3 Syntax

Following the semantics of elements of purpose-based policies on provenance, we define its syntax. The syntax is a carrier of policy models. We select a syntax which is different from the XACML proposed in Chapter 3, because the conditions for purpose-based access policies are relatively less. Therefore, we believe the applied syntax is more close to the nature of this policy model.

Generally, the policy model is tailored under the assumption that an expression of conditions (provenance partitions) maps to a collection of purposes. Namely, if a provenance contains certain provenance partitions, then it can be accessed for a collection of purposes and should be prohibited by another set of purposes.

Explanation Policy Type 1 {Provenance Partitions→AP} In one type of purpose-based access policies on provenance, it maps a tree structure of provenance partitions to a collection of allowed purposes. It indicates that if provenance partitions in a targeted provenance graph meet the access tree structure, the provenance graph can be accessed by the set of allowed purposes.

$$\mathscr{F}\ (AP, PG_i, AccessTree_j) = \{\text{AP} \mid \exists Prov_{Partition} \in PG_i\ \&\ Prov_{Partition} \subseteq AccessTree_j\ \}$$

The inputs are a targeted provenance graph and an access tree which is the policy. The access tree consists of provenance partitions as leaf nodes and operators as non-leaf nodes. The evaluation result for each provenance partition is value in a four-valued decision set. The operators which are non-leaf nodes merge values for all non-leaf nodes and generate final results for an access tree. If the result is 1_P, which implies that the provenance matches the access tree. Further, the purposes of the policy can be granted.

Explanation Policy Type 2 {Provenance Partitions→PP} This maps a combination of provenance partitions to a collection of prohibited purposes, which defines if provenance partitions in a targeted provenance graph meet the access tree structure, then the data can not be accessed for the set of prohibited purposes.

$$\mathscr{F}\ (PP, PG_i, AccessTree_j) = \{\text{PP} \mid \exists Prov_{Partition} \in PG_i\ \&\ Prov_{Partition} \subseteq AccessTree_j\ \}$$

Explanation Policy Type 3 {Provenance Partitions → AP|PP} This maps a combination of provenance partitions to a collection of allowed purposes and prohibited purposes. It grants a set of allowed purposes and a set of prohibited purposes to the request.

$$\mathscr{F}\ (AP\&PP, PG_i, AccessTree_j) =$$

$$\{\text{AP\&PP} \mid \exists Prov_{Partition} \in PG_i\ \&\ Prov_{Partition} \subseteq AccessTree_j\ \}$$

Explanation Policy Type 4 {Provenance Partitions|Data Labels→AP&PP} However, in an access policy, the input is not restricted to provenance partitions. While it may include data labels from the data and queries. It includes the subject that is the users attempt to access the data, categories of a piece of data *etc.* For the example below, when the targeted provenance graph meets the access tree, the requestor is within the list of applicable subjects, and the categories of data are within the list of applicable category, a collection of permissible and/or prohibited access purposes can be applied.

$$\mathscr{F}\ (AP\&PP, PG_i, AccessTree_j) =$$

$$\{\text{AP\&PP} \mid \exists Prov_{Partition} \in PG_i\ \&\ Prov_{Partition} \subseteq AccessTree_j$$
$$\&\exists s((s \in S_n) \wedge (r \preceq s))\ \&\exists k((k \in K_n) \wedge (K_D(i) \subseteq k))\}$$

5.2.4 Case Study

We propose a case study to implement our proposed access policies. Under the scenario illustrated in Figure 5.4, data is transferred from *System of Records A* to *System of Records B*. The provenance-based access policy determines which purposes the data can be accessed for when the data is delivered to *System of Records B*. For each piece of Data (i), there are several attributes attaching to the content of data $q = Q_D(i)$, which is shown as:

$$\text{Data } (i) = \{q = Q_D(i); k = K_D(i); g = G_D(i); p = P_D(i)\ \}$$

- Category k = K_D(i) is a tag attached to a piece of data, which is an attribute used to describe properties of the content q. For instance, if a piece of data is labeled a category of “Medical Records", it might be allowed to be used for *diagnose* purpose. Hence, the category is a condition in the policy to determine possible access purposes.
- Provenance Graph $g = G_D(i)$ is attached with data to record historical transactions performed on data. In this system, we employ OPM$^+$ which takes the form of a directed acyclic graph (DAG). In this framework, provenance is employed as conditions to determine access purposes.

- Purpose $p = P_D(i)$ lists a set of access purposes attached with Data (i), which is generated by the data producers. The final allowed purposes in the repository should be determined by both of the attached purposes and policy results.

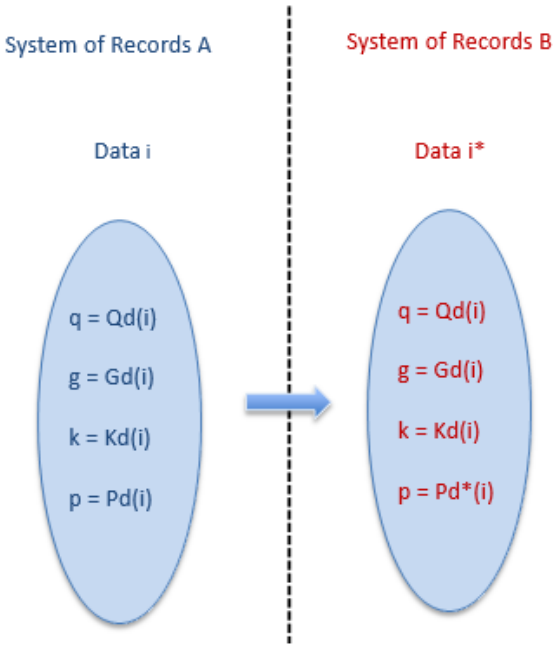

Figure 5.4: A Case Study

Each System of Records (SOR) is attached to System of Records Notices (SORN). A piece of data delivers between two SOR. Let one SOR be the source s, and the other one is the repository r. The data repository has an associated SORN $n = N(r)$ which lists rules. A rule maps tags of data including k, g, p for permitted purposes. Similarly, the source database defines rules to determine the purposes. For a piece of data i, the intersection of purposes from source, repository and $p = P_D(i)$ is the permitted usages.

Data(i) = {$Q_D(i)$; $K_D(i)$=assignment; $G_D(i)$; $P_D(i)$={education} }
Request: a user with a role of student asks to access Data(i)
< *Sample Policy from the Source*>
< *Subject* > students; teaching staff < /*Subject* >
< *Category* > assignments; exam paper < /*Category* >
< *ProvenancePartitions* >
directed provenance path= ($wasSubmittedBy|Submit, \v^*, wasGradedby|Grade$)
< /*ProvenancePartitions* >
< *AP* > education; research < /*AP* >
< *PP* > access investigation < /*PP* >
</*Sample Policy from the Source*>
< *Sample Policy from the Repository*>
< *subject* > students; teaching staff < /*subject* >
< *category* > assignments; exam paper < /*category* >
< *ProvenancePartitions* >
directed provenance path= ($wasSubmittedBy|Submit, \v^*, wasGradedby|Grade$)
< /*ProvenancePartitions* >
< /$AccessTree_{PS}$ >
< *AP* > education; analysis < /*AP* >
< *PP* > research < /*PP* >
</*Sample Policy from the Repository*>

We propose a simple example to illustrate the purpose-based access policies on provenance. Assume a user with a role of *student* asks to access Data(i). The policy from the Source determines a set of allowed and prohibited purposes. Under this scenario, the users are allowed to access Data(i) for the purposes of *education* & *research*. Similarly, the Repository also decides that Data(i) can be accessed for *education* & *analysis* but not *research*. Finally, the intersection for purposes determined by Source, Repository and $P_D(i)$ is *education*. Hence, Data(i) can only be accessed for *education* for the request.

5.3 Internal Policies Algebra

For purpose-based access policies in provenance-aware systems, *internal policy algebras* are required to combine purpose sets generated by individual policies within a system. While

external policy algebras integrate results of policies from different parties. Firstly, we explore the demands for internal policy algebras.

Under the system assumption of this framework, the sensitivity levels of purposes can be identifiable, because the purposes in policies are defined for the shared purpose DAGs in a system, where the hierarchies of each purpose are initially defined. Hence, it is possible to identify the hierarchy of each purpose within a system and further distinctively process purposes in different hierarchies. Therefore, for internal algebras, we create functions to process purposes with various distinct hierarchies.

In terms of the motivation, for purposes in DAG are tailored hierarchically, purposes are usually displayed from generalisation to specialisation in the graphs, based on various sensitivities. Semantically, under most scenarios, general purposes are less sensitive and are more easily to grant the permission. While specific purposes are more sensitive and should be cautious to grant the permission. Consequently, purposes in different hierarchies are merged with different operators. For example, it enforces maximum privilege (by *Addition* (+)) for general usage, and minimum privilege (by *Conjunction* (&)) for more specific usages.

Therefore, in this section, we firstly propose basic operations to merge purposes sets. Then, functions that are constructed by basic operations are defined, in order to propose an approach to merge purposes in distinctly different hierarchies.

5.3.1 Basic Operators

Basic operators are elements of constructed functions for policy algebras, and they are classified as *symmetric operators* and *asymmetric operators*. Specifically, for a symmetric operator, when the two expressions besides it switch, the result does not change. Namely, let $*$ be an asymmetric operator, S_1 and S_2 be two sets of purposes. $S_1 * S_2 = S_2 * S_1$, while, when the sequence of two expressions beside a symmetric operator switch, the result changes. Hence, let $\bullet$ be a symmetric operator, $S_1 * S_2 \neq S_2 * S_1$. Below, we define 5 asymmetric operators and 1 symmetric operators, which are tailored to merge purpose sets.

- Union (+). Addition of the sets S_1 and S_2 results in a combined set S_I, in which it concludes all the data items from both of the sets. It merges two sets of purposes by

returning their union and returns the maximum scope of the purposes of both. This is shown as:

$$S_I = [\![S_1 + S_2]\!] = [\![S_1]\!] \cup [\![S_2]\!]$$

- Intersection (&). Intersection of the sets S_1 and S_2 returns a set of data items, in which it contains the data items existing at both sets. Namely, it emerges two sets by returning their intersection. Intuitively, intersection returns purposes that in both purpose sets. This can be shown as:

$$S_I = [\![S_1 \& S_2]\!] = [\![S_1]\!] \cap [\![S_2]\!]$$

- Difference ($\boxminus$). The difference of sets S_1 and S_2 returns data items which only appear in one set but not both. Namely, it returns a set of purposes which are agreed by one policy but not agreed by both.

$$S_I = [\![S_1 \boxminus S_2]\!] = [\![S_1 + S_2]\!] - [\![S_1 \& S_2]\!]$$

- Precedence ($\uparrow \triangle$). It compares the hierarchies of data items with highest hierarchies (the top purposes) in both sets and outputs only one set with the higher top purpose. Particularly, we define two functions to return the highest hierarchy of purposes in the set as Max (S_m, S_n). Similarly, Min (S_m, S_n) returns the lowest hierarchy of purposes in the set.

Operator	Semantics $[\![]\!]_e$	symmetry
$S_1 + S_2$	$[\![S_1]\!]_e \cup [\![S_2]\!]_e$	$S_1 + S_2 = S_1 + S_2$ (symmetric)
$S_1 \& S_2$	$[\![S_1]\!]_e \cap [\![S_2]\!]_e$	$S_1 \& S_2 = S_1 \& S_2$ (symmetric)
$S_1 \boxminus S_2$	$[\![S_1 + S_2]\!] - [\![S_1 \& S_2]\!]$	$S_1 \boxminus S_2 = S_1 \boxminus S_2$ (symmetric)
$S_1 \uparrow \triangle S_2$	S_i\{I=maximum(Max(S_1),Max(S_2))\}	$S_1 \uparrow \triangle S_2 = S_1 \uparrow \triangle S_2$ (symmetric)
$S_1 \downarrow \triangle S_2$	S_i\{I=minimum(Max(S_1),Max(S_2))\}	$S_1 \downarrow \triangle S_2 = S_1 \downarrow \triangle S_2$ (symmetric)
$S_1 \uparrow \triangledown S_2$	S_i\{I=maximum(Min(S_1),Min(S_2))\}	$S_1 \uparrow \triangledown S_2 = S_1 \uparrow \triangledown S_2$ (symmetric)
$S_1 \downarrow \triangledown S_2$	S_i\{I=minimum(Min(S_1),Min(S_2))\}	$S_1 \downarrow \triangledown S_2 = S_1 \downarrow \triangledown S_2$ (symmetric)
$S_1 - S_2$	S_1 - $[\![S_1 \& S_2]\!]$	$S_1 - S_2 \neq S_1 + S_2$ (asymmetric)

Table 5.1: Schema of Basic Operators

$$S_I = [\![\ S_1 \uparrow \triangle S_2\ ]\!]\ (\mathrm{I} \in \{1,2\};\ \mathrm{I} = \text{maximum}\ \{\mathrm{Max}(S_i), \mathrm{Max}(S_i)\})$$

- Precedence ($\downarrow \triangle$). This compares the hierarchies of data items with the highest hierarchies in both sets and outputs only one set with the higher top purpose.

$$S_I = [\![\ S_1 \downarrow \triangle S_2\ ]\!]\ (\mathrm{I} \in \{1,2\};\ \mathrm{I} = \text{minimum}\ \{\mathrm{Max}(S_i), \mathrm{Max}(S_i)\})$$

- Precedence ($\uparrow \triangledown$). This compares the hierarchies of data items with the lowest hierarchies (the bottom purpose) in both sets, and outputs only one set with the higher bottom purpose.

$$S_I = [\![\ S_1 \uparrow \triangledown S_2\ ]\!]\ (\mathrm{I} \in \{1,2\};\ \mathrm{I} = \text{maximum}\ \{\mathrm{Min}(S_i), \mathrm{Min}(S_i)\})$$

- Precedence ($\downarrow \triangledown$). It compares the hierarchies of data items with the lowest hierarchies in both sets, and outputs only one set with the lower bottom purpose.

$$S_I = [\![\ S_1 \downarrow \triangledown S_2\ ]\!]\ (\mathrm{I} \in \{1,2\};\ \mathrm{I} = \text{minimum}\ \{\mathrm{Min}(S_i), \mathrm{Min}(S_i)\})$$

- Subtraction (-). Subtraction of the sets S_1 and S_2 returns the items in S_1 subtracting those appear in S_2, which is an asymmetric operator. Namely, when two sets shift positions beside the operator of *subtraction*, the result changes.

$$S_I = [\![\ S_1 - S_2\ ]\!] = S_1 \text{ - } [\![\ S_1 \& S_2\ ]\!]$$

$$S_I = [\![\ S_2 - S_1\ ]\!] = S_2 \text{ - } [\![\ S_1 \& S_2\ ]\!]$$

5.3.2 Functions for Internal Policy Algebras

Based on the basic data algebra operators, we define functions to process hierarchical purpose sets. In terms of the motivation, we assume the sensitivity of purposes under a DAG structure is hierarchical. Specifically, the more specific usages should be regarded as the more sensitive ones to be accessed. Therefore, data algebras tend to combine two sets of hierarchical purposes by various operators. Specifically, when combining two sets of hierarchical purposes, we would like to conjunct lower-hierarchy purposes and take the

intersection of higher-hierarchy purposes to generate a combined set.

Firstly of all, we propose two approaches for dividing a set of purposes as a higher hierarchical subset and a lower hierarchical subset. One approach is that each hierarchy of purpose is defined for the purpose DAG. Namely, it defines which collection of purposes within a system is higher hierarchically and which collection of purposes is lower hierarchically. Two sample purpose sets from the graph in Figure 5.3 are presented below to illustrate this. High hierarchical (HH) purposes and low hierarchical (LH) purposes are defined along with the generation of purpose DAGs, where the elements above the dashed line are lower hierarchy purposes and purposes below the dashed line are higher hierarchy purposes. The two sample purpose sets are:

$Record^{\updownarrow^1_2}$= {Admin, Record, Analysis, Service-Maintain, Service-offers}= {Admin, Record }$_{HH}$ + {Analysis, Service-Maintain, Service-offers}$_{LH}$

$Marketing^{\updownarrow^1_4}$= {General Purpose, Marketing, Direct-use, D-Email, D-Phone, Service-offers, Service-Updates}= {Admin, Record}$_{HH}$ + {Analysis, Service-Maintain, Service-offers}$_{LH}$

Under the default definition of the Purpose DAG, the purposes of the high hierarchy in $Record^{\updownarrow^1_2}$ is {Admin, Record}, while the low hierarchy purposes are {Analysis, Service-Maintain, Service-offers}. Similarly, the high and low hierarchical sets in $Marketing^{\updownarrow^1_4}$ are {Admin, Record} and {Analysis, Service-Maintain, Service-offers} respectively.

For the other approach to dividing purpose sets, it is determined by the *central purposes* of the sets. The *central purpose* is the keyword to define a set of purposes. In the following examples. *Record* and *Education* are the central purposes. Specifically, when two purpose sets are merged, the central purposes are compared. The central purpose of higher hierarchy determines the position of the parting line, where the row it stays at and below are the high hierarchical purposes. The rest are the low hierarchical purposes. To illustrate this, we still pick up two sample purpose sets from the graph in Figure 5.3.

$Record^{\updownarrow_2^1}$ = {Analysis, Record, Admin, General Purpose} = {Admin, Record }$_{HH}$ + {Analysis, Service-Maintain, Service-offers}$_{LH}$

$Education^{\updownarrow}$ = {Optimise, AI, Research, Study, Eduction, General Purpose} = {Optimise, AI, Research }$_{HH}$ + {Study, Eduction, General Purpose}$_{LH}$

To classify each purpose set as a high hierarchical set and a low hierarchical set, the central purposes which are *Record* and *Education* should be compared first. The purposes are displayed from top to bottom according to the order as from generalised purposes to specific purposes, so Hierarchy (Record) > Hierarchy (Education). The row where *Record* stays at and the those below are high hierarchical purposes, and the rest are low hierarchical purposes. Therefore, the high hierarchical collections for each set are {Admin, Record }$_{HH}$ and {Optimise, AI, Research }$_{HH}$. All the rest are purposes of the low hierarchy.

After dividing purposes of each set as two collections based on their hierarchies, we define the functions to merge purpose sets. Let HA_i represents high hierarchy allowed purposes, HP_i represents high hierarchy prohibited purposes, LA_i represents low allowed purposes, and LP_i represents low prohibited purposes, the u-ary operators are defined as below:

$$f_{\oplus}(S_i, S_j) = S_i \oplus S_j = \frac{(HA_i \& HA_j) - (HP_i - HP_j)}{(LA_i + LA_j) - (LP_i - LP_j)}$$

$$f_{\ominus}(S_i, S_j) = S_i \ominus S_j = \frac{(HA_i \& HA_j) - (HP_i \& HP_j)}{(LA_i + LA_j) - (LP_i \& LP_j)}$$

$$f_{\otimes}(S_i, S_j) = S_i \otimes S_j = \frac{(HA_i \& HA_j) - (HP_i - HP_j)}{(LA_i + LA_j) - (LP_i \& LP_j)}$$

$$f_{\oslash}(S_i, S_j) = S_i \oslash S_j = \frac{(HA_i \& HA_j) - (HP_i \& HP_j)}{(LA_i + LA_j) - (LP_i - LP_j)}$$

$$f_{\odot}(S_i, S_j) = S_i \odot S_j = \frac{(HA_i + HA_j) - (HP_i - HP_j)}{(LA_i + LA_j) - (LP_i \& LP_j)}$$

$$f_{\uplus}(S_i, S_j) = S_i \uplus S_j = \frac{(HA_i + HA_j) - (HP_i - HP_j)}{(LA_i + LA_j) - (LP_i \& LP_j)}$$

$$f_{\dot{+}}(S_i, S_j) = S_i \dot{+} S_j = \frac{(HA_i + HA_j) - (HP_i \& HP_j)}{(LA_i + LA_j) - (LP_i \& LP_j)}$$

$$f_{\Cap}(S_i, S_j) = S_i \Cap S_j = \frac{(HA_i + HA_j) - (HP_i \& HP_j)}{(LA_i \& LA_j) - (LP_i \& LP_j)}$$

$$f_{\Cup}(S_i, S_j) = S_i \Cup S_j = \frac{(HA_i + HA_j) - (HP_i \& HP_j)}{(LA_i \& LA_j) - (LP_i - LP_j)}$$

$$f_{\boxtimes}(S_i, S_j) = S_i \boxtimes S_j = \frac{(HA_i \boxminus HA_j) - (HP_i - HP_j)}{(LA_i \boxminus LA_j) - (LP_i \& LP_j)}$$

$$f_{\boxdot}(S_i, S_j) = S_i \boxdot S_j = \frac{(HA_i \boxminus HA_j) - (HP_i - HP_i)}{(LA_i + LA_j) - (LP_i \& LP_j)}$$

$$f_{\boxplus}(S_i, S_j) = S_i \boxplus S_j = \frac{(HA_i + HA_j) - (HP_i - HP_j)}{(LA_i \boxminus LA_j) - (LP_i \& LP_j)}$$

$$f_{\divideontimes}(S_i, S_j) = S_i \divideontimes S_j = \frac{(HA_i \boxminus HA_j) - (HP_i - HP_j)}{(LA_i \& LA_j) - (LP_i \& LP_j)}$$

The functions merge purposes in different hierarchies with different operators. These functions are utilised for internal algebras of this framework. In terms of which functions should be used and how to combine functions as an expression, this should be determined by each system's default settings or the policies. The integration of policies may involve multiple operators, hence we define the concept of FIDA expressions.

DEFINITION 9: A FIDA expression is defined as:

-If S is a set, then S is a FIDA expression;

-If S_1 and S_2 are FIDA expressions, so is a function $f\ (S_i, S_j)$.

-If function $f_\alpha(S_i, S_j)$ is a FIDA expression, $f_\beta(S_m, S_n)$ is another FIDA expression, so are $f_\alpha(S_i, S_j) \bullet f_\beta(S_m, S_n)$, where $\bullet$ is a function.

In an expression, *Intersection* and *Precedence* operators take high priorities, while Addition and Subtraction take low priorities. For the same priority operators, calculations should be run from left to right. For example, $S_1 \& S_2 + S_3 \triangleright S_4$ is interpreted as $(S_1 \& S_2) + (S_3 \triangleright S_4)$.

A function could also merge more than two purpose sets. A FIDA expression can be defined as in the example below:

$$f(S_1, S_2, \ldots S_n) = \frac{\sum_{i=1}^{n} HA_i - \sum_{i=1}^{n} HP_i}{\boxminus_{i=1}^{n} LA_i - \&_{i=1}^{n} LP_i}$$

5.4 External Policy Algebra

External policy algebras are tailored to merge policy results from different parties. Let us imagine the situation where a piece of data was delivered from a source to a recipient. The source, recipient and even data owners generate access policies for the piece of data, after which we propose external policy algebras to merge results from various parties, in order to generate a final intended purpose for the data.

The algebras can be generated by a server viewed as a third party or by the recipient. However, as policies from different authorities cannot be generated from uninformed purpose DAGs, the hierarchies of purposes are not based on the same criteria anymore. Hence, we do not distinguish the hierarchies of purposes when sets of purposes are merged.

Following, based on basic operators defined in the previous section, we define functions for external policy algebra, where m and n are two parties. The functions are defined to merge results from two parties. Further, an expression consisting of several functions could merge various parties flexibly. The functions to merge purpose sets from multi-parties are defined below, where AP represents allowed purposes and PP represent prohibited purposes. AP_m is the allowed purposes for S_m; PP_m is the prohibited purposes for S_m; AP_n is the allowed purposes for S_n; PP_n is the prohibited purposes for S_n.

$$F_1(S_m, S_n) = (AP_m + AP_n) - (PP_m \& PP_n)$$

$$F_2(S_m, S_n) = (AP_m + AP_n) - (PP_m - PP_n)$$

$$F_3(S_m, S_n) = (AP_m \& AP_n) - (PP_m \& PP_n)$$

$$F_4(S_m, S_n) = (AP_m \& AP_n) - (PP_m - PP_n)$$

$$F_5(S_m, S_n) = (AP_m \boxminus AP_n) - (PP_m \triangleleft PP_n)$$

$$F_6(S_m, S_n) = (AP_m \boxminus AP_n) - (PP_m \triangleright PP_n)$$

$$F_7(S_m, S_n) = (AP_m \triangle AP_n) - (PP_m \boxminus PP_n)$$

$$F_8(S_m, S_n) = (AP_m \triangledown AP_n) - (PP_m \& PP_n)$$

The functions for external algebras generally produce intended purposes as $IP = AP$ - PP. However, the basic operators to merge the allowed purposes AP and prohibited purposes

PP are different. Such as F_1 combines allowed purposes by *Addition* (+), while it takes the intersection (&) for prohibited purposes.

Moreover, an expression to merge purpose sets from more than two parties could consist of a set of external policy functions. This expression is defined as:

DEFINITION 10: A FIDA expression for the functions of external algebra:

-If S is a set, then S is a FIDA expression;

-If S_1 and S_2 are FIDA expressions, so is a function F (S_i, S_j).

-If function $F_\alpha(S_i, S_j)$ is a FIDA expression, $F_\beta(S_m, S_n)$ is another FIDA expression, so are $F_\alpha(S_i, S_j) \bullet F_\beta(S_m, S_n)$, where $\bullet$ is a function.

5.5 Evaluation

We did experiments to evaluate the performance of the proposed policies and data algebras. We implement the process of policy generation and internal policy algebras and external algebras. The purpose hierarchy has been taken into consideration for the implementation.

We process the experiments on a 3.40 GHz Intel Computer with 16GB of memory. The machine's operating system is Microsoft Windows 8 and the database is Oracle Database. We set up synthetic datasets based on the version proposed by Wisconsin Benchmark [107]. To be more specific, every database consists of 3 numeric and 6 string columns, we set up data values refer to the specification from the paper [107]. The provenance model is OPM$^+$ proposed in Chapter 3, and 200 purposes were generated. Particularly, in the policy generation phrase, the 200 purposes are randomly selected to attach to policies.

After establishing provenance graphs and purposes, we tested the time span for the policy generation. The aim to do this experiment is to compare the time costs for each type of policies. The more complicated of a policy contains more restrictions, the longer time requires generating the policy. To measure the time of generating a policy, it counts from generating random provenance subgraphs, then combing with conditions and access purposes randomly. Thus, the reported time takes the whole policy generating process into account. We repeated the process ten times, and take the average time to draw the figure. Following,

we implemented the policy algebras proposed in this chapter as well. We compare the time span for internal policy algebras and external policy algebras. The experiment also records the average values of ten times of simulation. Notably, the internal policy algebras take longer time compared with the external policy algebras, because in our framework, only the internal policy algebras distinguish the hierarchies of purposes. Hence, internal policy algebras calling more functions take longer time than external policy algebras in the implementation.

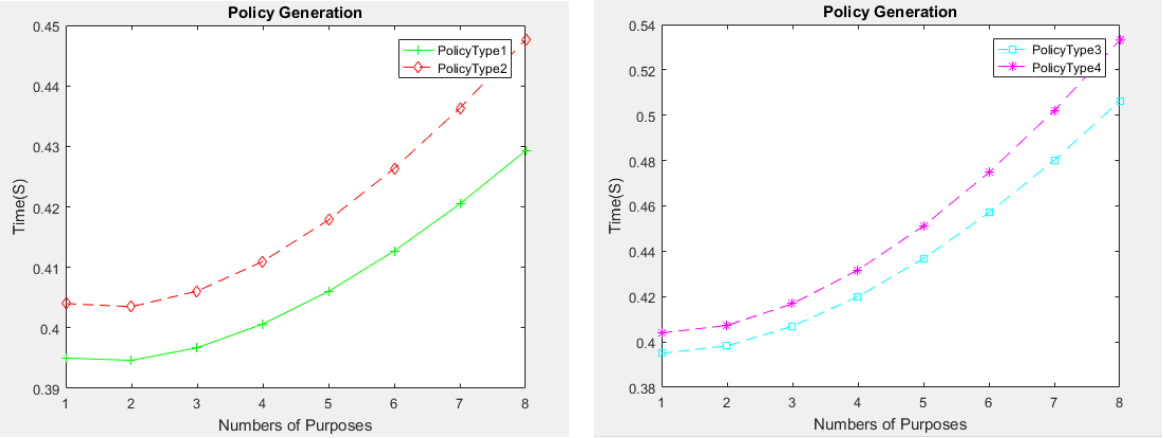

Figure 5.5: Experiment

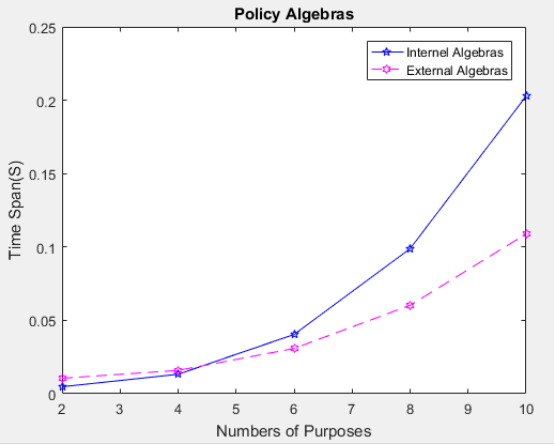

Figure 5.6: A Case Study

5.6 Conclusion and Future work

In this chapter, purpose-based access policies on provenance are proposed, in order to establish a comprehensive scope of access control mechanisms on provenance. First, we define the policy model including syntax and semantics, which maps attributes in provenance

to allowed and prohibited purpose sets. The policies determine which purposes a piece of data can be accessed for, and this is based on whether the provenance contains certain provenance partitions. Moreover, as the sensitivities of purposes are different, the purposes are classified as various hierarchies in this framework.

We define internal and external policy algebras for purpose-based access policies. To the best of our knowledge, we define the functions that merge purposes in various hierarchies by different basic operators for the first time. Moreover, external policy algebras to merge policies from multi-parties are also created.

Fan X., Varadharajan V., Hitchens M. (2015) Provenance Based Classification Access Policy System Based on Encrypted Search for Cloud Data Storage. In: Lopez J., Mitchell C. (eds) Information Security. ISC 2015. Lecture Notes in Computer Science, vol 9290. Springer, Cham

DOI: https://doi.org/10.1007/978-3-319-23318-5_16

7

Provenance-based Encryption Scheme for Fine-grained Access Control

In this chapter, we propose an encryption scheme to implement provenance-based access control policies. The scheme we proposed is based on an existing attribute-based encryption scheme. The goal of the proposed technique is that the scheme could generate keys by a fast approach when the access tree structures share similar conditions.

7.1 Introduction

Our proposed scheme could be utilised for the application of Cloud Computing. As a new emerging computing paradigm, Cloud Computing merges techniques of utility computing, virtual and distributed systems and service-oriented architecture [124]. In the last a few years, cloud computing research has raised a tide of academical and industrial interest. It has been viewed as the fifth utility[125] after water, gas, electricity, and telecommunication and

holds optimistic prospects for development. Cloud Servers provides centralised outsourced services to reduce the cost of computing, data management, and also enhances efficiencies and flexibility. Cloud computing models have been classified as, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS), the three main types. As a result of its benefits, cloud computing has been widely utilised in a market, such as web-based email (Microsoft, Google, Yahoo *etc.*), Dropbox, IBM's Blue Cloud and so on. In the market, cloud computing applications have proven to bring abundant profit for service providers and cloud customers.

However, security and privacy concerns are some of the main obstacles to the development and extensive utilisation of cloud computing. On the one hand, cloud servers are not entirely trusted never to embezzle users' documents to strive for their profit; on the other hand, as large-scale and diversified data processing, adequate security protection against malicious attackers is a challenge. Techniques involving preservation of security in the cloud still need considerable improvement.

Aside from preserving confidentiality, efficient fine-granted access management is another requirement for service-oriented cloud computing. Servers are expected to identify authentications accurately and also to permit access to service to eligible users and customers and block unauthorised access. For instance, for a company's data storage in the cloud, the access rights to documents/data are defined by access policies. Based on the policies, only nominated senior managers and leaders can read some sensitive commercial strategies. Effective access control not only supports information and service delivery but also defends from attacks.

7.1.1 Our Contributions

In this chapter, we propose a Provenance-based Encryption (PBE) scheme for fine-grained access control. PBE extends from ciphertext policy attribute-based access control (CP-ABE)[21], which implements provenance-based access control policies.

We propose an assumption to implement provenance-based access control policies in the Cloud. In the Cloud, a file could be generated by several data owners. Each data owner generates an access control policy for the section of data they contribute. As the access control policies are based on the provenance, PBE is able to generate keys and encrypt sections of a

file according to the provenance-based access control policies.

In terms of the motivation for this proposed work, the calculation for a large number of public keys for encrypting data is time-consuming. There is an approach that is able to generate the keys much more efficiently when the conditions of access control policies exist overlapping. Under provenance-based access control policy model, since the conditions of a large number of policies are selected from a provenance graph attaching with the data. Even though the conditions for each policy might be different, there might exist overlapping of conditions among policies. Another reason to result in the overlapping is that the sections of data produced by different owners shared the same history/provenance. Thus, our proposed technique performs well to implement provenance-based access control policies by improving the efficiency of algorithms.

To be more specific, we propose an approach to calculate keys for encryption when a tree of access structure grows from T to T'. This is more efficient than running the encryption according to T' from scratch, since all the components in the old ciphertext corresponding to the leaf nodes can be reused.

7.1.2 Chapter Organisation

The organisation of the chapter is shown as follows.

- Related work on attribute-based encryption and provenance-based access control is presented in Section 7.2.
- In Section 7.3, we present the system model in the cloud and the security assumptions.
- In Section 7.4, the Open Provenance Model and our provenance-based access control policy are illustrated.
- Then, followed by a preliminary in Section 7.5, the algorithms in PBE are shown in Section 7.6.
- We prove the security of PBE and discuss its performance in Section 7.7.
- In the end, we conclude the chapter in Section 7.8.

7.2 Related Work

Here, we review attribute-based encryption (ABE) schemes. Initially, the idea of ABE was introduced by Sahai and Waters [126] as a new type of Identity-Based Encryption (IBE) scheme which is called Fuzzy Identify-Based Encryption in this chapter. In Fuzzy IBE, a set of attributes is viewed as an identity, and the ciphertext is encrypted by an identity $\mathcal{W}$, while each identity is associated with a private key. Hence, a private key for an identify $\mathcal{W}'$ can decrypt the ciphertext encrypted by $\mathcal{W}$, only if the identities $\mathcal{W}$ and $\mathcal{W}'$ are close to each other, and it leaves a few open problems such as missing of expressibility of its threshold.

In ABE, private keys and ciphertext are associated with sets of attributes or access structures over attributes. If and only if private keys match encrypted attributes, then the ciphertext can be decrypted. Based on either attributes or policies over access structures associated with private keys, ABE schemes are classified as key-policy attribute-based encryption (KP-ABE) and Ciphertext-policy attribute-based encryption (CP-ABE).

In 2006, an original KP-ABE was presented by Goyal *et al.*[103]. In this scheme, a private key is generated by a monotonic access tree structure, while a ciphertext is encrypted by a set of attributes. Only if the encrypted attributes match the private keys, can the ciphertext be decrypted? On the contrary, in CP-ABE[21] the associations with ciphertext and private keys are switched. Namely, the access policies are utilised to encrypt, and private keys are generated by attributes. As a result, CP-ABE is seen to be conceptually closer to classic access control models.

However, these classic schemes cannot satisfy various access control applications. Some extended schemes are proposed. An ABE encryption scheme linked with non-monotonic access structure is presented by Ostrovsky *et al.* in 2007[127]. In this scheme, the difference from the previous schemes is that any type of access formula over attributes can be expressed, as the monotonic access structure was moved to non-monotonic access structure. The algorithm was achieved by introducing an approach of revocation.

Subsequently, Wang *et al.* provided a hierarchical attribute-based encryption scheme [128] which combines CP-ABE and an identity-based encryption scheme[123]. The system for the scheme consists of four parties: root master, domain masters, data owners and users. The private keys are generated hierarchically: the root master generates first level keys and distributes to domain masters; then domain masters use domain master keys to generate secret

keys for authorised users. This scheme achieves full delegation and revocation and it was proven to be semantically secure. However, it is difficult to implement, as it does not organise the relationship well between attributes and domain authorities. Namely, one attribute might be managed by multiple domain authorities.

Bobby *et al.* proposed a ciphertext-policy attribute-set-based encryption (ASBE) scheme [129] to support efficiency and flexibility in regulating attributes of users. In CP-ABE, attributes are organised as a single set. ASBE extends it by introducing a recursive set structure. It sets different values to attributes, which can also address revocation issues. However, a delegation algorithm is missing from the work. Based on this paper, Wan *et al.*[124] introduced a Hierarchical attribute-set-based encryption (HASBE) in Cloud Computing. Similar to the system model with ASBE, this scheme achieves flexible attribute set combinations and revocation of users. HASBE generates hierarchical keys for domain managers and authorised users.

7.3 System Architecture and the Policies

7.3.1 System Architecutre

First of all, we propose a system assumption under which PBE performs. The system consists of four parties which are *a cloud server provider*, *data producers*, *data consumers* and *a trusted authority*, illustrated in Figure 7.1 below.

The cloud service provider manages files stored in the Cloud. It stores ciphertext from data producers and allows data consumers to download ciphertext based on their own interest. In this system, we assume that a file might be generated by multiple producers. For instance, a doctor generates a case file for a patient. The case file could be continuing supplemented by the diagnosis from doctors and nursing reports from nurses. In order to preserve the confidentiality of the files, they are encrypted to store in the Cloud. However, the security sensitivity of the sections generated by different entities might be different. The reports from nurses could be reviewed by both doctors and nurses, but the doctors' diagnosis can only be accessed by doctors. It indicates that the access control policies for each section are generated based on the provenance.

The trusted authority grants keys to data consumers based on their credentials, only when the private keys match the corresponding access structures, can the users decrypt the ciphertext. Data users download ciphertext of their interest and utilise their keys to decrypt the files. The ciphertext can be encrypted by access control policies, where the encryption keys are various combinations of attributes. The more corresponding access structures a user can satisfy, the more sections can the user decrypt.

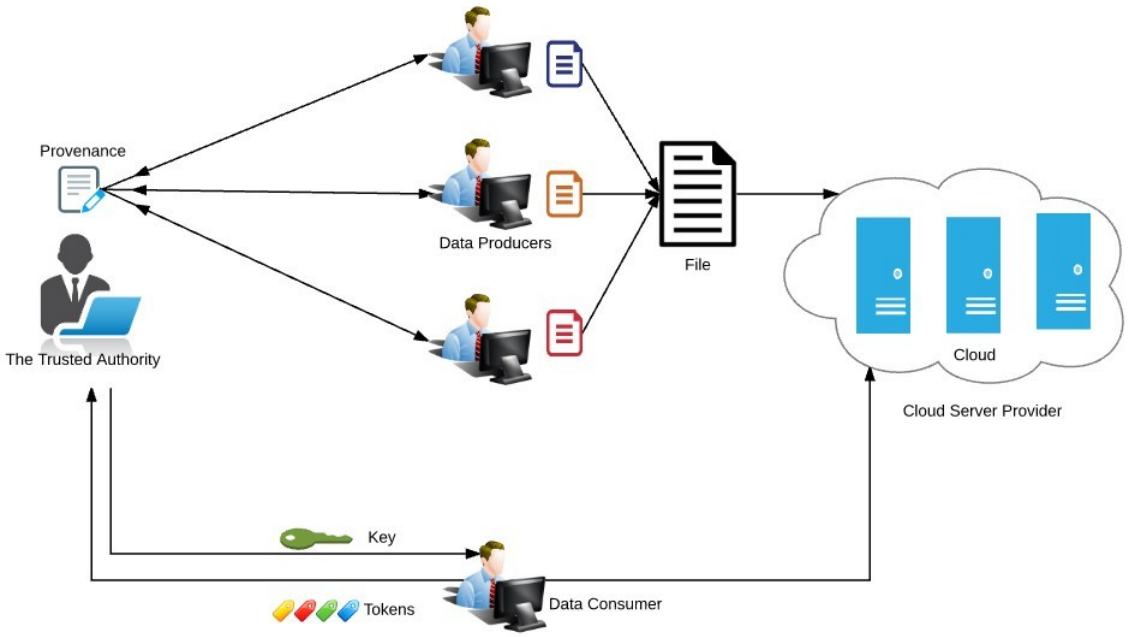

Figure 7.1: The System Architecture

7.3.2 Threat Model

The cloud provider is assumed untrustworthy and might be compromised by with malicious users attempting to obtain the content of stored files. Hence, the files sent to the cloud are encrypted against attack. A producer generates a piece of data and encrypts it by a set of attributes under a structure defined by itself. The attributes could be roles of users, features of users and processes or dependency paths from provenance.

Each user generates provenance-based access control policies for the section of data they generated. These restrictions or conditions for different policies could be similar. Just as the example shown in Figure 7.2, the data owner generates a section of data M and they tailored a policy for accessing M. While another data owner generates another piece of data M' and developed the access control policy as the Figure 7.3. The example shows that access

control policies for different sections can be different but might share the same conditions. Specifically, to decrypt a ciphertext a user has to provide the following credentials: {“Faculty of Science" || “Faculty of Engineering"; “Female" & “>25 years old" & “Driving Licence"; “A permit to read files been graded"}. Following, the access control structure for M′ grows which requires the users to provide one more credential (IEEE members) in order to decrypt the section.

There are some security concerns under this system. Some users will also try to access the files beyond their privileges. For example, a student may want to obtain the answers of exams in advance and boosting its performance. To do so, they may collude with other students, or even with the server. Moreover, we assume all the parties in our system is preloaded with secret/public key pairs, and traditional challenge-response protocols can carry out entity authentication.

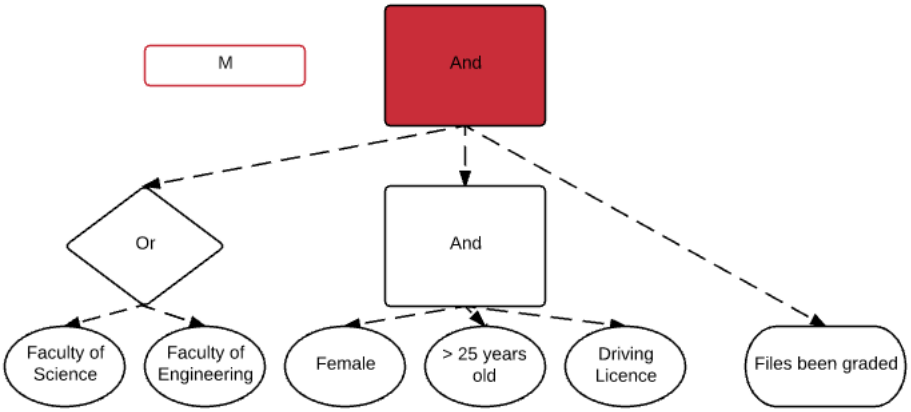

Figure 7.2: Example Policy Structures A

From the example, even though the pieces of data might be encrypted by different access structures, the access structures are relevant to each other. To be more specific, when a user’s credentials satisfy the access structure for a section of ciphertext of a file, another piece of ciphertext might require a user to provide one or more credentials to decrypt. In other words, access structures for two sections in a file are relevant, and one access tree might be the subtree for another access tree.

Our proposed scheme provides an approach to fast generate keys for the related access control structures. Thus, a PBE scheme enables encryption plaintext based on provenance-based access control policies to be fast and agile.

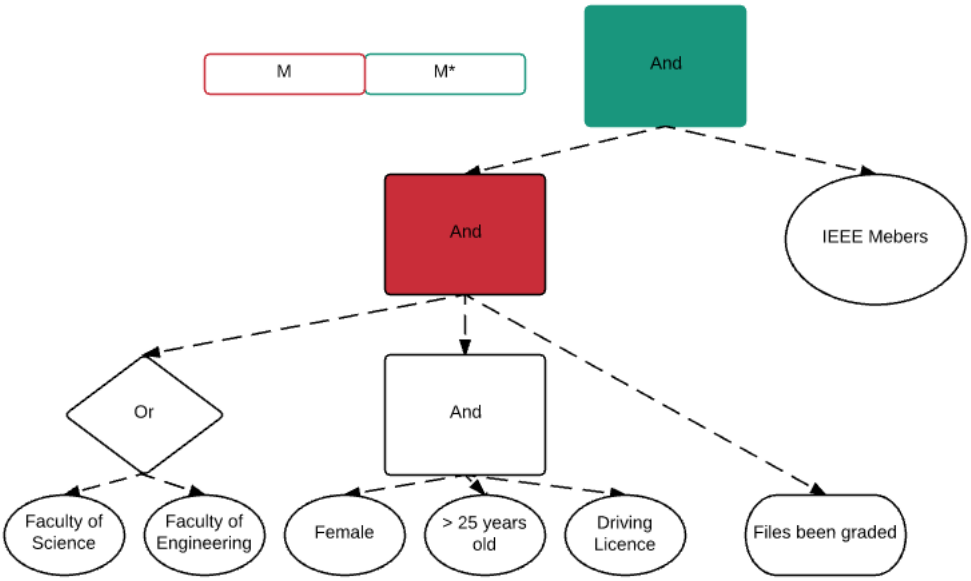

Figure 7.3: Example Policy Structures B

7.3.3 Provenance-based Access Control Policy

The provenance-based access control policies are defined by encryptors who are also the producers of data. The policies are organised under a tree structure which is illustrated in detail as follows. Data consumers could access a file when their credentials meet the access control policies. Notably, in our provenance-based access policies, we employ provenance partitions as access control conditions. The example provenance partitions could be: {“Graded by professors"; “Submitted by a doctor"; “Generated by Microsoft Company"}. These provenance partitions are recorded in provenance graphs.

7.4 Provenance-based Partitioned Encryption Scheme

7.4.1 Algorithms

Our proposed PBE includes four basic algorithms: Setup, Encrypt, EncryptPlus, KeyGen, and Decrypt. We define each algorithm as follow:

Setup. The setup algorithm takes the implicit security parameter as input and outputs the public parameters PK and a master key MK. It keeps MK hidden.

Encrypt (PK, M, $\mathcal{T}$). This algorithm encrypts data with public parameters PK and the corresponding access tree which consists of a set of attributes. It encrypts plaintext and produces ciphertext. Only users who own attributes to satisfy its access trees could decrypt the message.

EncryptPlus (PK, $M^*, Y, \mathcal{T}^*$). It develops access trees and generates new roots to encrypt plaintext. This process could be repeated.

Key Generation (MK, S). The key generation algorithm generates private keys by taking master keys MK and a set of attributes S as input. The private keys will be sent to users to decrypt messages.

Decrypt (PK, CT, SK). The inputs of a decrypted algorithm are ciphertext; public parameters PK with access trees and private keys SK, which are a set of attributes. If users' attributes satisfy one or more access trees, and they can decrypt the corresponding sections of data.

7.4.2 PBE Scheme

Our PBE scheme develops from ABE schemes to implement provenance-based access control policies. Under the system assumes that each data owner generates an access control policy for the section of data she contributes. As the access control policies are based on the provenance, the access control structures for the sections in a file are related. PBE is able to generate keys and encrypt each section of a file according to the provenance-based access control policies.

Concretely, we propose an approach to calculate keys for encryption when a tree of access structure grows from T to T'. This is more efficient than running the encryption according to T' from scratch, since all the components in the old ciphertext corresponding to the leaf nodes can be reused.

Particularly, in order to allow the method to work, we assume the root s=$q_R(0)$ of the old tree T must be known by the producer that will grow the tree to T' and generate a new root for the new tree. If the producer is the same encryptor for the old ciphertext, obviously, she knows the values. However, if the producer is different from the one who created the old ciphertext, then the producer must obtain the value of $q_R(0)$ from the previous producer/encryptor.

Setup Initially, the setup algorithm will select a bilinear group G_0 of prime order p with generator g. Then it will randomly choose two exponents $\{\alpha, \beta\} \in Z_p$. It keeps the master key MK (β, g^{α}) secreatly and generates public keys as:

$$\{\text{PK} = G_0, g, \text{h}=g^{\beta}, \text{f}=g^{1/\beta}, e(g,g)^{\alpha}\}$$

Encrypt(PK, M, $\mathcal{T}$). The encryption algorithm encrypts a message $\mathcal{M}$ under the tree access structure $\mathcal{T}$. Firstly, the algorithm chooses a polynomial q_x for each node x in the tree $\mathcal{T}$ in a top-down manner. Starting with the root node R, the algorithm chooses a random s $\in Z_p$ and sets $q_R(0)$ = s. For each node x in the tree $\mathcal{T}$, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is, $d_x = k_x$ - 1. It sets $q_x(0) = q_{parent(x)}$ (index(x)) and chooses d_x other points randomly to define q_x. Let, Y be the set of leaf nodes in $\mathcal{T}$. The ciphertext is then constructed by giving the tree access structure $\mathcal{T}$ and computing

$$CT = (\mathcal{T}, \tilde{C} = M_1 e(g,g)^{\alpha s}, C = h^s, \forall\ y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)})$$

EncryptPlus(PK, M^*, Y, $\mathcal{T}^*$). When a producer adds a piece of data message M^* to the original message M, if the confidentiality of M^* is more sensitive than M, the producer defines that data consumers have to provide a few more attributes than the original combination of attributes for decrypting M^*.

The producer develops the access tree and generates a new root node, namely, the previous tree is a subtree of the new tree. Firstly, s=$q_R(0)$ was shared with the producer to develop the access tree. These new polynomials are generated in a down-top-down manner: it starts with the node connecting to the previous root, in a down-top manner to the new root; then from the new root, generating polynomials for the rest nodes in a top-down manner. It stops when all

the polynomials are generated. Each polynomial for new nodes is generated as: the degree d_x for new nodes' polynomial q_x is still, d_x=q_x-1, setting coefficient and constant as unknowns. The polynomials should satisfy $q_x(0) = q_{parent(x)}$ (index(x)). Hence, this polynomial gives an equation of the unknowns. In other words, the selected random numbers have to satisfy the equation. Then, each new node is generated one by one. When the polynomial ($q_R(0^*) = s^*$) for the new root node was generated, utilizing previous methods to define polynomial for other new nodes in the tree, then, encrypt M^* as

$$CT^* = (\mathcal{T}^*, \tilde{C}^* = M_1^* e(g,g)^{\alpha s^*}, C^* = h^{s^*}, \forall\ y \in Y^* : C_y^* = g^{q_y{}^*(0)}, C_y'^* = H(att(y))^{q_y{}^*(0)})$$

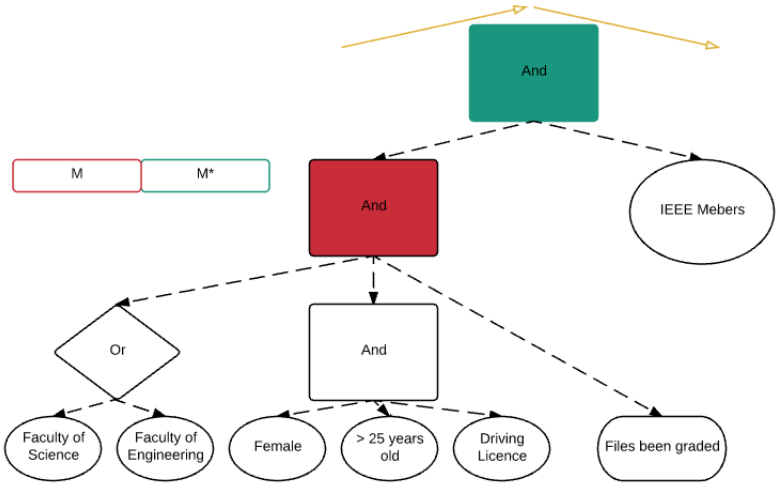

Figure 7.4: Generating new polynomials

The access structure could continue to be developed by repeating the algorithm Encrypt-Plus. Particularly, data producers can generate the new root when a threshold is "Or". In this way, when a user's attributes satisfy any/both sides of the subtree, the user satisfies the access structure.

KeyGen (MK, S). The key generation algorithm will take each set of attributes as input and generate keys based on the attributes set. It chooses a random $r \in Z_p$, and random $r_j \in Z_p$ for each attribute j $\in S_i$. Then it computes the key as below

$$SK_i = (\mathrm{D}= g^{(\alpha+r)/\beta}, \forall\ \mathrm{j} \in S_i\colon\ D_j = g^r \cdot H(j)^{r_j}, D_j' = g^{r_j})$$

Decrypt (CT, SK). Data consumers utilise their private keys to decrypt ciphertext. Ciphertext could be decrypted when the private keys satisfy the corresponding access structure.

To decrypt a piece of ciphertext M_k, this algorithm takes a ciphertext $CT_k = (\mathcal{T}, \tilde{C}_k, C_k, \forall\ y \in Y : C_{y_k}, C'_{y_k})$, a private key SK, which is associated with a set of attributes, and a node x from $\mathcal{T}_k$.

If the node x is a leaf node, we define i= att(x) and define as follows: if $i \in S$, then

DecryptNode(CT_k,SK,x)= $\frac{e(D_i,C_{kx})}{e(D'_i,C_{kx'})} = \frac{e(g_r \cdot H(i)^{r_i}, h^{q_x(0)})}{e(g^{r_i}, H(i)^{q_x(0)})} = e(g,g)^{rq_x(0)}$

If $i \notin S$, then we let DecryptNode(CT_k,SK,x)= $\perp$

We now consider the case when x is not a leaf node. The algorithm DecryptNode (CT_k, SK, x) proceeds as: for all nodes z that are children of x, it calls DecryptNode (CT_k, SK, z) and stores the output as F_z. Let S_x be an arbitrary $k_x - sized$ set of child nodes z such that $F_z \neq \perp$. If no such set exists then the node was not satisfied and the function returns $\perp$.

Otherwise, it computes F_x and returns the following result.

$F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)}$, $where^{i=index(z)}_{S'_x=\{index(z):z\in S_x\}}$
$=\prod_{z\in S_x} (e(g,g)^{r\cdot q_z(0)})^{\Delta_{i,S'_x}(0)}$
$=\prod_{z\in S_x} (e(g,g)^{r\cdot q_{parent(z)}(index(z))})^{\Delta_{i,S'_x}(0)}$ (by construction)
$=\prod_{z\in S_x} (e(g,g)^{r\cdot q_x(i)})^{\Delta_{i,S'_x}(0)}$
$=e(g,g)^{r\cdot q_x(0)}$ (using polynomial interpolation)

Now that we have completed the definition of the function DecryptNode, the algorithm starts by calling this function on the root node R of the tree T. If the tree is satisfied by S, we get A= DecryptNode (CT_k, SK, r)= $e(g,g)^{rq_R(0)} = e(g,g)^{rs_k}$. The algorithm now decrypts by calculating

$$\tilde{C}_k/(e(C_k, D)/A) = \tilde{C}_k(e(h^{s_k}, g^{(\alpha+r)/\beta})/e(g,g)^{rs_k}) = M_k$$

7.5 Security Proof

As PBE is extended from CP-ABE, we prove the security of PBE based on the security of CP-ABE. Namely, if any vulnerability can break our proposed scheme, it can break CP-ABE. Hence, PBE is expected to be secure under the random oracle model and the generic bilinear group model as CP-ABE.

7.5.1 Security Model for PBE

Next, we define a security model for our provenance-based encryption scheme. This model describing interactions between a challenger and an adversary, allows the adversary to query for any secret keys which are unable to decrypt the challenged ciphertext. In PBE, the ciphertext is encrypted by an access structure and privates keys are associated with a set of attributes. Hence, in this security model, the adversary selects an access structure for encryption to be challenged and queries for secrets keys do not satisfy this access structure.

Setup. The challenger generates public and private keys and sends public keys to the adversary.

Phase 1. The adversary queries master keys by sending sets of attributes $S_1, S_2, ...S_q$. The challenger runs algorithm *Key Generation* to generate the secret key SK_i based on the attribute set S_i and sends it back to the adversary.

Challenge. The adversary sends two equal length messages M_0, M_1 and a challenge access structure $\mathcal{T}$. None of the previously queried sets satisfy $\mathcal{T}$. Then the challenger selects $b \in \{0, 1\}$ randomly and encrypts M_b. Next, the ciphertext CT_b will be sent back to the challenger.

Phase 2. The adversary repeats phase 1 to send attribute sets S_{q+1}, $S_{q+2}, ...$ for a query, where these sets still do not satisfy the challenged access structure. The challenger calculates corresponding keys and sends them back to the adversary.

Guess. Finally, the adversary makes a guess and outputs b′. It wins the game if b′=b. The advantage to winning this game is defined as $Pr[b' = b] - 1/2$.

7.5.2 Security Proof

Definition 1: If all polynomial time adversaries have at most a negligible advantage in the security game, then the PBE scheme is secure.

Theorem 1: Assume there is no polytime adversary with a non-negligible advantage to break the security of CP-ABE, then there is no polytime adversary with a non-negligible advantage which can break the security of PBE.

Proof: We simulate PBE under the above security model. Assume there is an adversary $\mathcal{A}$ against our presented PBE scheme with a non-negligible advantage. We can construct an adversary $\mathcal{B}$ with nonnegligible, using $\mathcal{A}$, to break CP-ABE scheme. The security model of CP-ABE[21] consists of five algorithms which are Setup, Phase1, Challenge, Phase2, and Guess.

- Initialisation. The adversary $\mathcal{B}$ gets CP-ABE public key, PK = {$\mathbb{G}$, g, h=g^{β}, f=$g^{1/\beta}$, $e(g,g)^{\alpha}$}, but the adversary does not know the corresponding secret key SK = (β, g^{α}) for access structure $\mathbb{A}$.

- Setup. Adversary $\mathcal{B}$ develops access structure $\mathbb{A}$ to generate $\mathbb{A}^*$ and delivers the public key to adversary $\mathcal{A}$.

- Phase 1. In querying phase 1, adversary $\mathcal{A}$ queries a set of private keys S, where S can not satisfy $\mathbb{A}^*$. To answer these queries, the adversary sends them to CP-ABE challenger and receives the private keys:

$$SK_i = (\text{D}= g^{(\alpha+r)/\beta}, \forall\ \text{j} \in S_i\text{:}\ D_j = g^r \cdot H(j)^{r_j}, D_j\text{'}= g^{r_j})$$

- Challenge. When Phase 1 stops, adversary $\mathcal{A}$ generate two messages M_0, $M_1 \in \mathbb{G}$ and an access structure $\mathcal{T}$ to be challenged. $\mathcal{B}$ delivers M_0 and M_1 to CP-ABE challenger which sent the ciphertext back, $CT = (\tau, \tilde{C} = M_1 e(g,g)^{\alpha s}), C = h^s, \forall\ y \in Y : C_y = g^{q_y(0)}$, $C'_y = H(att(y))^{q_y(0)})$. Following, $\mathcal{B}$ calculate ciphertext for adversary $\mathcal{A}$ as $CT^* =$

$(\mathcal{T}^*, \tilde{C}^* = M_1^* e(g,g)^{\alpha s^*}, C^* = h^{s^*}, \forall\ y \in Y^* :\ C_y^* = g^{q_y{}^*(0)}, C_y'^* = H(att(y))^{q_y{}^*(0)})$. The ciphertext CT^* will be sent to $\mathcal{A}$.

- Phase2. Repeating algorithms in Phase1, $\mathcal{B}$ answers queries from adversary $\mathcal{A}$.
- Guess. In the end, adversary $\mathcal{A}$ make a guess $b' \in \{0, 1\}$, and adversary $\mathcal{B}$ answers its own game by giving b'. Based on the CP-ABE security model, the advantage of $\mathcal{B}$ against PBE is

$$Adv_{\mathcal{B}} = |Pr[b = b'] - 1/2| = Adv_{\mathcal{A}}$$

The above proof indicates $\mathcal{A}$ cannot make a correct guess as adversary $\mathcal{B}$ has a nonnegligible advantage against the CP-ABE scheme.

7.6 Discussion

Our scheme is extended from CP-ABE[21]. Hence we illustrate security features of PBE comparing CP-ABE.

Fine-grained Access Control. As we employ historical transactions as attributes, our system supports provenance-based access control policies. It supports more expressions of policy conditions compared to CP-ABE. For instance, in this system, whether owning a credential on the permission to “review graded files" could be set as an access restriction. In practice, comparing traditional attributes in CP-ABE, processes performed on files might be more significant to classify the files and further make access decisions. Introducing transactions in provenance as attributes supports more accurate and reasonable policies.

Flexibility. File co-authors could develop new access structures upon the original access structure based on their preferences. They could require data consumers to provide more attributes to access their generated data, in order to enhance the security preservation of specific content.

Efficiency. Considering the efficiency and computational cost, for both developing access structure by data producers and also computing the hidden elements from data consumers, the solution proposed in this chapter is efficient and has a low computational cost. Specifically, when data consumers calculate the original root of the original access tree, based on the

result, they continue a few steps to compute roots of new access trees. This approach is practical to implement in reality.

Expressiveness. As our scheme develops from CP-ABE rather than KP-ABE, the plaintext is encrypted by policies, and keys are associated with attributes. Therefore, this solution is more natural to implement access control policies.

7.7 Conclusion

In this chapter, we provide a scheme to implement fine-grained access control policies based on provenance, which is PBE scheme. It is an extension of Attribute-based Encryption schemes. PBE scheme encrypts ciphertext based on provenance-based access control policies, and it proposes an approach to fast generate encryption keys under related access tree structures. Furthermore, we prove the security of PBE under the random oracle model and the generic bilinear group model based on the security of CP-ABE. In the end, we discuss the properties of the PBE scheme by comparing it with CP-ABE.

To competently perform rectifying security service, two critical incident response elements are necessary: information and organisation.

Robert E. Davis

8

Conclusion

Even though provenance research has been explored in recent years, it still stays at its initial stage. The goal of policies proposed in this thesis is that utilizing provenance as conditions to determine whether data can be accessed or not. Two major aspects are proposed including three access control policy frameworks involving provenance and two cryptographic schemes to implement the policies.

Our proposed policies are based on OPM$^+$, which supports more fine-grained access control policies. At the first time, we renovated several existing policies models by introducing new created elements to support fine-grained access control policies involving provenance. These new techniques serve provenance related policies at a better performance, as provenance graphs usually contain a large number of records and attributes. Moreover, our policies languages are proofed by the experiments that they are able to define provenance partitions properly.

Following, we summarize and conclude our main contributions in the thesis one by one:

1. PACLP: A Partition-Based Access Control Policy Language for Provenance

We propose a fine-grained provenance access control language PACLP under the extended OPM (OPM$^+$). The OPM$^+$ containing attribute sets facilitates more fine-grained access control policies. Moreover, in this framework, various types of provenance partitions are defined. The partitions which are a collection of connected vertices can be expressed by XPath. Succinctly, our proposed policy language is able to define which provenance partitions can (or can not) be accessed and how to transform the partitions under specific conditions. Moreover, we also present algorithms to evaluate a request based on access control policies and transform graphs for returning permitted provenance subgraphs to the requestors.

We did experiments to implement PACLP, which shows our proposed language models can express more fine-grained policies. In the experiments, we measure the numbers of provenance partitions can be expressed by PACLP and other policy model [42]. The results indicate that PACLP could define more partitions comparing with the other model. The experiments to simulate policy generation and individual policy results merging are also implemented.

2. A fine-grained Policy Model for Provenance-based Access Control

We propose a fine-grained Provenance-based Access Control framework for defining atomic targets and policy algebras, which utilises provenance as conditions to determine the accessibility of data. Specifically, three types of atomic targets are provided. Path atomic target only employs attributes in provenance as conditions, which implies that if certain operations have been performed on the data, it can be accessed or not for certain requestors. Associated atomic target takes both attributes of provenance and requests as policy conditions. It denotes that if the requestor has performed certain operations on the targeted data, it can be accessed or not. Both atomic targets take a form of a string of vertices linked by Xpath, where each vertex is defined by a quaternion of attributes.

Moreover, the result of an atomic target of our provenance-based access control policy is one value in a four-valued decision set. The atomic targets are organised under a tree structure. New logic operators to merge the four-valued results are proposed accordingly.

We evaluate the policy generation and policy algebras under the proposed framework. Because of the transformation from atomic targets to atomic conditions, the access control

section usually takes longer time compared with the target section.

3. Purpose-based Privacy Policies on Purposes and the Policy Algebras

Further, purpose-based privacy policies on provenance are proposed, in order to establish a comprehensive scope of access control mechanisms on provenance. First, we define the policy model including syntax and semantics. This maps attributes in provenance to allowed and prohibited purposes sets. Particularly, the purposes with different sensitivities are classified as various levels.

We define internal and external policy algebras for the purpose-based policies. Differing from previous policy algebras to the best our knowledge, we are the first to define functions to merge two purpose sets with elements in different sensitive levels by various operators. Moreover, external policy algebras to merge policy results from multi-parties are also presented. Similar to the previous two chapters, we did experiments to evaluate the policy model.

4. Provenance-based Classification Policy Based on Encrypted Search for Cloud Data Storage

We propose a Provenance-based Classification Policy that is able to classify encrypted files based on provenance. To preserve the confidentiality of provenance, the provenance itself is in an encrypted form. Hence, we propose a scheme which enables the cloud server to check whether the provenance contains the specific keywords without decrypting it. Namely, the scheme allows searching encrypted provenance. Furthermore, the scheme is also able to check the identity of users who sent these files. We have described the scheme in detail and developed a provenance-based classification security game. In addition, we prove that the proposed scheme is semantically secure based on a hard problem.

5. Provenance-based Hierarchical Encryption for Fine-grained Access Control in Cloud Computing

In the end, we propose a scheme to implement fine-grained access control policies based on provenance, which is the PBE scheme. It is an extension of Attribute-based Encryption schemes. Sections of a piece of data might be generated by various data owners who attach provenance-based access control policies based on their own preferences. The PBE scheme

encrypts the sections based on each access control policy, and it proposes an approach to generate keys for encryption under related access tree structures. Furthermore, we prove the security of PBE under the random oracle model and the generic bilinear group model based on the security of CP-ABE. Moreover, we discuss the properties of PBE scheme by comparing it with CP-ABE.

8.1 Future Work

In relation to the access control on provenance, but we will endeavor to solve these problems with further work in the future.

1. Mechanisms to Update Access Control Policies on Provenance

As provenance data is dynamic and develops along with continuing operations performed on data, access control policies might be updated as required. Hence, mechanisms to delete or change provenance partitions defined in access control policies should be proposed.

2. System Architecture

Other than simple tailoring access control policies, there are more approaches to facilitate access control on provenance, including designing proper system architectures. The application to construct a personalised database by combining several useful databases will implement statement-level access control and version management.

3. Cryptographic Schemes

Cryptographic schemes are tools to implement access control policies. Therefore, more schemes are expected to be tailored for access control policies on provenance especially for key management and distribution, when multi-parities generate access control policies for a piece of data.

A

Appendix

A.1 Cryptography Tools

In this section, we introduce cryptographical tools relevant to this thesis, including group, filed, bilinear groups, hash function, random oracle model, public-key encryption, and digital signature.

A.1.1 Group

A group consists of a set of elements and an operation which is executed between any two elements in the set. The formal definition of a group is described as follows:

Definition 11. *(Group) A group* $(G, \otimes)$ *is a set G equipped with an operation* $\otimes$*, and satisfies the following properties:*

Closure. *For all* $g, h \in G$, $g \otimes h \in G$;

Associativity. *For all $g, h, \eta \in G$, $(g \otimes h) \otimes \eta = g \otimes (h \otimes \eta)$;*

Identity. *There exists $1_G \in G$ called the identity of $(G, \otimes)$, such that $1_G \otimes g = g \otimes 1_G = g$ for all $g \in G$;*

Inverse. *For all $g \in G$, there exists $g^{-1} \in G$ called the inverse of g such that $g \otimes g^{-1} = g^{-1} \otimes g = 1_G$.*

For simplicity, a group $(G, \otimes)$ is often denoted as G when the operation $\otimes$ is clear. The number of the elements in G is called the order of G and denoted as $|G|$. A group G is a finite group if $|G|$ is finite; otherwise, it is an infinite group. A group G is an Abelian group if for all $g, h \in G$, $g \otimes h = h \otimes g$.

Let $\mathcal{G}(1^\ell)$ be a group generator which takes as input 1^ℓ and outputs a group G with order p, namely $\mathcal{G}(1^\ell) \rightarrow (p, G)$.

Definition 12. *(Order of Group Element) Suppose that $g \in G$, the order of g in G is the least $i \in \mathbb{Z}^+$ such that $g^i = 1_G$. If for all $i \in \mathbb{Z}^+$, $g^i \neq 1_G$, the order of g is infinite. The order of g is denoted as $ord(g)$.*

Especially, if any element in a group G can be expressed by a specially element in G, G is called as a cyclic group. The formal definition of a cyclic group is as follows:

Definition 13. *(Cyclic Group.) A group G is a cyclic group if there exists $g \in G$, for all $h \in G$, there exists $i \in \mathbb{Z}$ such that $h = g^i$. The element g is called as a generator of the group G. G is said to be generated by g and denoted as $G = \langle g \rangle$.*

A.1.2 Field

A field consists of a set of elements and two operations defined between any two elements in the set. The formal definition of a field is described as follows.

Definition 14. *(Field) A field $(\mathbb{F}, \oplus, \otimes)$ consists of a set $\mathbb{F}$ and two operations: addition $\oplus$ and multiplication $\otimes$, and satisfies the following properties.*

Addition Group. *$(\mathbb{F}, \oplus)$ is an Abelian group. The identity of the group $(\mathbb{F}, \oplus)$ is denoted as $0_{\mathbb{F}}$ and called additive identity or zero-element;*

Multiplication Group. *Let* $\mathbb{F}^* = \mathbb{F} - \{0_{\mathbb{F}}\}$. $(\mathbb{F}^*, \otimes)$ *is an Abelian group. The identity of the group* $(\mathbb{F}^*, \otimes)$ *is denoted as* $1_{\mathbb{F}}$ *and called as multiplicative identity;*

Distributivity. *For all* $g, h, \eta \in \mathbb{F}$, $(g \oplus h) \otimes \eta = (g \otimes \eta) \oplus (h \otimes \eta)$.

A.1.3 Bilinear Maps

Let G_1 and G_2 be two groups of multiplicative cyclic groups of prime order p, and g be a generator of G, A bilinear map is a map e: G*G → G with the following properties:

1. Bilinearity: for all u, v ∈ G and a, b ∈ Z_p*, $e(u_a, v_b) = e(u, v)^{ab}$.
2. Non-degeneracy: e(g,g)≠ 1.
3. Computable: for any u,v ∈ G, e(u,v) can be computed.

A.1.4 Bilinear Groups

In this section, we review the knowledge related to bilinear group.

Definition 15. *(Bilinear Map [78]) Suppose that* G_1*,* G_2 *and* G_T *are three cyclic groups with the same order p. Let g and h be the generators of* G_1 *and* G_2*, respectively. A bilinear map (pairing) is a map* $e : G_1 \times G_2 \rightarrow G_T$ *satisfying the following properties :*

Bilinearity. *For all* $x \in G_1, y \in G_2$ *and* $a, b \in \mathbb{Z}_p$, $e(x^a, y^b) = e(x, y)^{ab}$.

Non-degeneracy. $e(g, h) \neq 1_{G_T}$ *where* 1_{G_T} *is the identity of the group* G_T.

Computability. *For all* $x \in G_1$ *and* $y \in G_2$*, there exists an efficient algorithm to compute* $e(x, y)$.

Definition 16. *(Bilinear Groups [130])* G_1, G_2*, and* G_T *constitute a bilinear group if there exists a bilinear map* $e : G_1 \times G_2 \rightarrow G_T$*, where* $|G_1| = |G_2| = |G_T| = p$.

Galbraith, Paterson and Smart [130] divided pairing operations used in cryptography into three types:

- $G_1 = G_2$;

- $G_1 \neq G_2$, there exists an efficiently computable homomorphism map $\psi : G_1 \rightarrow G_2$;
- $G_1 \neq G_2$, there are no efficiently computable homomorphism maps between groups G_1 and G_2.

We say that a pairing is symmetric if $G_1 = G_2$ and denote the symmetric bilinear group as (e, p, G_1, G_T). Pairing is often constructed on suitable elliptic curves, so its efficiency is determined by the selected elliptic curves. When selecting elliptic curves for a pairing, two factors must be considered: the group size l of the elliptic curves and the embedding degree d. Generally, to achieve the security of 1,024-bit RSA, the two parameters l and d should satisfy $l \times d \geq 1,024$ [131, 132].

In the rest of this thesis, we denote $\mathcal{GG}(1^\ell) \rightarrow (e, p, G_1, G_2, G_T)$ as a bilinear group generator which takes as input 1^ℓ and outputs bilinear groups (e, p, G_1, G_2, G_T) with order p and a bilinear map $e : G_1 \times G_2 \rightarrow G_T$.

A.1.5 Hash Function

Carter and Wegman [133] introduced the universal classes of hash functions and divided them into tree types. Roughly speaking, a hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{0,1\}^\lambda$ is a deterministic function which maps a bit string with any length to a bit string with fixed length λ. A hash function should meet the following properties [134]:

- Mixing Transformation. The output of $\mathcal{H}$ should be computationally indistinguishable from a uniform binary string in $[0, 2^\lambda]$;
- Pre-image Resistance. Given a value y, it is computationally infeasible to find a value x such that $y = \mathcal{H}(x)$;
- Collusion Resistance. It is computationally infeasible to find $x \neq y$ such that $\mathcal{H}(x) = \mathcal{H}(y)$.

Hash function is an important cryptographical primitive and has been used as a building block to design encryption schemes [117], digital signature schemes[135], message authentication code (MAC) scheme [136], *etc.*

A.1.6 Random Oracle Model

A hash function should satisfy the mixing transformation property, namely, the output of a hash function is computationally indistinguishable from the uniform distribution over its output's space. If the output of a hash function is uniform distribution over its output's space, it is a very powerful and ideal hash function called *random oracle* [134]. A random oracle is a powerful hash function as it combines the properties: deterministic, efficient and uniform output. Furthermore, a random oracle is an ideal hash function as there is nothing so powerful in computing mechanism or machinery in current computing models.

Bellare and Rogaway [135] introduced the notion of *random oracle model*. In this model, a special entity called *Simulator* can simulate every party's behavior. So, whenever a party wants to obtain the output of a random oracle $\mathcal{H}$ on a value x, he must make a *random oracle query* on the value x to the *Simulator*. The *Simulator* maintains a $\mathcal{H}$-table consisting of pairs $(z, \mathcal{H}(z))$. For a query on the value x, the *Simulator* checks whether x is listed in the table. If it is in the table, the *Simulator* responds with the value $\mathcal{H}(x)$ (*deterministic*); otherwise, the *Simulator* creates a new value $\mathcal{H}(x)$ uniformly at random from the output's space of $\mathcal{H}$, adds the pair $(x, \mathcal{H}(x))$ to the table and responds with $\mathcal{H}(x)$ (*uniform*).

Random oracle model is a very effective tool to prove the security of cryptographic protocols. Generally, protocols designed in this model are more efficient than those designed in a standard model, whereas, a scheme even though it is proven to be secure in the random oracle model does not necessarily guarantee that it is secure in the standard model [137].

Unless otherwise specified, by saying a scheme is secure, we mean that it is secure in the standard model in this thesis.

A.2 Access Tree

Definition 1 (Access Structure[138]). We assume $\{P_1, P_2, ...P_n\}$ to be a set of parties. A collection A $\in 2^{\{P_1, P_2, ...P_n\}}$ is monotone if $\forall$ B,C: if B $\in$ A and B $\subseteq$ C then C $\in$ A. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) A of non-empty subsets of $\{P_1, P_2, ...P_n\}$, i.e., A$\in 2^{\{P_1, P_2, ...P_n\}} \setminus \{\emptyset\}$. The sets in A are called the authorised sets, and the sets not in A are called the unauthorised sets. In our policies, the parties are taken as attributes. Then, the access structure A contains a set of

authorised attributes.

Access Tree $\mathcal{T}$ [21]. Let $\mathcal{T}$ be a tree with root r. The leaf nodes are set as attributes, and each non-leaf node represents a threshold gate which is described by a threshold value and its children. If num_x is the number of children of a node x and k_x is its threshold value, then $0 < k_x < num_x$. When $k_x = 1$, the threshold gate is an *OR* gate; when $k_x = num_x$, it is an *AND* gate; when $1 < k_x < num_x$, it satisfies at least any k_x attributes.

Satisfying an Access Tree[21]. Let $\mathcal{T}$ be an access tree with root r. Assume $T_x(r)$ is a subtree of $\mathcal{T}$ with a root node x. For every subtree, if x is a non-leaf node, it returns 1 if and only if at least k_x of its children nodes returns 1; if x is a leaf node, it returns 1 if the attribute satisfies x or belongs to a subclass of x.

Attributes of the Access Tree. In our proposed system, the attributes could be credentials on user's roles, features, processes executed on data, dependencies paths from provenance *etc.*

References

[1] L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. T. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. G. Stephan, and J. V. den Bussche. *The open provenance model core specification (v1.1).* pp. 743–756 (2011).

[2] Q. Ni, S. Xu, E. Bertino, R. S. Sandhu, and W. Han. *An access control language for a general provenance model*. In *Secure Data Management, 6th VLDB Workshop, SDM 2009, Lyon, France, August 28, 2009. Proceedings*, pp. 68–88 (2009). URL http://dx.doi.org/10.1007/978-3-642-04219-5_5.

[3] L. Sun, J. Park, D. Nguyen, and R. S. Sandhu. *A provenance-aware access control framework with typed provenance*. IEEE Trans. Dependable Sec. Comput. **13**(4), 411 (2016). URL http://dx.doi.org/10.1109/TDSC.2015.2410793.

[4] A. M. Bates, B. Mood, M. Valafar, and K. R. B. Butler. *Towards secure provenance-based access control in cloud environments*. In *Third ACM Conference on Data and Application Security and Privacy, CODASPY'13, San Antonio, TX, USA, February 18-20, 2013*, pp. 277–284 (2013). URL http://doi.acm.org/10.1145/2435349.2435389.

[5] L. Fang, W. Susilo, C. Ge, and J. Wang. *Public key encryption with keyword search secure against keyword guessing attacks without random oracle*. Inf. Sci. **238**, 221 (2013). URL http://dx.doi.org/10.1016/j.ins.2013.03.008.

[6] W. Yau, S. Heng, and B. Goi. *Off-line keyword guessing attacks on recent public key encryption with keyword search schemes*. In *Autonomic and Trusted Computing, 5th International Conference, ATC 2008, Oslo, Norway, June 23-25, 2008, Proceedings*, pp. 100–105 (2008). URL http://dx.doi.org/10.1007/978-3-540-69295-9_10.

[7] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. *Public key encryption with keyword search*. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pp. 506–522 (2004). URL http://dx.doi.org/10.1007/978-3-540-24676-3_30.

[8] D. J. Park, K. Kim, and P. J. Lee. *Public key encryption with conjunctive field keyword search*. In *Information Security Applications, 5th International Workshop, WISA 2004, Jeju Island, Korea, August 23-25, 2004, Revised Selected Papers*, pp. 73–86 (2004). URL http://dx.doi.org/10.1007/978-3-540-31815-6_7.

[9] J. Baek, R. Safavi-Naini, and W. Susilo. *Public key encryption with keyword search revisited*. In *Computational Science and Its Applications - ICCSA 2008, International Conference, Perugia, Italy, June 30 - July 3, 2008, Proceedings, Part I*, pp. 1249–1259 (2008). URL http://dx.doi.org/10.1007/978-3-540-69839-5_96.

[10] J. Baek, R. Safavi-Naini, and W. Susilo. *On the integration of public key data encryption and public key encryption with keyword search*. In *Information Security, 9th International Conference, ISC 2006, Samos Island, Greece, August 30 - September 2, 2006, Proceedings*, pp. 217–232 (2006). URL http://dx.doi.org/10.1007/11836810_16.

[11] A. Cuzzocrea. *Big data provenance: State-of-the-art analysis and emerging research challenges*. In *Proceedings of the Workshops of the EDBT/ICDT 2016 Joint Conference, EDBT/ICDT Workshops 2016, Bordeaux, France, March 15, 2016.* (2016). URL http://ceur-ws.org/Vol-1558/paper37.pdf.

[12] R. Sandhu and P.Samarati. *Access control: Principles and practice*. IEEE Communications Magazine(Sept.) pp. 40–48 (1994).

[13] L. Kerr and J. Alves-Foss. *Combining mandatory and attribute-based access control*. In *49th Hawaii International Conference on System Sciences, HICSS 2016, Koloa, HI, USA, January 5-8, 2016*, pp. 2616–2623 (2016). URL http://dx.doi.org/10.1109/HICSS.2016.328.

[14] D. Nguyen, J. Park, and R. S. Sandhu. *A provenance-based access control model for dynamic separation of duties*. In *Eleventh Annual International Conference on Privacy, Security and Trust, PST 2013, 10-12 July, 2013, Tarragona, Catalonia, Spain, July 10-12, 2013*, pp. 247–256 (2013). URL http://dx.doi.org/10.1109/PST.2013.6596060.

[15] O. Benjelloun, A. D. Sarma, A. Y. Halevy, M. Theobald, and J. Widom. *Databases with uncertainty and lineage*. VLDB J. **17**(2), 243 (2008). URL http://dx.doi.org/10.1007/s00778-007-0080-z.

[16] P. Buneman, A. Chapman, and J. Cheney. *Provenance management in curated databases*. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, pp. 539–550 (2006). URL http://doi.acm.org/10.1145/1142473.1142534.

[17] M. K. Anand, S. Bowers, T. M. McPhillips, and B. Ludäscher. *Efficient provenance storage over nested data collections*. In *EDBT 2009, 12th International Conference on Extending Database Technology, Saint Petersburg, Russia, March 24-26, 2009, Proceedings*, pp. 958–969 (2009). URL http://doi.acm.org/10.1145/1516360.1516470.

[18] T. Heinis and G. Alonso. *Efficient lineage tracking for scientific workflows*. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, Vancouver, BC, Canada, June 10-12, 2008*, pp. 1007–1018 (2008). URL http://doi.acm.org/10.1145/1376616.1376716.

[19] J. Crampton and C. Morisset. *Ptacl: A language for attribute-based access control in open systems*. In *Principles of Security and Trust - First International Conference, POST 2012, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2012, Tallinn, Estonia, March 24 - April 1, 2012, Proceedings*, pp. 390–409 (2012). URL http://dx.doi.org/10.1007/978-3-642-28641-4_21.

[20] D. Boneh and M. K. Franklin. *Identity-based encryption from the weil pairing*. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, pp. 213–229 (2001). URL http://dx.doi.org/10.1007/3-540-44647-8_13.

[21] J. Bethencourt, A. Sahai, and B. Waters. *Ciphertext-policy attribute-based encryption*. In *2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA*, pp. 321–334 (2007). URL http://dx.doi.org/10.1109/SP.2007.11.

[22] S. B. Davidson and S. Roy. *Provenance: Privacy and security* .

[23] B. Lee, A. Awad, and M. Awad. *Towards secure provenance in the cloud: A survey*. In *8th IEEE/ACM International Conference on Utility and Cloud Computing, UCC 2015, Limassol, Cyprus, December 7-10, 2015*, pp. 577–582 (2015). URL http://doi.ieeecomputersociety.org/10.1109/UCC.2015.102.

[24] Y. S. Tan, R. K. L. Ko, and G. Holmes. *Security and data accountability in distributed systems: A provenance survey*. In *10th IEEE International Conference on High Performance Computing and Communications & 2013 IEEE International Conference on Embedded and Ubiquitous Computing, HPCC/EUC 2013, Zhangjiajie, China, November 13-15, 2013*, pp. 1571–1578 (2013). URL https://doi.org/10.1109/HPCC.and.EUC.2013.221.

[25] Y. Simmhan, B. Plale, and D. Gannon. *A survey of data provenance in e-science*. SIGMOD Record **34**(3), 31 (2005). URL http://doi.acm.org/10.1145/1084805.1084812.

[26] N. N. Vijayakumar and B. Plale. *Towards low overhead provenance tracking in near real-time stream filtering*. In *Provenance and Annotation of Data, International Provenance and Annotation Workshop, IPAW 2006, Chicago, IL, USA, May 3-5, 2006, Revised Selected Papers*, pp. 46–54 (2006). URL https://doi.org/10.1007/11890850_6.

[27] A. Lesas, O. Boucelma, and J. Lacroix. *PBAC4M: provenance-based access control for mobile*. In *Proceedings of the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, Melbourne, Australia, November 7-10, 2017.*, pp. 529–530 (2017). URL http://doi.acm.org/10.1145/3144457.3144513.

[28] T. Ma, H. Wang, J. Cao, J. Yong, and Y. Zhao. *Access control management with provenance in healthcare environments*. In *20th IEEE International Conference on Computer Supported Cooperative Work in Design, CSCWD 2016, Nanchang, China, May 4-6, 2016*, pp. 545–550 (2016). URL https://doi.org/10.1109/CSCWD.2016.7566048.

[29] *The U.S. Public Law*. 107-204 (The Public Company Accounting Reform and Investor Protection Act, 2002).

[30] L. Moreau and I. T. Foster, eds. *Provenance and Annotation of Data, International Provenance and Annotation Workshop, IPAW 2006, Chicago, IL, USA, May 3-5, 2006,*

Revised Selected Papers, vol. 4145 of *Lecture Notes in Computer Science* (Springer, 2006).

[31] J. Freire, D. Koop, and L. Moreau, eds. *Provenance and Annotation of Data and Processes, Second International Provenance and Annotation Workshop, IPAW 2008, Salt Lake City, UT, USA, June 17-18, 2008. Revised Selected Papers*, vol. 5272 of *Lecture Notes in Computer Science* (Springer, 2008).

[32] D. L. McGuinness, J. Michaelis, and L. Moreau, eds. *Provenance and Annotation of Data and Processes - Third International Provenance and Annotation Workshop, IPAW 2010, Troy, NY, USA, June 15-16, 2010. Revised Selected Papers*, vol. 6378 of *Lecture Notes in Computer Science* (Springer, 2010). URL http://dx.doi.org/10.1007/978-3-642-17819-1.

[33] P. T. Groth and J. Frew, eds. *Provenance and Annotation of Data and Processes - 4th International Provenance and Annotation Workshop, IPAW 2012, Santa Barbara, CA, USA, June 19-21, 2012, Revised Selected Papers*, vol. 7525 of *Lecture Notes in Computer Science* (Springer, 2012). URL http://dx.doi.org/10.1007/978-3-642-34222-6.

[34] B. Ludäscher and B. Plale, eds. *Provenance and Annotation of Data and Processes - 5th International Provenance and Annotation Workshop, IPAW 2014, Cologne, Germany, June 9-13, 2014. Revised Selected Papers*, vol. 8628 of *Lecture Notes in Computer Science* (Springer, 2015). URL http://dx.doi.org/10.1007/978-3-319-16462-5.

[35] E. R. Lindgreen and I. S. Herschberg. *On the validity of the bell-la padula model*. Computers & Security **13**(4), 317 (1994). URL http://dx.doi.org/10.1016/0167-4048(94)90023-X.

[36] R. S. Sandhu. *On five definitions of data integrity*. In *Database Security, VII: Status and Prospects, Proceedings of the IFIP WG11.3 Working Conference on Database Security, Lake Guntersville, Alabama, USA, 12-15 September, 1993*, pp. 257–267 (1993).

[37] R. Zhou, C. Xu, W. Li, and J. Zhao. *An id-based hierarchical access control scheme with constant size public parameter*. I. J. Network Security **18**(5), 960 (2016). URL http://ijns.femto.com.tw/contents/ijns-v18-n5/ijns-2016-v18-n5-p960-968.pdf.

[38] P. D. McDaniel. *Data provenance and security*. IEEE Security & Privacy **9**(2), 83 (2011). URL https://doi.org/10.1109/MSP.2011.27.

[39] F. A. Bhuyan, S. Lu, R. G. Reynolds, I. Ahmed, and J. Zhang. *Quality analysis for scientific workflow provenance access control policies*. In *2018 IEEE International Conference on Services Computing, SCC 2018, San Francisco, CA, USA, July 2-7, 2018*, pp. 261–264 (2018). URL https://doi.org/10.1109/SCC.2018.00044.

[40] L. González-Manzano, M. Slaymaker, J. M. de Fuentes, and D. Vayenas. *Soneucon$_{abc}$pro: An access control model for social networks with translucent user provenance*. In *Security and Privacy in Communication Networks - SecureComm 2017 International Workshops, ATCS and SePrIoT, Niagara Falls, ON, Canada, October 22-25, 2017, Proceedings*, pp. 234–252 (2017). URL https://doi.org/10.1007/978-3-319-78816-6_17.

[41] U. Braun, A. Shinnar, and M. I. Seltzer. *Securing provenance*. In *3rd USENIX Workshop on Hot Topics in Security, HotSec'08, San Jose, CA, USA, July 29, 2008, Proceedings* (2008). URL http://www.usenix.org/events/hotsec08/tech/full_papers/braun/braun.pdf.

[42] T. Cadenhead, V. Khadilkar, M. Kantarcioglu, and B. M. Thuraisingham. *A language for provenance access control*. In *First ACM Conference on Data and Application Security and Privacy, CODASPY 2011, San Antonio, TX, USA, February 21-23, 2011, Proceedings*, pp. 133–144 (2011). URL http://doi.acm.org/10.1145/1943513.1943532.

[43] L. Moreau. *The foundations for provenance on the web*. Foundations and Trends in Web Science **2**(2-3), 99 (2010). URL http://dx.doi.org/10.1561/1800000010.

[44] R. Dánger, V. Curcin, P. Missier, and J. Bryans. *Access control and view generation for provenance graphs*. Future Generation Comp. Syst. **49**, 8 (2015). URL http://dx.doi.org/10.1016/j.future.2015.01.014.

[45] L. Chen, P. Edwards, J. D. Nelson, and T. J. Norman. *An access control model for protecting provenance graphs*. In *13th Annual Conference on Privacy, Security and Trust, PST 2015, Izmir, Turkey, July 21-23, 2015*, pp. 125–132 (2015). URL http://dx.doi.org/10.1109/PST.2015.7232963.

[46] K. Kuwabara and S. Yasunaga. *Use of metadata for access control and version management in RDF database*. In *Knowledge-Based and Intelligent Information and Engineering Systems - 15th International Conference, KES 2011, Kaiserslautern,*

Germany, September 12-14, 2011, Proceedings, Part I, pp. 326–336 (2011). URL http://dx.doi.org/10.1007/978-3-642-23851-2_34.

[47] E. Bertino, A. A. Jabal, S. B. Calo, C. Makaya, M. Touma, D. C. Verma, and C. Williams. *Provenance-based analytics services for access control policies*. In *2017 IEEE World Congress on Services, SERVICES 2017, Honolulu, HI, USA, June 25-30, 2017*, pp. 94–101 (2017). URL https://doi.org/10.1109/SERVICES.2017.24.

[48] F. Capobianco, C. Skalka, and T. Jaeger. *ACCESSPROV: tracking the provenance of access control decisions*. In *9th USENIX Workshop on the Theory and Practice of Provenance, TaPP 2017, Seattle, WA, USA, June 23, 2017*. (2017). URL https://www.usenix.org/conference/tapp17/workshop-program/presentation/capobianco.

[49] J. Park, D. Nguyen, and R. S. Sandhu. *A provenance-based access control model*. In *Tenth Annual International Conference on Privacy, Security and Trust, PST 2012, Paris, France, July 16-18, 2012*, pp. 137–144 (2012). URL http://dx.doi.org/10.1109/PST.2012.6297930.

[50] J. Lacroix and O. Boucelma. *Provenance-based access control in the cloud*. In *2013 IEEE International Conference on Services Computing, Santa Clara, CA, USA, June 28 - July 3, 2013*, pp. 755–756 (2013). URL http://dx.doi.org/10.1109/SCC.2013.51.

[51] D. Nguyen, J. Park, and R. S. Sandhu. *Adopting provenance-based access control in openstack cloud iaas*. In *Network and System Security - 8th International Conference, NSS 2014, Xi'an, China, October 15-17, 2014, Proceedings*, pp. 15–27 (2014). URL http://dx.doi.org/10.1007/978-3-319-11698-3_2.

[52] M. Decat, B. Lagaisse, and W. Joosen. *Scalable and secure concurrent evaluation of history-based access control policies*. In *Proceedings of the 31st Annual Computer Security Applications Conference, Los Angeles, CA, USA, December 7-11, 2015*, pp. 281–290 (2015). URL http://doi.acm.org/10.1145/2818000.2818008.

[53] F. Yan and P. W. L. Fong. *Efficient IRM enforcement of history-based access control policies*. In *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12, 2009*, pp. 35–46 (2009). URL http://doi.acm.org/10.1145/1533057.1533066.

[54] P. A. Bonatti, S. D. C. di Vimercati, and P. Samarati. *An algebra for composing access control policies*. ACM Trans. Inf. Syst. Secur. **5**(1), 1 (2002). URL http://doi.acm.org/10.1145/504909.504910.

[55] D. Wijesekera and S. Jajodia. *A propositional policy algebra for access control*. ACM Trans. Inf. Syst. Secur. **6**(2), 286 (2003). URL http://doi.acm.org/10.1145/762476.762481.

[56] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo. *An algebra for fine-grained integration of XACML policies*. In *SACMAT 2009, 14th ACM Symposium on Access Control Models and Technologies, Stresa, Italy, June 3-5, 2009, Proceedings*, pp. 63–72 (2009). URL http://doi.acm.org/10.1145/1542207.1542218.

[57] Q. Ni, E. Bertino, and J. Lobo. *D-algebra for composing access control policy decisions*. In *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12, 2009*, pp. 298–309 (2009). URL http://doi.acm.org/10.1145/1533057.1533097.

[58] C. Hanson, T. Berners-Lee, L. Kagal, G. J. Sussman, and D. J. Weitzner. *Data-purpose algebra: Modeling data usage policies*. In *8th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY 2007), 13-15 June 2007, Bologna, Italy*, pp. 173–177 (2007). URL http://doi.ieeecomputersociety.org/10.1109/POLICY.2007.14.

[59] E. Bertino, G. Ghinita, M. Kantarcioglu, D. Nguyen, J. Park, R. S. Sandhu, S. Sultana, B. M. Thuraisingham, and S. Xu. *A roadmap for privacy-enhanced secure data provenance*. J. Intell. Inf. Syst. **43**(3), 481 (2014). URL http://dx.doi.org/10.1007/s10844-014-0322-7.

[60] S. R. Hussain, C. Wang, S. Sultana, and E. Bertino. *Secure data provenance compression using arithmetic coding in wireless sensor networks*. In *IEEE 33rd International Performance Computing and Communications Conference, IPCCC 2014, Austin, TX, USA, December 5-7, 2014*, pp. 1–10 (2014). URL http://dx.doi.org/10.1109/PCCC.2014.7017068.

[61] J. Li, X. Chen, Q. Huang, and D. S. Wong. *Digital provenance: Enabling secure data forensics in cloud computing*. Future Generation Comp. Syst. **37**, 259 (2014). URL http://dx.doi.org/10.1016/j.future.2013.10.006.

[62] S. S. M. Chow, C. Chu, X. Huang, J. Zhou, and R. H. Deng. *Dynamic secure cloud storage with provenance*. In *Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday*, pp. 442–464 (2012). URL http://dx.doi.org/10.1007/978-3-642-28368-0_28.

[63] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano. *Public key encryption with keyword search*. IACR Cryptology ePrint Archive **2003**, 195 (2003). URL http://eprint.iacr.org/2003/195.

[64] D. Boneh and B. Waters. *Conjunctive, subset, and range queries on encrypted data*. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pp. 535–554 (2007). URL http://dx.doi.org/10.1007/978-3-540-70936-7_29.

[65] B. R. Waters, D. Balfanz, G. Durfee, and D. K. Smetters. *Building an encrypted and searchable audit log*. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2004, San Diego, California, USA* (2004). URL http://www.isoc.org/isoc/conferences/ndss/04/proceedings/Papers/Waters.pdf.

[66] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. *Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions*. In *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, pp. 205–222 (2005). URL http://dx.doi.org/10.1007/11535218_13.

[67] D. Khader. *Public key encryption with keyword search based on k-resilient IBE*. In *Computational Science and Its Applications - ICCSA 2007, International Conference, Kuala Lumpur, Malaysia, August 26-29, 2007. Proceedings. Part III*, pp. 1086–1095 (2007). URL http://dx.doi.org/10.1007/978-3-540-74484-9_95.

[68] G. D. Crescenzo and V. Saraswat. *Public key encryption with searchable keywords based on jacobi symbols*. In *Progress in Cryptology - INDOCRYPT 2007, 8th International Conference on Cryptology in India, Chennai, India, December 9-13, 2007, Proceedings*, pp. 282–296 (2007). URL http://dx.doi.org/10.1007/978-3-540-77026-8_21.

[69] C. Cocks. *An identity based encryption scheme based on quadratic residues*. In *Cryptography and Coding, 8th IMA International Conference, Cirencester, UK, December 17-19, 2001, Proceedings*, pp. 360–363 (2001). URL http://dx.doi.org/10.1007/3-540-45325-3_32.

[70] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. *Improved searchable public key encryption with designated tester*. In *Proceedings of the 2009 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2009, Sydney, Australia, March 10-12, 2009*, pp. 376–379 (2009). URL http://doi.acm.org/10.1145/1533057.1533108.

[71] K. Emura, A. Miyaji, M. S. Rahman, and K. Omote. *Generic constructions of secure-channel free searchable encryption with adaptive security*. Security and Communication Networks **8**(8), 1547 (2015). URL http://dx.doi.org/10.1002/sec.1103.

[72] J. W. Byun, H. S. Rhee, H. Park, and D. H. Lee. *Off-line keyword guessing attacks on recent keyword search schemes over encrypted data*. In *Secure Data Management, Third VLDB Workshop, SDM 2006, Seoul, Korea, September 10-11, 2006, Proceedings*, pp. 75–83 (2006). URL http://dx.doi.org/10.1007/11844662_6.

[73] H. S. Rhee, W. Susilo, and H. Kim. *Secure searchable public key encryption scheme against keyword guessing attacks*. IEICE Electronic Express **6**(5), 237 (2009). URL http://dx.doi.org/10.1587/elex.6.237.

[74] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee. *Trapdoor security in a searchable public-key encryption scheme with a designated tester*. Journal of Systems and Software **83**(5), 763 (2010). URL http://dx.doi.org/10.1016/j.jss.2009.11.726.

[75] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee. *Constructing PEKS schemes secure against keyword guessing attacks is possible?* Computer Communications **32**(2), 394 (2009). URL http://dx.doi.org/10.1016/j.comcom.2008.11.018.

[76] A. Sahai and B. Waters. *Fuzzy identity-based encryption*. In R. Cramer, ed., *Proceedings: Advances in Cryptology - EUROCRYPT 2005*, vol. 3494 of *Lecture Notes in Computer Science*, pp. 457–473 (Springer, Aarhus, Denmark, 2005).

[77] A. Shamir. *Identity-based cryptosystems and signature scheme*. In G. R. Blakley and D. Chaum, eds., *Proceedings: Advances in Cryptology - CRYPTO 1984*, vol. 196 of

Lecture Notes in Computer Science, pp. 47–53 (Springer, Santa Barbara, California, USA, 1984).

[78] D. Boneh and M. Franklin. *Identity-based encryption from the weil pairing*. In J. Kilian, ed., *Proceedings: Advances in Cryptology - CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 213–229 (Springer, Santa Barbara, California, USA, 2001).

[79] J. Bethencourt, A. Sahai, and B. Waters. *Ciphertext-policy attribute-based encryption*. In *Proceedings: IEEE Symposium on Security and Privacy - S & P 2007*, pp. 321–334 (Oakland, California, USA, 2007).

[80] L. Cheung and C. Newport. *Provably secure ciphertext policy abe*. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, eds., *Proceedings: ACM Conference on Computer and Communications Security-CCS 2007*, pp. 456–465 (ACM, Alexandria, Virginia, USA, 2007).

[81] J. Herranz, F. Laguillaumie, and C. Ráfols. *Constant size ciphertexts in threshold attribute-based encryption*. In P. Q. Nguyen and D. Pointcheval, eds., *Proceedings: Public Key Cryptography-PKC 2010*, Lecture Notes in Computer Science, pp. 19–34 (Springer, Paris, France, 2010).

[82] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. *Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption*. In H. Gilbert, ed., *Proceedings: Advances in Cryptology - EUROCRYPT 2010*, vol. 6110 of *Lecture Notes in Computer Science*, pp. 62–91 (Springer, Riviera, French, 2010).

[83] B. Waters. *Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization*. In D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, eds., *Proceedings: Public Key Cryptography - PKC 2011*, vol. 6571 of *Lecture Notes in Computer Science*, pp. 53–70 (Springer, aormina, Italy, 2011).

[84] V. Goyal, O. Pandey, A. Sahai, and B. Waters. *Attribute-based encryption for fine-grained access control of encrypted data*. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, eds., *Proceedings: 13th ACM Conference on Computer and Communications Security - CCS 2006*, pp. 89–98 (ACM, Alexandria, VA, USA, 2006).

[85] R. Ostrovsky, A. Sahai, and B. Waters. *Attribute-based encryption with non-monotonic access structures*. In P. Ning, S. D. C. di Vimercati, and P. F. Syverson, eds., *Proceedings: ACM Conference on Computer and Communications Security - CCS 2007*, pp. 159–203 (Alexandria, Virginia, USA, 2007).

[86] M. Chase. *Multi-authority attribute based encryption*. In S. P. Vadhan, ed., *Proceedings: Theory of Cryptography Conference-TCC'07*, vol. 4392 of *Lecture Notes in Computer Science*, pp. 515–534 (Springer, Amsterdam, The Netherlands, 2007).

[87] M. Chase and S. S. Chow. *Improving privacy and security in multi-authority attribute-based encryption*. In E. Al-Shaer, S. Jha, and A. D. Keromytis, eds., *Proceedings: ACM Conference on Computer and Communications Security-CCS'09*, pp. 121–130 (ACM, Chicago, Illinois, USA, 2009).

[88] A. Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. Phd thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996).

[89] N. Attrapadung and H. Imai. *Dual-policy attribute based encryption*. In M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, eds., *Proceedings: Applied Cryptography and Network Security-ACNS 2009*, vol. 5536 of *Lecture Notes in Computer Science*, pp. 168–185 (Springer, Paris-Rocquencourt, France, 2009).

[90] A. Rial and B. Preneel. *Blind attribute-based encryption and oblivious transfer with fine-grained access control*. In *Benelux Workshop on Information and System Security - WISSec'10*, pp. 1–20 (2010).

[91] J. Hur and D. K. Noh. *Attribute-based access control with efficient revocation in data outsourcing systems*. IEEE Transactions on Parallel and Distributed Systems **22**(7), 1214 (2011).

[92] S. Yu, C. Wang, K. Ren, and W. Lou. *Achieving secure, scalable, and fine-grained data access control in cloud computing*. In *Proceedings: IEEE INFOCOM 2010*, pp. 534–542 (IEEE, San Diego, CA, USA, 2010).

[93] S. Yu, K. Ren, and W. Lou. *FDAC: Toward fine-grained data access control in wireless sensor networks*. IEEE Transactions on Parallel and Distributed Systems **22**(4), 673 (2011).

[94] S. M*ü*ller, S. Katzenbeisser, and C. Eckert. *Distributed attribute-based encryption*. In P. J. Lee and J. H. Cheon, eds., *Proceedings: Information Security and Cryptology-ICISC'08*, vol. 5461 of *Lecture Notes in Computer Science*, pp. 20–36 (Springer, Seoul, Korea, 2008).

[95] R. Gennaro, S. law Jarecki, H. Krawczyk, , and T. Rabin. *Secure distributed key generation for discrete-log based cryptosystems*. In J. Stern, ed., *Proceedings: Advances in Cryptology-EUROCRYPT'99*, vol. 1592 of *Lecture Notes in Computer Science*, pp. 295–310 (Springer, Prague, Czech Republic, 1999).

[96] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. *Robust threshold DSS signatures*. Information and Computation **164**(1), 54 (2001).

[97] H. Lin, Z. Cao, X. Liang, and J. Shao. *Secure threshold multi-authority attribute based encryption without a central authority*. In D. R. Chowdhury, V. Rijmen, and A. Das, eds., *Proceedings: International Conference on Cryptology in India-INDOCRYPT'08*, vol. 5365 of *Lecture Notes in Computer Science*, pp. 426–436 (Springer, Kharagpur, India, 2008).

[98] M. Naor, B. Pinkas, and O. Reingold. *Distributed pseudo -random functions and KDCs*. In J. Stern, ed., *Proceedings: Advances in Cryptology - EUROCRYPT 1999*, vol. 1592 of *Lecture Notes in Computer Science*, pp. 327–346 (Springer, Prague, Czech Republic, 1999).

[99] Z. Liu, Z. Cao, Q. Huang, D. S. Wong, and T. H. Yuen. *Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles*. In V. Atluri and C. Diaz, eds., *Proceeedings: European Symposium on Research in Computer Security - ESORICS 2011*, vol. 6879 of *Lecture Notes in Computer Scienc*, p. 278297 (Springer, Leuven, Belgium, 2011).

[100] A. Lewko and B. Waters. *Decentralizing attribute-based encryption*. In K. G. Paterson, ed., *Proceedings: Advances in Cryptology - EUROCRYPT 2011*, vol. 6632 of *Lecture Notes in Computer Science*, pp. 568–588 (Springer, Tallinn, Estonia, 2011).

[101] J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie. *Multi-authority ciphertext-policy attribute-based encryption with accountability*. In *Proceedings: ACM Symposium on Information, Computer and Communications Security - ASIACCS 2011*, pp. 386–390 (ACM, 2011).

[102] S. Abriola, M. E. Descotte, and S. Figueira. *Model theory of xpath on data trees. part II: binary bisimulation and definability*. Inf. Comput. **255**, 195 (2017). URL https://doi.org/10.1016/j.ic.2017.01.002.

[103] V. Goyal, O. Pandey, A. Sahai, and B. Waters. *Attribute-based encryption for fine-grained access control of encrypted data*. In *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006*, pp. 89–98 (2006). URL http://doi.acm.org/10.1145/1180405.1180418.

[104] A. N. Ravari, J. H. Jafarian, M. Amini, and R. Jalili. *GTHBAC: A generalized temporal history based access control model*. Telecommunication Systems **45**(2-3), 111 (2010). URL http://dx.doi.org/10.1007/s11235-009-9239-9.

[105] S.Kleene. *Introduction to metamathematics.* (D. Van Nostrand, Princeton, NJ, 1950).

[106] W. H. Jobe. *Functional completeness and canonical forms in many-valued logics*. J. Symb. Log. **27**(4), 409 (1962). URL http://dx.doi.org/10.2307/2964548.

[107] D. Bitton, D. J. DeWitt, and C. Turbyfill. *Benchmarking database systems A systematic approach*. In *9th International Conference on Very Large Data Bases, October 31 - November 2, 1983, Florence, Italy, Proceedings*, pp. 8–19 (1983). URL http://www.vldb.org/conf/1983/P008.PDF.

[108] J. Byun and N. Li. *Purpose based access control for privacy protection in relational database systems*. VLDB J. **17**(4), 603 (2008). URL http://dx.doi.org/10.1007/s00778-006-0023-0.

[109] L. Lin, J. Hu, and J. Zhang. *Packet: a privacy-aware access control policy composition method for services composition in cloud environments*. Frontiers of Computer Science **10**(6), 1142 (2016). URL http://dx.doi.org/10.1007/s11704-016-5503-9.

[110] A. Banerjee and D. A. Naumann. *History-based access control and secure information flow*. In *Construction and Analysis of Safe, Secure, and Interoperable Smart Devices, International Workshop, CASSIS 2004, Marseille, France, March 10-14, 2004, Revised Selected Papers*, pp. 27–48 (2004). URL http://dx.doi.org/10.1007/978-3-540-30569-9_2.

[111] K. Krukow, M. Nielsen, and V. Sassone. *A logical framework for history-based access control and reputation systems*. Journal of Computer Security **16**(1), 63 (2008). URL http://content.iospress.com/articles/journal-of-computer-security/jcs299.

[112] W. Diffie and M. E. Hellman. *New directions in cryptography*. IEEE Transactions on Information Theory **IT-22**(6), 644 (1976).

[113] C. Rackoff and D. R. Simon. *Non-interactive zero-knowledge proof of knowledge and chosen ciphertext attack*. In J. Feigenbaum, ed., *Proceedings: Advances in Cryptology - CRYPTO 1991*, vol. 576 of *Lecture Notes in Computer Science*, pp. 129–140 (Springer, Santa Barbara, California, USA, 1992).

[114] T. ElGamal. *A public key cryptosystem and a signature scheme based on discrete logarithms*. IEEE Transactions on Information Theory **IT-31**(4), 469 (1985).

[115] R. L. Rivest, A. Shamir, and L. M. Adleman. *A method for obtaining digital signatures and public-key cryptosystems*. Communications of the ACM **21**(2), 121 (1978).

[116] R. Cramer and V. Shoup. *A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack*. In H. Krawczyk, ed., *Proceedings: Advances in Cryptology - CRYPTO 19998*, vol. 1462 of *Lecture Notes in Computer Science*, pp. 13–25 (Springer, Santa Barbara, California, USA, 1998).

[117] E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern. *Rsa-oaep is secure under the rsa assumption*. In J. Kilian, ed., *Proccedings: Advances in Cryptology - CRYPTO 2001*, vol. 2139 of *Lecture Notes in Computer Science*, pp. 260–274 (Springer, Santa Barbara, California, USA, 2001).

[118] S. Goldwasser, S. Micali, and R. L. Rivest. *A digital signature scheme secure against adaptive chosen-message attacks*. SIAM Journal on Computing **17**(2), 281 (1988).

[119] A. M. Odlyzko. *Discrete logarithms in finite fields and their cryptographic significance*. In T. Beth, N. Cot, and I. Ingemarsson, eds., *Proceedings: Advances in Cryptology - CRYPTO 1984*, vol. 209 of *Lecture Notes in Computer Science*, pp. 224–314 (Springer, Paris, France, 1985).

[120] U. M. . Maurer. *Towards the equivalence of breaking the diffie-hellman protocol and computing discrete logarithms*. In *Proceedings: Advances in Cryptology - CRYPTO*

1994, vol. 839 of *Lecture Notes in Computer Science*, pp. 271–281 (Springer, Santa Barbara, California, USA, 1994).

[121] D. Boneh. *The decision diflie-hellman problem*. In J. P. Buhler, ed., *Proceedings: Algorithmic Number Theory - ANT 1998*, vol. 1423 of *Lecture Notes in Computer Science*, pp. 48–63 (Springer, Portland, Oregon, USA, 1998).

[122] J. Groth and A. Sahai. *Efficient non-interactive proof systems for bilinear groups*. In *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, pp. 415–432 (2008). URL https://doi.org/10.1007/978-3-540-78967-3_24.

[123] D. Boneh, X. Boyen, and E. Goh. *Hierarchical identity based encryption with constant size ciphertext*. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pp. 440–456 (2005). URL http://dx.doi.org/10.1007/11426639_26.

[124] Z. Wan, J. Liu, and R. H. Deng. *HASBE: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing*. IEEE Trans. Information Forensics and Security **7**(2), 743 (2012). URL http://dx.doi.org/10.1109/TIFS.2011.2172209.

[125] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic. *Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility*. Future Generation Comp. Syst. **25**(6), 599 (2009). URL http://dx.doi.org/10.1016/j.future.2008.12.001.

[126] A. Sahai and B. Waters. *Fuzzy identity-based encryption*. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pp. 457–473 (2005). URL http://dx.doi.org/10.1007/11426639_27.

[127] R. Ostrovsky, A. Sahai, and B. Waters. *Attribute-based encryption with non-monotonic access structures*. IACR Cryptology ePrint Archive **2007**, 323 (2007). URL http://eprint.iacr.org/2007/323.

[128] G. Wang, Q. Liu, J. Wu, and M. Guo. *Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers*. Computers & Security **30**(5), 320 (2011). URL http://dx.doi.org/10.1016/j.cose.2011.05.006.

[129] R. Bobba, H. Khurana, and M. Prabhakaran. *Attribute-sets: A practically motivated enhancement to attribute-based encryption*. IACR Cryptology ePrint Archive **2009**, 371 (2009). URL http://eprint.iacr.org/2009/371.

[130] S. D. Galbraith, K. G. Paterson, and N. P. Smart. *Pairings for cryptographers*. Discrete Applied Mathematics **156**(16), 3113 (2008).

[131] B. Lynn. *The pairing-based cryptography (PBC) library* (2006). Http://crypto.stanford.edu/pbc/.

[132] D. R. L. Brown. *Standards for Efficient Cryptography SEC 2: Recommended Elliptic Curve Domain Parameters*. Certicom Research, 2.0 ed. (2010). Http://www.secg.org/download/aid-784/sec2-v2.pdf.

[133] J. Carter and M. N. Wegman. *Universalclasses of hashfunctions*. Journal of Computer and System Sciences **18**(2), 143 (1979).

[134] W. Mao. *Moden Cyptography Theory & Practice* (Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, USA, 2003).

[135] M. Bellare and P. Rogaway. *Random oracles are practical: A paradigm for designing efficient protocols*. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, eds., *Proceedings: ACM conference on Computer and communications security - CCS 1993*, pp. 62–73 (ACM, Fairfax, VA, USA, 1993).

[136] M. Bellare, R. Canetti, and H. Krawczyk. *Keying hash functions for message authentication*. In N. Koblitz, ed., *Proceedings: Advances in Cryptology - CRYPTO 1996*, vol. 1109 of *Lecture Notes in Computer Science*, pp. 1–15 (Springer, Santa Barbara, California, USA, 1996).

[137] R. Canetti, O. Goldreich, and S. Halevi. *The random oracle methodology, revisited*. In J. S. Vitter, ed., *Proceedings: ACM Symposium on the Theory of Computing - STOC 1998*, pp. 209–218 (ACM, Dallas, Texas, USA, 1998).

[138] A. Beimel. *Secure schemes for secret sharing and key distribution*. PhD thesis (1996).